Chat.erl

Práctica en Grupo – Arquitectura Software

Curso 2016/2017

Alejandro Díaz	alejandro.diazp@udc.es
Dan Fouz	dan.fouz@udc.es
Daniel Moledo	daniel.moledo@udc.es
David Méndez	d.mendez.alvarez@udc.es
Diego Hermida	diego.hermida@udc.es
Jose Dosil	jose.dosilt@udc.es
Roberto Carballa	roberto.carballa@udc.es

Índice

Introducción

El objetivo de la práctica será implementar una aplicación de intercambio de mensajes empleando `Erlang`, un lenguaje funcional, ligero y en tiempo real.
La interfaz será en línea de comandos, coloreando mensajes mediante el uso códigos de escape ANSI.

Funcionalidades

La aplicación cubre una serie de funcionalidades que se detallan a continuación.

Cliente

Como cliente, se podrán ejecutar las siguientes operaciones:

- Unirse a una sala: Un cliente podrá unirse a una sala de un servidor. El servidor avisará a la sala de que hay un nuevo usuario.
- Elegir un *nickname* con el que identificarse. Debe ser único.
- Enviar mensajes: Un cliente podrá enviar mensajes que serán recibidos por todos los usuarios de la sala.
- Cambiar de sala: Un cliente podrá unirse a otra sala en cualquier momento. Desde ese instante dejará de recibir mensajes de la sala previa (sólo se puede estar en una sala a la vez).
- Crear una sala: Un cliente podrá crear una sala, a la cual podrá unirse cualquier otro usuario. El nombre de la sala es único dentro del servidor o *multiserver* (se hablará de este concepto más adelante).
- Listar salas disponibles: Se podrá consultar la lista de salas disponibles en cualquier momento.
- Hacer un *whisper*: Un usuario podrá enviar mensajes a otro sin conocimiento por parte de la sala.
- Desconexión: El usuario podrá salir de la sala en cualquier momento. El servidor avisará a la sala. Además, el *nickname* del usuario quedará disponible para que cualquiera lo use.

Servidor

Como servidor, podrán ejecutarse las siguientes operaciones:

- Creación de salas: De forma análoga a la funcionalidad como cliente.
    - Al iniciar un servidor podrá especificársele, si se desea, que lea un fichero de texto con los nombres de las salas que deben estar disponibles desde el inicio.

- Parar el servidor: Inhabilita la funcionalidad del servidor. Desde ese momento, los usuarios que estén conectados sus salas dejarán de recibir mensajes[1].

Arquitectura

A continuación se mostrará la arquitectura de la aplicación, en sus posibles variantes.

Cliente – servidor básico

Con esta configuración, se crean N servidores que dan servicio a M clientes. Cada servidor es independiente del resto, gestionando sus salas de forma individual. Una representación de esta aproximación es la siguiente:

