Wason (cliente de mensajería instantánea)

A lo largo del semestre irás modificando un pequeño programa cliente que sirve para enviar mensajes a otros usuarios. En cada nueva versión, tu programa usará uno o varios de los algoritmos que veamos a lo largo del curso.

En esta versión no habrá ningún algoritmo criptográfico, únicamente servirá para poder enviar y recibir mensajes entre los usuarios de Wason. Para lograr esto habrá un servidor que va a funcionar como intermediario para conectar a todos los clientes.

Tu cliente se conectará al servidor y mantendrá una conexión abierta para poder enviar y recibir mensajes. En los archivos cliente.py y cliente.html puedes encontrar un pequeño código que envía y recibe mensajes del servidor.

Tu tarea consiste en escoger uno de los programas cliente (dependiendo de si quieres usar Python o JavaScript/HTML) y modificar el código para que tenga una interfaz cómoda. Lo mínimo que se espera de la interfaz es que imprima los mensajes de la siguiente forma:

02:30:31 Alice: hola
02:31:03 Yo: Hola!!!
02:31:59 Alice: qué bonito cliente de mensajería

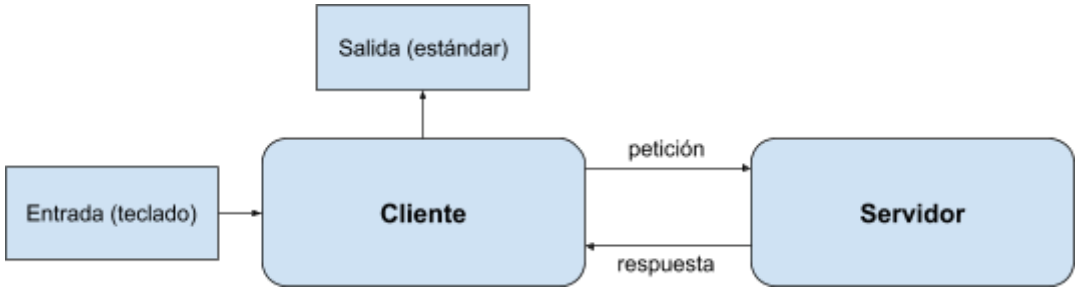
En una mejor interfaz se puede elegir entre varias conversaciones con usuarios distintos. Por ejemplo, usando un menú de opciones o un comando especial se puede cambiar de la conversación con Alice a la conversación con Bob

02:21:31 Bob: y entonces qué???
02:22:03 Yo: no sé
02:22:59 Yo: lo voy a pensar
02:23:10 Yo: 🤔🤔

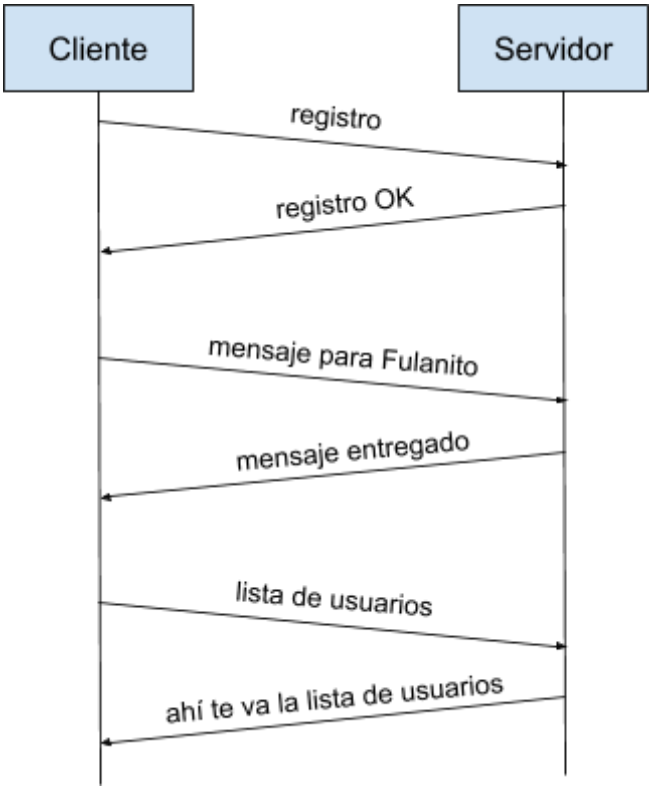
En esta versión deberás hacer los cambios necesarios para llevar a cabo lo siguiente:

- Registro de usuario.
- Mostrar la lista de usuarios conectados.
- Enviar mensajes a cualquier usuario conectado.
- Recibir y mostrar mensajes enviados por otros usuarios.

