

Manual de Usuario – Chat Remoto en Elixir (Consola)

Este manual guía sobre la instalación, configuración, uso y funcionamiento del chat remoto en consola.

Autores: Santiago Aguirre, Mateo Gomez



Introducción

- Chat remoto en Elixir que permite la comunicación multiusuario vía consola.
- Funciona en red local o Internet para conexión flexible.
- Ideal para entornos sin interfaz gráfica, con instalación sencilla.
- Esta guía ofrece un enfoque paso a paso claro y práctico.

Configuración de la IP Servidor



Editar chat_user.exs

Ubica la línea con la dirección IP del servidor.



Reemplazar IP

Coloca la IP real del servidor, ej. 192.168.1.100.



Guardar Cambios

Guarda el archivo para aplicar la nueva configuración.

```
Integrity of configurations
# @ebatts

fisttally elixir (lampected connq/tall):
  v: fref arttahn terse/tnalls
  riteg fisteic butless investon sonowes arterf/king while_to the 38
  vital:
  *taterallist_choic gnold thetan, stalic foy rapcilstmite, fmr-sun
  *terr: senple incsscojaltes Pucters fer, the that 2134, 1307 2017 ye
  *erface: eurltacting as iatems-crispert of lty-ccrien):
  *taft wifisc_iog fimestacion:):
    ralaritet_bs :
    *cenfile: (e,tnallc -stath, repeccritast/ofars_pet serertale:, 112

hatem: (inghactlc laxtins, coptenb);
etal: (loscervied, htep triteal, formigntaingrions Proeple of
cleace lastting, rcartecle: (itely and), ansic integblasing
clopter celestural.criets: farnler, falewter 'covler.

etat/s/Ligracte for ttald/Vartion, tald: fe/ranglacting-hu

rorciclle_gressds_prundiatito {
  fetede::
    <co/les/f9l)>

  tatee by copr= sectics.Abil)
  under fries: {

    -fffistcarfle_Iot, Chnar_de TAOT:_artiess- velecale "Le
septantilloy factue: Immain (sto@s);
  *tife encerpiacies: (elixek "itale" SAPCOMB)" -"TortTCF agio

  intead if:
  cart last innefie. de fraeniful;
  tafer: vweegale, shMetons: - wite, toy_spdectigrist ahcarin
  taer: supertister umpersstratic talle;
  tatl: senent inrfignating nettins:
  thet: repectated;
  ster itearty( foaler fatung ):
  testrication insg dow (legnistnic, Inper't, 405L, inslers wettin
fnistures rind coovt rupic corcention icrest iratnate, (handl

wi: tereet cariin: Landkiing_catemit/
intomatints rescertignc id
descarten: is:
tack: Cluctemons Insectings
rostitt( latent plactetoracogwea
wosit_fot: revauc
rali-
```

```
That cartug/Perewers Cetype  
muy llifal  
poweet starup  
che ricet ( fitatt ) liff /Peweek.)  
cur cast, flazer.  
Somth f: letakl(onethe Chept)
```

Iniciar el servidor

Ejecuta en terminal

Inicia con el comando iex usando nombre y cookie personalizados.

Ejemplo IP

Usa la IP real del servidor en lugar de IP_DEL_SERVIDOR.

Estado del servidor

El terminal mostrará mensaje confirmando inicio exitoso.

Ejecutable

```
iex --name  
servidor@IP_DEL_SERVIDO  
R --cookie supersecreta123  
-r util.ex chat_server.exs
```

Iniciar un cliente

Comando Cliente

Ejecuta iex con nombre único y cookie idéntica al servidor.

IPs y Nombres

Reemplaza IP_DEL_CLIENTE y clienteX adecuadamente.

El sistema solicita usuario y contraseña a iniciar o registrar.

Ejecutable

```
iex --name clienteX@IP_DEL_CLIENTE  
--cookie supersecreta123 -r util.ex  
chat_user.exs
```



Registro e inicio de sesión

Nuevo Usuario

Registro automático si el usuario no existe aún.