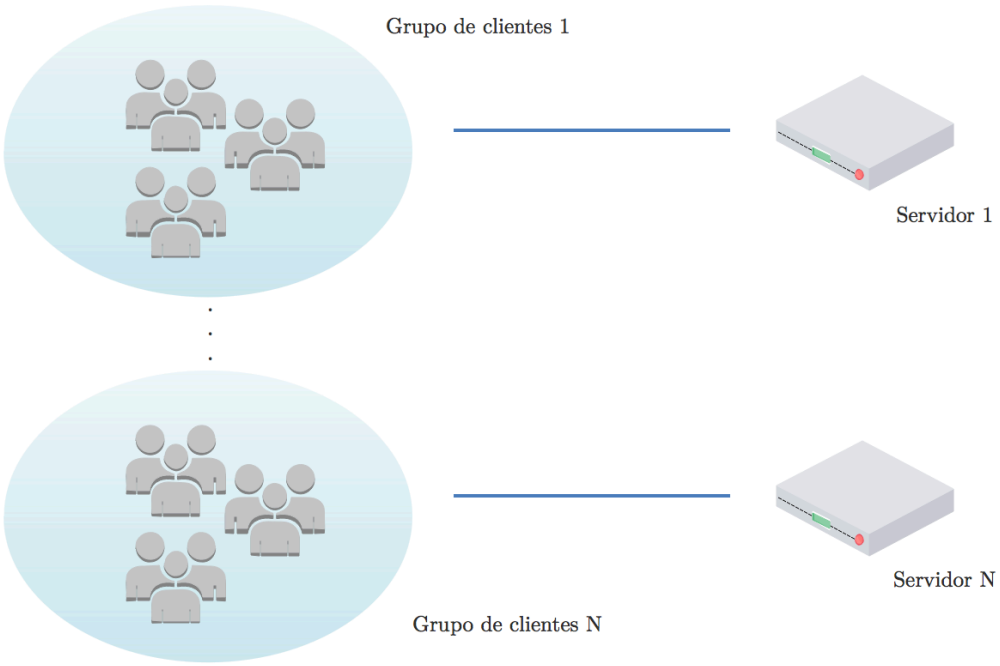

Figura 1: Aplicación con servidores independientes

Multiserver

En esta aproximación, existe un nodo central que actúa como balanceador de carga, junto con una serie de nodos servidor que procesan las peticiones.
Esta aproximación responde al siguiente esquema:

[1] Puede haber casos en los que se sigan recibiendo mensajes, si el cliente puede conectarse a otro servidor.

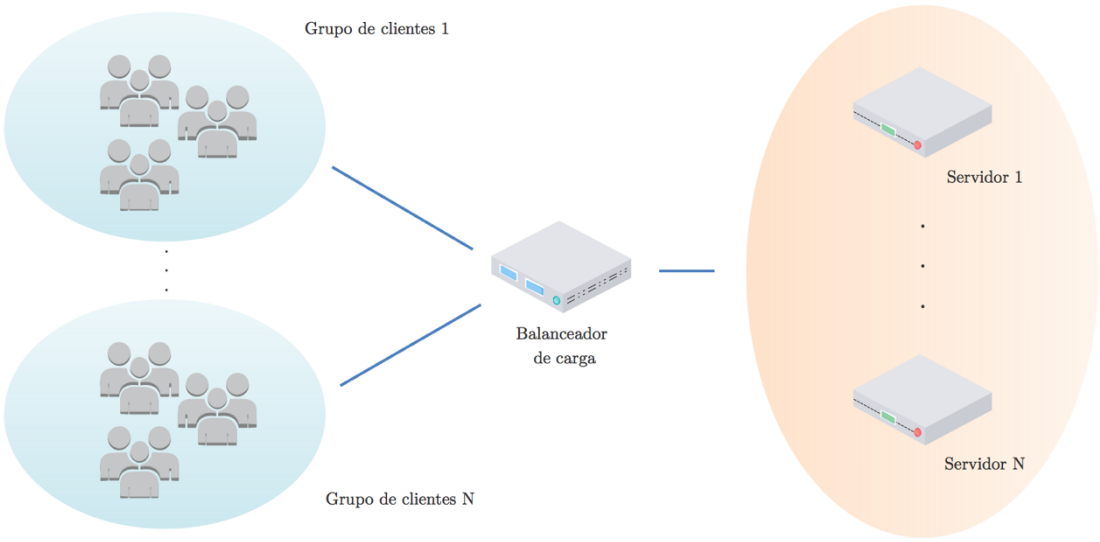

Figura 2: Aplicación multiservidor

Satisfacción de requisitos

- **Escalabilidad**: Mediante el uso de un balanceador de carga, la aplicación puede atender más peticiones y gestionar más salas de forma simultánea, lo cual nos lleva a soportar un mayor número de usuarios a la vez.
- **Disponibilidad**: Dado que no hay límite a la hora de añadir servidores, la aplicación puede garantizar su disponibilidad mediante la adición de nuevos nodos servidor en caso de que sea necesario.
- **Seguridad**: La aplicación incorpora el estándar SSL para el envío de mensajes cifrados entre cliente y servidor.
- **Usabilidad**: El uso del *multiserver* permite mejorar la usabilidad por parte del usuario. En lugar de conectarse a un servidor concreto, se conecta al *multiserver* y éste le asigna uno. Además, un usuario podrá cambiarse de forma transparente a una sala que esté en otro servidor.

Instalación y ejecución

Para compilar la aplicación, se hará lo siguiente:

```
$: cd <directorio raíz de la app>
$: mkdir bin # si no existe
$: make
```

Acto seguido se estará en condiciones de ejecutar la aplicación.

Funcionalidades cliente

1. Iniciar el proceso cliente:
    - `make {run|run_ssl_client}  name=<nombre>@<IP>` (desde directorio raíz de la app)
    - `erl -name <nombre>@<IP>` (desde el directorio `bin`).

