MANUAL TÉCNICO

NATALIA REY LOROÑO

ROBERTO REY RODRÍGUEZ

ÁLVARO LÓPEZ CASAS

PROGRAMACIÓN DE SERVICIOS Y PROCESOS

2º DAM

ÍNDICE

APLICACIÓN SERVIDOR

CLASE HILO CLIENTE

Clase para los hilos de los clientes.

ATRIBUTOS

- socket (`Socket`): Socket que se utiliza para comunicarse con el cliente.
- objectOutputStream (`ObjectOutputStream`): Stream con el que se envían objetos al servidor.
- objectInputStream (`ObjectInputStream`): Stream con el que se reciben objetos del servidor.
- server (`Servidor`): Servidor al que pertenece este hilo.
- identificador (`String`): Identificador único del cliente con el que este hilo se comunica.
- escuchando (`boolean`): Almacena true cuando está escuchando.

MÉTODOS

HILOCLIENTE

```
public HiloCliente(Socket socket, Servidor server)
```

Parámetros:

- socket (Socket): Socket de comunicación con el cliente
- server (Servidor): Servidor que gestiona este cliente

Explicación del método:

Constructor que inicializa los streams de entrada y salida para la comunicación cliente-servidor.

DESCONECTAR

```
public void desconnectar()
```

Explicación del método:

Método encargado de cerrar el socket con el que se está comunicando.

RUN

```
public void run()
```

Explicación del método:

Método principal del hilo que ejecuta el método escuchar en un ciclo y finaliza cerrando la conexión.

ESCUCHAR

```
public void escuchar()
```

Explicación del método:

Método que escucha continuamente los mensajes enviados por el cliente y los procesa.

EJECUTAR

```
public void ejecutar(LinkedList<String> lista)
```

Parámetros:

- lista (LinkedList<String>): Lista de mensajes o comandos recibidos.

Explicación del método:

Procesa los comandos recibidos y ejecuta las acciones correspondientes (conexión, desconexión o envío de mensajes).

ENVIARMENSAJE

```
private void enviarMensaje(LinkedList<String> lista)
```

Parámetros:

- lista (LinkedList<String>): Lista que contiene los datos del mensaje.

Explicación del método:

Envía un mensaje al cliente a través del socket.

CONFIRMARCONEXION

```
private void confirmarConexion(String identificador)
```

Parámetros:

- Identificador (String): Identificador del cliente que solicita la conexión.

Explicación del método:

Confirma la conexión de un nuevo cliente, lo agrega a la lista de usuarios conectados y notifica a los demás clientes.

GETIDENTIFICADOR

```
public String getIdentificador()
```

Return:

- String: Identificador único del cliente

Explicación del método:

Retorna el identificador único asignado al cliente.

CONFIRMARDESCONEXIÓN

```
private void confirmarDesConexion()
```

Explicación del método:

Notifica a los clientes conectados que un cliente se ha desconectado, elimina al cliente de la lista del servidor y cierra la conexión.

CLASE SERVIDOR

Clase en la que se maneja la comunicación del lado del servidor.

ATRIBUTOS

- serverSocket (`ServerSocket`): Socket servidor que escucha conexiones de nuevos clientes.
- clientes (`LinkedList<HiloCliente>`): Lista de hilos de comunicación para gestionar a cada cliente conectado.
- ventana (`VentanaS`): Ventana que gestiona la interfaz gráfica del servidor.
- puerto (`String`): Puerto que el servidor utiliza para escuchar conexiones.
- correlativo (`int`): Valor estático para diferenciar a los múltiples clientes que se conectan al servidor.

MÉTODOS

SERVIDOR

```
public Servidor(String puerto, VentanaS ventana)
```

Parámetros:

- Puerto (String): Puerto en el que el servidor escuchará las conexiones
- Ventana (VentanaS): Ventana que muestra la interfaz gráfica del servidor

Explicación del método:

Constructor que inicializa el servidor, configura los atributos y arranca el hilo.

RUN

```
public void run()
```

Explicación del método:

Método principal del servidor que escucha permanentemente nuevas conexiones y crea un hilo para cada cliente que se conecta. Maneja errores de conexión y muestra alertas en caso de fallo.

GETUSUARIOSCONECTADOS

```
public LinkedList<String> getUsuariosConectados()
```

Return:

- LinkedList<String>: Lista con los identificadores de todos los clientes conectados.

