***Proyecto de Chat - Documento Teórico***

Este documento describe de manera teórica los componentes y funcionalidades del proyecto de chat desarrollado.

**Objetivo del Proyecto**

El objetivo del proyecto es crear una aplicación de chat que permita a varios usuarios comunicarse entre ellos, ya sea de forma global o privada, utilizando **Sockets** y una arquitectura cliente-servidor.

**Componentes Principales**

**1. Servidor**

El servidor gestiona las conexiones de los clientes, distribuye los mensajes y mantiene un registro de los usuarios conectados.

**Características principales:**

* **Gestor de Conexiones:** Acepta conexiones entrantes de los clientes y crea un hilo independiente para cada uno.
* **Mensajería Global:** Permite a los clientes enviar mensajes que todos los usuarios conectados pueden ver.
* **Mensajería Privada:** Facilita la comunicación entre usuarios de forma individual.
* **Notificaciones:** Informa a todos los clientes cuando un usuario se conecta o desconecta.

**Clases Relacionadas:**

* **Servidor:** La clase principal que inicia el servidor, acepta conexiones y gestiona los usuarios.
* **ManejadorCliente:** Ejecutada en un hilo para cada cliente, recibe y procesa mensajes enviados al servidor.

**2. Cliente**

El cliente permite a los usuarios conectarse al servidor, enviar mensajes y gestionar comunicaciones privadas.

**Características principales:**

* **Interfaz Gráfica:** Proporciona una ventana para enviar y recibir mensajes.
* **Configuración de Conexión:** Solicita al usuario los datos de conexión (IP, puerto y nombre).
* **Gestor de Mensajes:** Procesa mensajes recibidos desde el servidor y los distribuye a ventanas globales o privadas.
* **Mensajería Privada:** Abre ventanas independientes para cada conversación privada.

**Clases Relacionadas:**

* **Cliente:** La clase principal que gestiona la conexión y la interfaz gráfica.
* **VentanaPrivada:** Representa las ventanas emergentes utilizadas para mensajes privados.

**3. Utilidades (EstiloUI)**

Define el estilo visual del proyecto para garantizar una apariencia uniforme y profesional.

**Características principales:**

* Colores personalizados para fondos, botones y fuentes.
* Estilo unificado para scrolls, botones y ventanas.

**Clases Relacionadas:**

* **EstiloUI:** Clase utilitaria con métodos para aplicar estilos a componentes gráficos.

**Flujo de Trabajo**

1. **Inicio del Servidor:**
   * El servidor se inicia en un puerto definido (por defecto, 10101).
   * Espera conexiones entrantes de clientes.
2. **Conexión del Cliente:**
   * El cliente solicita la IP del servidor, el puerto y el nombre del usuario.
   * Una vez conectado, envía el nombre al servidor para registrarse.
3. **Envío de Mensajes:**
   * **Mensajes Globales:** Los clientes pueden enviar mensajes que el servidor reenvía a todos los usuarios conectados.
   * **Mensajes Privados:** Los clientes pueden seleccionar un destinatario para enviar un mensaje privado. Estos mensajes no son visibles en el chat global.
4. **Recepción de Mensajes:**
   * Los mensajes globales se muestran en la ventana principal del cliente.
   * Los mensajes privados abren (o actualizan) ventanas independientes para cada destinatario.
5. **Desconexión:**
   * Cuando un cliente se desconecta, el servidor notifica a los usuarios conectados y actualiza la lista de usuarios disponibles.

**Consideraciones Técnicas**

1. **Sockets TCP:**
   * Se utilizan para la comunicación entre clientes y el servidor.
   * Garantizan una conexión fiable y ordenada.
2. **Hilos:**
   * Cada cliente conectado al servidor se gestiona en un hilo independiente para permitir comunicaciones simultáneas.
3. **Interfaz Gráfica:**
   * Basada en Swing, permite a los usuarios interactuar de manera intuitiva con la aplicación.
   * Diseño modular que facilita la extensión de funcionalidades.
4. **Validaciones:**
   * Los datos de conexión (IP, puerto y nombre) son validados para evitar errores.
   * Los mensajes privados verifican que el destinatario esté conectado antes de enviarse.

**Futuras Mejoras**

* **Cifrado de Mensajes:** Implementar un sistema de cifrado para garantizar la privacidad de las comunicaciones.
* **Soporte para Emojis:** Permitir a los usuarios enviar emojis en los mensajes.
* **Historial de Conversaciones:** Guardar mensajes para que puedan consultarse después de cerrar la aplicación.
* **Notificaciones:** Agregar notificaciones de escritorio cuando se reciban mensajes nuevos.

**Conclusión**

El proyecto de chat implementa con éxito una arquitectura cliente-servidor usando sockets TCP. La aplicación es escalable y ofrece un diseño modular que facilita futuras mejoras y personalizaciones. Gracias a su diseño robusto, es ideal para entender los principios fundamentales de la programación en red y el desarrollo de interfaces gráficas.