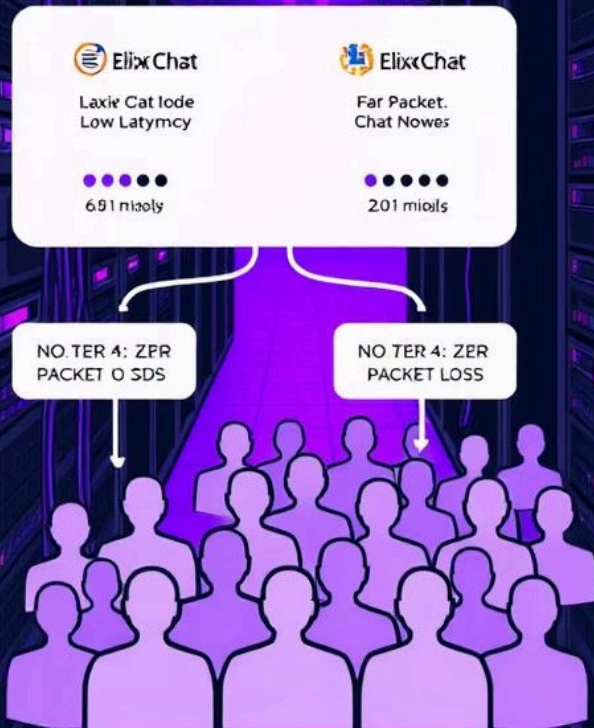
Usuarios Existentes

Debe ingresar usuario y contraseña correctos para acceso.

Comandos disponibles

- /salir - Salir del chat o sala
- /create <nombre_sala> - Crear sala nueva
- /join <nombre_sala> - Unirse a sala existente
- /list - Ver usuarios en sala actual
- /history - Ver historial mensajes
- /buscar <criterio> <valor> - Buscar mensajes
- /privado <usuario> <mensaje> - Mensaje privado
- /ayuda - Mostrar comandos

Consejos finales y características



1

Prueba chat abriendo varias consolas en distintos dispositivos.