Explicación del método:

Devuelve una lista con los identificadores de los clientes actualmente conectados al servidor.

AGREGARLOG

`public void agregarLog(String texto)`

Parámetros:

- Texto (String): Línea de texto a agregar al log de la interfaz gráfica.

Explicación del método:

Agrega una línea de texto al log de la ventana de la interfaz gráfica del servidor.

CLASE VENTANAS

Clase en la que se maneja la comunicación del lado del servidor.

ATRIBUTOS

- DEFAULT_PORT (`String`): Puerto predeterminado que usa el servidor para la conexión (valor por defecto: "10101").
- servidor (`Servidor`): Instancia del servidor que maneja la comunicación.
- jScrollPane1 (`JScrollPane`): Panel de desplazamiento que contiene el área de texto del log.
- txtClientes (`JTextArea`): Área de texto donde se muestra el log del servidor.

MÉTODOS

VENTANAS

`public VentanaS()`

Explicación del método:

Constructor que inicializa los componentes gráficos de la ventana, configura el comportamiento de cierre y crea una instancia del servidor.

INITCOMPONENTS

`private void initComponents()`

Explicación del método:

Configura y organiza los elementos gráficos de la ventana (layout, título, panel de desplazamiento, y área de texto del log).

MAIN

`public static void main(String[] args)`Return:

Explicación del método:

Punto de entrada de la aplicación. Configura el tema gráfico y crea la ventana del servidor.

AGREGARLOG

`public void agregarLog(String texto)`

Parámetros:

- Texto (String): Línea de texto que se agrega al log.

Explicación del método:

Agrega una línea de texto al área de log para registrar actividades o eventos del servidor.

GETPUERTO

```
private String getPuerto()
```

Return:

- String: El puerto ingresado por el usuario o el valor predeterminado si no se especifica.

Explicación del método:

Solicita al usuario que ingrese el puerto a través de un cuadro de diálogo. Si el usuario no confirma, la aplicación se cierra.

ADDSERVIDORINICIADO

```
public void addServidorIniciado()
```

Explicación del método:

Agrega un mensaje al log indicando que el servidor se ha inicializado correctamente.

APLICACIÓN CLIENTE

CLASE CLIENTE

Clase para gestionar el cliente.

ATRIBUTOS

- socket (`Socket`): Socket utilizado para comunicarse con el servidor.
- objectOutputStream (`ObjectOutputStream`): Stream para enviar objetos al servidor.
- objectInputStream (`ObjectInputStream`): Stream para recibir objetos del servidor.
- ventana (`VentanaC`): Ventana que representa la interfaz gráfica del cliente.
- identificador (`String`): Identificador único del cliente dentro del chat.
- escuchando (`boolean`): Bandera que indica si el cliente está escuchando mensajes del servidor.
- host (`String`): Dirección IP del servidor.
- puerto (`int`): Puerto donde el servidor escucha las conexiones de los clientes.

MÉTODOS

CLIENTE

```
Public Cliente(VentanaC ventana, String host, Integer puerto, String nombre)
```

Parámetros:

- ventana (VentanaC): Ventana de la interfaz gráfica.
- host (String): Dirección IP del servidor.
- puerto (Integer): Puerto donde se realiza la conexión.
- nombre (String): Nombre o identificador del cliente.

Explicación del método:

Constructor que inicializa los atributos, establece el estado inicial del cliente y arranca el hilo.

RUN

```
public void run()
```

Explicación del método:

Establece la conexión con el servidor, inicializa los streams de comunicación y comienza a escuchar mensajes del servidor.

DESCONECTAR

```
public void desconectar()
```

Explicación del método:

Cierra el socket, los streams de comunicación y detiene el estado de escucha del cliente.

ENVIARMENSAJE

```
public void enviarMensaje(String cliente_receptor, String mensaje)
```

Parámetros:

- cliente_receptor (String): Identificador del cliente receptor del mensaje.
- mensaje (String): Contenido del mensaje a enviar.

Explicación del método:

Envía un mensaje al servidor con el formato adecuado, indicando el destinatario y el contenido.

ESCUCHAR

```
public void escuchar()
```

Explicación del método:

Escucha continuamente los mensajes enviados por el servidor y los procesa según el tipo de mensaje recibido.

EJECUTAR

```
public void ejecutar(LinkedList<String> lista)
```

Parámetros:

- lista (LinkedList<String>): Lista que contiene los datos del mensaje recibido.