El siguiente diagrama da una vista general de la conexión.



Y aquí se muestra la comunicación de algunos mensajes entre el cliente y el servidor.



Formato de mensajes

Los mensajes enviados entre cliente y servidor se encuentran en formato JSON. Al final de este documento se muestran los campos que deben incluir los mensajes enviados y recibidos.

Opción Python

Requisitos: Python 3.6 o superior junto con los módulos websockets y aioconsole (estos pueden instalarse con pip).

Si decides usar Python, usarás tanto cliente.py como entrada.py para implementar el cliente. Esta separación entre la entrada y la salida del cliente facilita mostrar en pantalla los mensajes enviados y recibidos.
Para usarlo realiza lo siguiente:

1. Primero se ejecuta cliente.py para que se conecte al servidor.
2. Luego se ejecuta entrada.py para que se conecte al cliente.
3. entrada.py lee texto desde la entrada estándar y cada línea la envía al cliente para que este la muestre en pantalla y la mande al servidor.

No hace falta saber detalles de la comunicación con el servidor. Solo tienes que editar los primeros métodos de cliente.py (los que tienen un comentario), y posiblemente entrada.py para agilizar la interfaz de entrada.
Para construir los mensajes en formato JSON usa objetos de tipo diccionario y las funciones loads y dumps del módulo json.

Opción Javascript

En esta versión tanto la entrada como la salida pueden mostrarse en una misma página (como una aplicación web). Esta opción puede resultar más parecida a las aplicaciones conocidas como Whatsapp, Facebook Messenger o Telegram. Puedes usar esta versión únicamente si sabes algo de HTML y Javascript, o si estás dispuesto a aprenderlo por tu cuenta. Puedes usar las herramientas que consideres necesarias para hacer tu cliente tan robusto, bonito y usable como quieras.

Formato de mensajes

Mensajes enviados por el cliente

Tipo	Campos adicionales	Explicación
registro	nombre : string	Para registrarse en el servidor.
msj_chat	destino : string msj : string	Nombre del destinatario. Texto del mensaje.
usuarios	Ninguno	Para pedir lista de usuarios conectados.

Ejemplos

```
{
  “tipo”:      “registro”,
  “nombre”:    “Fulanito14”
}

{
  “tipo”:      “msj_chat”,
  “destino”:   “otro_usuario”,
  “msj”:       “Hola amigo cómo te va?”
}
```

Mensajes enviados por el servidor

Tipo	Campos adicionales	Explicación
respuesta	ok : boolean	**true**. La petición fue correcta. **false**. Hubo algún error. Ver detalles.
	peticion : string	“**conexion**”. Primer mensaje del servidor; indica si la conexión se hizo correctamente. “**usuarios**”. Para lista de usuarios conectados. “**registro**”. Respuesta de la solicitud de registro. “**msj_enviado**”. Cuando se mandó un mensaje a un usuario. “**otro**”. Para errores no relacionados con lo anterior.
	detalles : string	Información correspondiente a la petición realizada. Puede ser vacía.
msj_nuevo	origen : string	Nombre del usuario que mandó el mensaje.
	hora : string	Hora en que se mandó el mensaje.
	msj : string	Cuerpo del mensaje.

Ejemplos

```
{
  “tipo”:      “respuesta”,
  “ok”:        false,
  “peticion”:  “registro”,
  “detalles”:  “Ya existe un usuario con el nombre ‘omar’”
}
```

```
{
  “tipo”:      “msj_nuevo”,
  “origen”:    “Fulanito14”,
  “hora”:      “19 Feb 2019 22:29:06”,
  “msj”:       “Hola amigo cómo te va?”
}
```