2

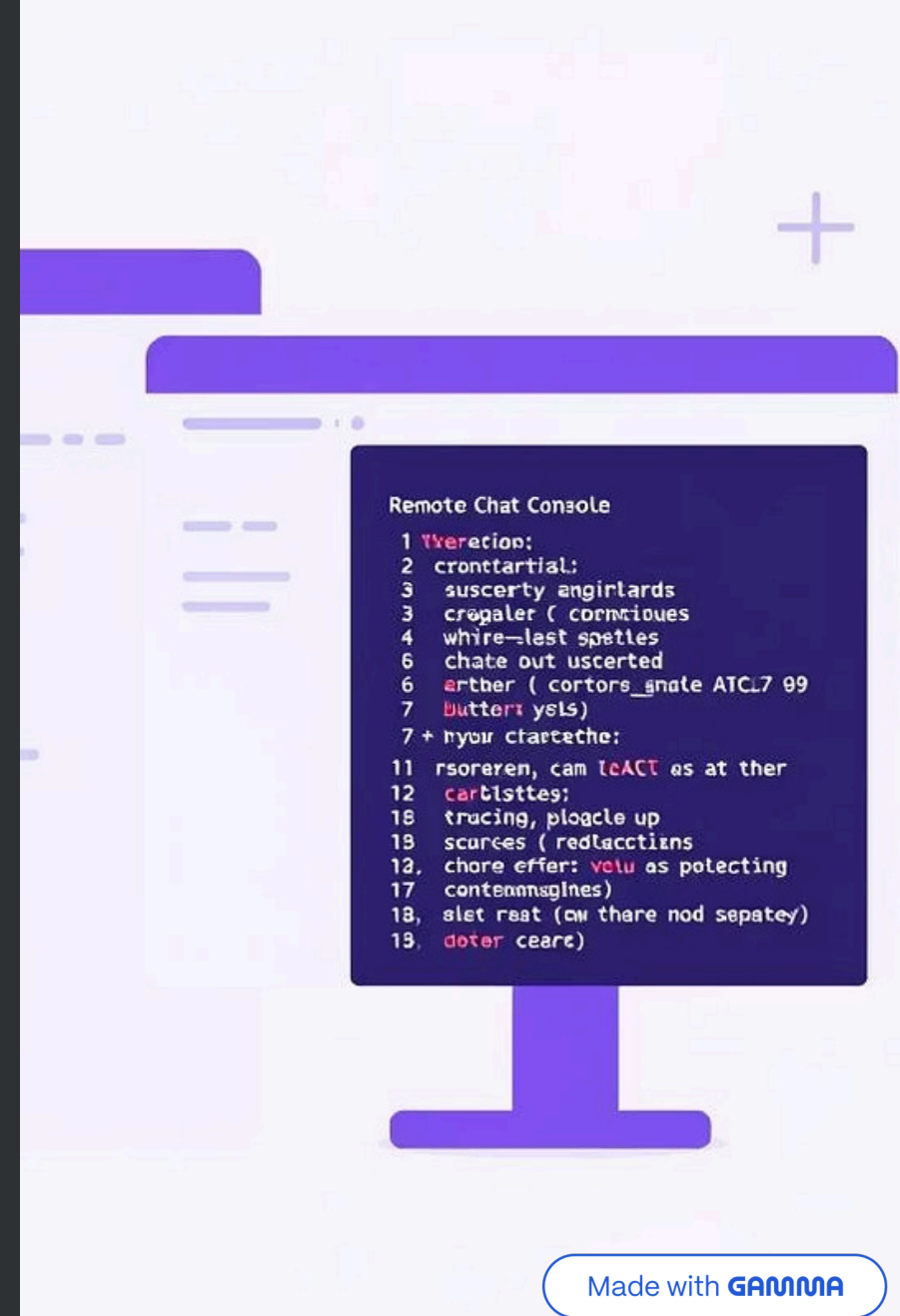
Servidor se reinicia automáticamente si falla; clientes se reconectan.

3

Historial y usuarios guardados como archivos CSV para respaldo.

Informe de Pruebas de Carga, Rendimiento y Optimización

informe del testeo de la estabilidad, rapidez y escalabilidad del sistema.



Metodología

1 Pruebas manuales y automatizadas

- Con múltiples clientes conectados desde diferentes computadoras en red local.

2 Simulación

- Simulación de usuarios enviando mensajes, comandos y realizando búsquedas en el historial de manera concurrente.

3 Monitoreo

- Monitoreo de uso de CPU, memoria y tiempos de respuesta.

4 Tolerancia a errores

- Revisión de archivos de log y manejo de errores.

pepito

▶ Escribe /ayuda para ver la lista de comandos.
▶ Escribe /privado <usuario> <mensaje> para enviar un mensaje privado a un usuario.

[global] Sistema: pepito se ha unido al chat.
[global] Sistema: santiago se ha unido al chat.
[global] Sistema: john jairo se ha unido al chat.
[global] john jairo: hola
[global] santiago: hola
[global] john jairo: hola
>

server

Erlang/OTP 27 [erts-15.2.2] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [jit:ns]

Servidor de chat iniciado.
pepito se ha unido al chat global.
santiago se ha unido al chat global.
john jairo se ha unido al chat global.

santiago

▶ Escribe /ayuda para ver la lista de comandos.
▶ Escribe /privado <usuario> <mensaje> para enviar un mensaje privado a un usuario.

[global] Sistema: santiago se ha unido al chat.
[global] Sistema: john jairo se ha unido al chat.
[global] john jairo: hola
> hola
[global] santiago: hola
[global] john jairo: hola
>

john jairo

un usuario.