Explicación del método:

Procesa los mensajes recibidos desde el servidor y realiza las acciones correspondientes (conexión, desconexión, nuevo mensaje, etc.).

ENVIARSOLICITUDCONEXION

```
private void enviarSolicitudConexion(String identificador)
```

Parámetros:

- identificador (String): Identificador del cliente que solicita conectarse al servidor.

Explicación del método:

Envía una solicitud de conexión al servidor para agregar al cliente a la lista de clientes conectados.

CONFIRMARDESCONEXION

```
Public void confirmarDesconexion()
```

Explicación del método:

Envía una notificación al servidor para indicar la desconexión del cliente y permitir que el servidor lo elimine de sus listas.

GETIDENTIFICADOR

```
Public String getIdentificador()
```

Return:

- String: Identificador único del cliente.

Explicación del método:

Retorna el identificador único asignado al cliente.

CLASE VENTANAC

Clase que maneja la interfaz gráfica del cliente.

ATRIBUTOS

- btnEnviar (`JButton`): Botón para enviar mensajes al servidor.
- cmbContactos (`JComboBox`): ComboBox que contiene la lista de contactos disponibles para el cliente.
- jLabel1 (`JLabel`): Etiqueta que indica el destinatario del mensaje.
- jScrollPane1 (`JScrollPane`): Panel de desplazamiento que contiene el historial de mensajes.
- txtHistorial (`JTextArea`): Área de texto donde se muestra el historial de mensajes.
- txtMensaje (`JTextField`): Campo de texto donde el cliente escribe su mensaje.
- DEFAULT_PORT (`String`): Puerto predeterminado para la conexión (valor por defecto: "10101").
- DEFAULT_IP (`String`): Dirección IP predeterminada del servidor (valor por defecto: "127.0.0.1").
- cliente (`Cliente`): Instancia del cliente asociado a la ventana

MÉTODOS

VENTANAC

```
public VentanaC()
```

Explicación del método:

Constructor que inicializa los componentes de la ventana, solicita los datos de configuración (IP, puerto y nombre) y crea una instancia del cliente.

INITCOMPONENTES

```
private void initComponents()
```

Explicación del método:

Configura y organiza los elementos gráficos de la ventana, incluyendo el historial de mensajes, campo de texto, comboBox de contactos y botón de envío.

BTNENVIARACTIONPERFORMED

```
private void btnEnviarActionPerformed(ActionEvent evt)
```

Parámetros:

- evt (ActionEvent): Evento de acción generado al hacer clic en el botón de enviar.

Explicación del método:

Envía un mensaje al servidor utilizando el cliente, actualiza el historial con el mensaje enviado y limpia el campo de texto.

FORMWINDOWCLOSING

```
private void formWindowClosing(WindowEvent evt)
```

Parámetros:

- evt (WindowEvent): Evento de cierre de ventana.

Explicación del método:

Notifica al servidor que el cliente se ha desconectado antes de cerrar la ventana.

MAIN

```
public static void main(String[] args)
```

Explicación del método:

Punto de entrada de la aplicación. Configura el tema gráfico y muestra la ventana del cliente.

ADDCONTACTO

```
public void addContacto(String contacto)
```

Parámetros:

- contacto (String): Identificador del contacto a agregar.

Explicación del método:

Agrega un nuevo contacto a la lista del comboBox de contactos.

ADDMENSAJE

```
public void addMensaje(String emisor, String mensaje)
```

Parámetros:

- emisor (String): Identificador del remitente del mensaje.
- mensaje (String): Contenido del mensaje.

Explicación del método:

Agrega un nuevo mensaje al historial de la conversación, indicando el emisor y el contenido.

SESIONINICIADA

```
public void sesionIniciada(String identificador)
```

Parámetros:

- identificador (String): Identificador del cliente.

Explicación del método:

Configura el título de la ventana para indicar la sesión del cliente.

GETIPPUERTONOMBRE

```
private String[] getIPPuertoNombre()
```

Return:

- String[]: Array que contiene la IP, el puerto y el nombre del cliente.

Explicación del método:

Solicita al usuario los datos de configuración de conexión (IP, puerto y nombre) mediante un cuadro de diálogo.

ELIMINARCONTACTO

```
Public void eliminarContacto(String identificador)
```

Parámetros:

- identificador (String): Identificador del contacto a eliminar.

Explicación del método:

Elimina un contacto del comboBox de contactos cuando un cliente cierra sesión.