   NOTA: El nombre debe ser único en caso de que se creen varios procesos cliente en la misma máquina (misma IP o *hostname*).

2. Conectarse a un servidor (debe haber un servidor funcionando, si no dará un error):
    - Deberá iniciarse el proceso desde la shell: `client:start()`
    - Una vez iniciado el proceso cliente, se pedirá por teclado que se introduzca un servidor al que conectarse. Aquí se introducirá el identificador del servidor o *multiserver*: `<nombre_server>@<IP>`
    - Al conectarse a un servidor, se intentará acceder a la sala por defecto, la sala `main`.
3. Elegir un *nickname*:
    - Se pedirá por teclado un nombre. Debe ser único para todo el servidor, si no saltará un error.
    - Si todo ha ido bien, se recibirá un mensaje indicando que el usuario se ha unido correctamente.
4. Enviar mensajes:
    - Para enviar un mensaje, simplemente se introducirá el texto por teclado y se pulsará [Enter].
    - El mensaje se enviará a toda la sala (*broadcast*).
5. Cambiar de sala:
    - Para cambiarse de sala, introducirá: `/join <nombre_sala>` (la sala debe existir, si no, lanzará un error).
6. Crear una sala: `/create <nombre_sala>`.
7. Listar las salas: `/available_rooms`
8. Hacer un *whisper*:
    - Enviará un mensaje a un usuario dentro de la sala, pero sin que el resto de miembros de la sala lo vean.
    - Para hacerlo, se introducirá: `/whisper <user> <message>`
9. Abandono de la sala actual:
    - El cliente abandonará la sala actual y se unirá a la sala `main` o sala principal. Para ello, se introducirá: `/leave`
10. Desconexión del servidor:
    - Además de cerrar la conexión, libera el *nickname* de la lista de nombres ocupados. Deberá introducirse: `/exit`

Funcionalidades servidor

1. Iniciar el nodo[2] servidor:
    - `make {run|run_ssl_server} name=<nombre>@<IP>` (desde directorio raíz de la app)
    - `erl -name <nombre>@<IP>` (desde el directorio `bin`).

    NOTA: Es importante que la IP sea correcta. El token `<nombre>@<IP>` es lo que se les pedirá a los clientes para conectarse al nodo servidor.

2. Arrancar el servidor: `server:start()`
    - En caso de que exista un fichero `rooms.txt`, la aplicación creará automáticamente las salas leyendo el fichero secuencialmente (cada línea es el nombre de una sala).
    - Si no existe el fichero, sólo se incluye la sala por defecto, `main`.
3. Parar el servidor: `server:stop()`
    - En la mayoría de los casos, inhabilita el envío de mensajes en todas las salas gestionadas por dicho servidor.

Funcionalidades *multiserver*

1. Iniciar el nodo *multiserver* (requiere dos o más nodos servidor iniciados, **sin** haber hecho `server:start()`):
    - Se escoge uno de esos nodos servidor iniciados (o se crea otro y se inicia), y se ejecuta `server:start_master()`
    - En caso de que exista un fichero `rooms.txt`, la aplicación creará automáticamente las salas leyendo el fichero secuencialmente (cada línea es el nombre de una sala) **sólo** para el proceso categorizado como `master`.
    - Si no existe el fichero, sólo se incluye la sala por defecto, `main`.
2. Arrancar el resto de nodos servidor: `server:start_sserver('<nombre_master>@<IP_master>')` (sí, es `sserver`).
3. Parar el *multiserver*: Para todos los nodos, se hará: `server:stop()`

Limitaciones y errores conocidos

Por falta de tiempo, o debido a que hay aspectos que decidimos que quedaban fuera del alcance de la práctica, hemos rechazado implementar:

- Gestión de una sala por varios nodos servidor de forma paralela.

[2] **nodo**: *Shell* de `Erlang` con un nombre distintivo asociado.

- Internacionalización de la aplicación.
- Mejoras de diseño en la interfaz de usuario.
    - Asignar a cada usuario un color, y que los mensajes de ese usuario se muestren en dicho color.
- Salas privadas (protegidas por contraseña).
- Inicio de sesión de usuarios (*nickname* y contraseña).
- Persistencia de salas y usuarios (almacenaje de información en BD).