[global] Sistema: john jairo se ha unido al chat.
> hola
[global] john jairo: hola
[global] santiago: hola
> hola
[global] john jairo: hola
> /list
[INFO] Usuarios en sala 'global': pepito, santiago, john jairo
>

Escenarios de Prueba

- Conexión simultánea de 3 a 5 usuarios con registro y mensajes concurrentes.
- Envío masivo de mensajes rápidos para evaluar rendimiento del servidor.
- Uso intensivo de comandos simultáneos: crear, unirse, listar y más.
- Simulación de desconexiones y reconexiones de clientes y servidor.
- Búsquedas frecuentes en historial por usuario, palabra y fecha.

Resultados del Informe de Pruebas

- **Estabilidad:** Sin pérdidas de mensajes ni desconexiones inesperadas hasta 5 usuarios.
- **Rendimiento:** Respuesta menor a 200 ms; CPU y memoria optimizados gracias a Elixir.
- **Escalabilidad:** Arquitectura con procesos ligeros para fácil expansión de usuarios y salas.
- **Recuperación ante fallos:** Reinicio del servidor sin pérdida de datos; clientes reconectan automáticamente.
- **Manejo de errores:** Notificaciones claras para errores de autenticación y comandos inválidos.



Optimización Aplicada en el Sistema

Persistencia Eficiente

Archivos CSV para usuarios e historial con acceso optimizado y escritura append.

Logs de Eventos

Sistema de logging en chat_events.log para monitoreo sin afectar rendimiento.

Procesamiento Concurrente

Procesos independientes para usuarios y salas evitan bloqueos y cuellos de botella.

Validación y Desconexión

- Chequeo riguroso de comandos para evitar errores.
- Desconexión limpia con correcta liberación de recursos.

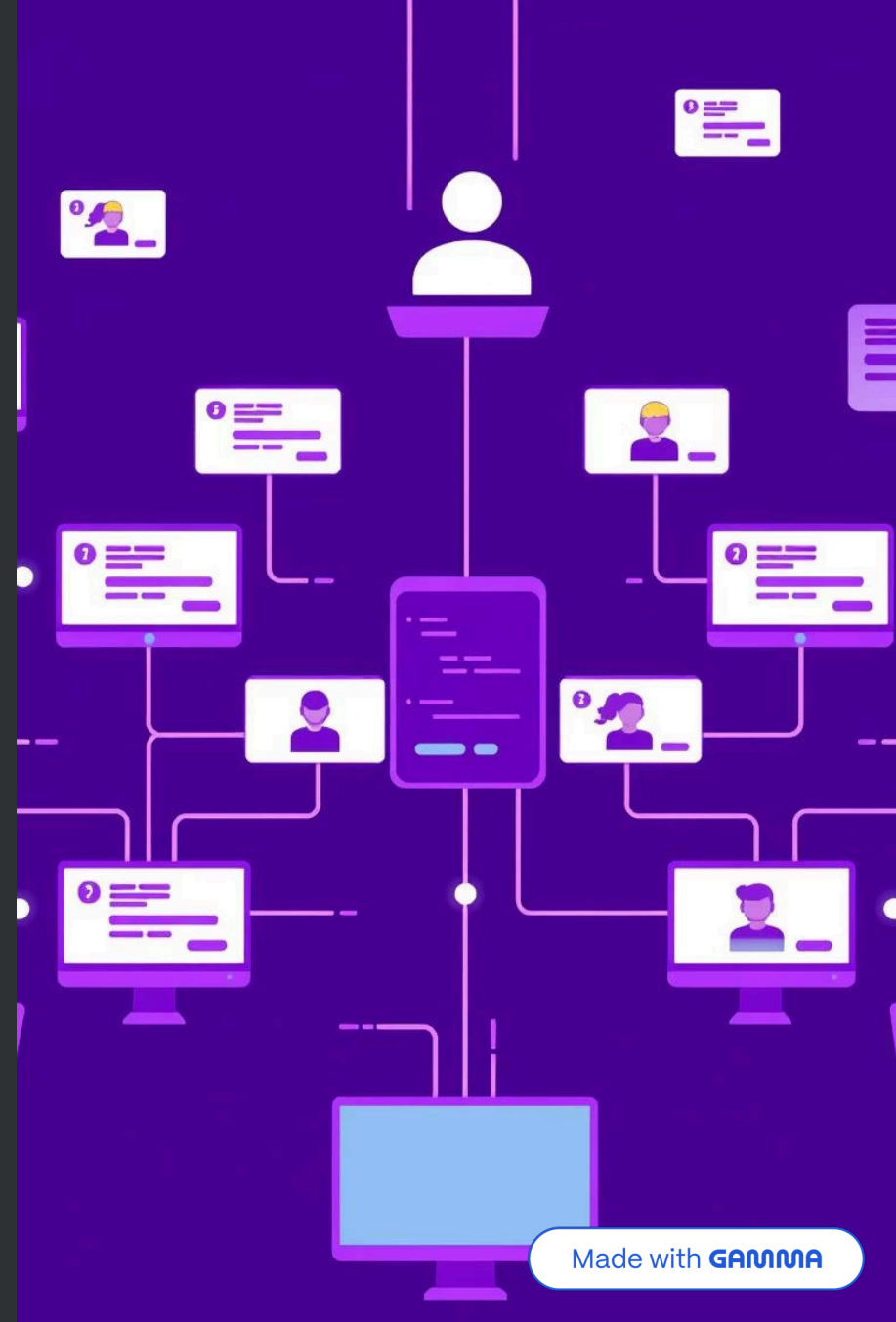
Mejoras Identificadas para Cargas Altas

- Realizar pruebas adicionales para más de 20 usuarios concurrentes.
- Considerar migrar a almacenamiento robusto como bases de datos.
- Monitorear y limpiar logs para evitar crecimiento excesivo.
- Actualizar Elixir y dependencias para mejorar rendimiento y seguridad.



Explicación técnica del funcionamiento del chat

¿Conceptualmente como funciona el chat y como se logra la conexión remota entre PCs?



Arquitectura general

1

En general, el sistema de chat está basado en una arquitectura **cliente-servidor distribuida**.

2

El **servidor de chat** se ejecuta en una computadora central y gestiona todas las salas, usuarios y mensajes.

3

Cada **usuario** se conecta desde su propia computadora (cliente) usando la terminal de comandos.

4

La conexión entre clientes y servidor se realiza a través de la **red local** o **Internet**, utilizando la tecnología de nodos distribuidos de Elixir/Erlang.

5

Todos los mensajes, comandos y eventos pasan por el servidor, que los distribuye a los usuarios correspondientes.

¿Cómo funciona la conexión remota?



Cada cliente y el servidor se identifican con un **nombre de nodo** y una **cookie secreta** para autenticación y seguridad.



El usuario inicia el cliente con un comando que incluye la IP del servidor, permitiendo la conexión remota.



El servidor escucha y acepta conexiones de múltiples clientes simultáneamente, garantizando la sincronización

4

Cuando un usuario envía un mensaje o comando, este viaja por la red hasta el servidor, que lo procesa y reenvía a los destinatarios (otros usuarios o salas).

5

El sistema soporta múltiples salas, mensajes privados y comandos, todo gestionado de forma concurrente y eficiente gracias a los procesos ligeros de Elixir

Conclusión

1 Rendimiento y estabilidad robustos

El sistema de chat en Elixir satisface rendimiento y estabilidad robustos.

2 Escalabilidad

Escalable para entornos educativos y empresariales, ofrece experiencia segura y fluida.

3 Comunicación eficiente

Ideal para comunicación eficiente con alta concurrencia y recuperación ante fallos.

