Universidad del Valle de Guatemala
Facultad de Ingeniería
Sistemas Operativos Sección 21
Profesor: Luis Zarceño

“Protocolo WebSockets Proyecto 1”

Clase Sistemas operativos

16 de marzo del 2025, Guatemala de la Asunción

Protocolo de Comunicación WebSocket para Chat en Tiempo Real

1. Objetivo

Establecer una comunicación bidireccional en tiempo real entre clientes y servidor, permitiendo:

- Registro y autenticación de usuarios.
- Envío de mensajes (broadcast y privados).
- Gestión de estados (ACTIVO, OCUPADO, INACTIVO).
- Consulta de listado e información de usuarios.
- Cierre controlado de conexiones.

2. Establecimiento de la Conexión

2.1 Handshake Inicial

Cliente: Envía una solicitud HTTP para actualizar a WebSocket:

GET /chat HTTP/1.1
Host:
chatservidor.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key:
<clave_unica>
Sec-WebSocket-Version: 13

- **Servidor:** Responde con un código 101 Switching Protocols para confirmar el cambio a WebSocket.

3. Formato de Mensajes

- **Formato:** JSON (fácilmente parseable y legible).

Estructura general de cada mensaje:

```
{
 "type": "<tipo_mensaje>",
 "sender": "<nombre_usuario>",
 "target":
 "<destinatario_opcional>",
 "content":
 "<contenido_del_mensaje>",
 "timestamp": "<fecha_y_hora>"
}
```

4. Definición de Mensajes y Flujo de Comunicación

4.1 Registro de Usuario

Mensaje de registro:

```
{
 "type": "register",
 "sender":
 "<nombre_usuario>",
 "content": null
}
```

Respuesta del servidor:

```
{
 "type":
 "register_success",
 "sender": "server",
 "content": "Registro exitoso",
 "userList": ["usuario1", "usuario2", ...],
 "timestamp": "<fecha_y_hora>"
}
```

4.2 Mensajería

4.2.1 Mensajes de Chat General (Broadcast)

Cliente:

```
{
 "type": "broadcast",
 "sender": "<nombre_usuario>",
 "content": "<mensaje>",
 "timestamp":
 "<fecha_y_hora>"
}
```

4.2.2 Mensajes de Chat Privado Cliente:

```
{
 "type": "private",
 "sender": "<nombre_usuario>",
 "target":
 "<nombre_destinatario>",
 "content": "<mensaje>",
 "timestamp": "<fecha_y_hora>"
}
```

4.3 Listado de Usuarios

Cliente:

```
{
 "type": "list_users",
 "sender": "<nombre_usuario>"
}
```

Servidor:

```
{
 "type":
 "list_users_response",
 "sender": "server",
 "content": ["usuario1", "usuario2", "usuario3"],
 "timestamp": "<fecha_y_hora>"
}
```

4.4 Información de Usuario Específico

Cliente:

```
{
 "type": "user_info",
 "sender":
 "<nombre_usuario>", "target":
 "<usuario_objetivo>"
}
```

Servidor:

```
{
 "type":
 "user_info_response",
 "sender": "server",
 "target": "<usuario_objetivo>",
 "content": {"ip": "<ip_del_usuario>", "status": "<estado_del_usuario>"},
 "timestamp": "<fecha_y_hora>"
}
```

4.5 Cambio de Estado (Status)

Cliente:

```
{
 "type": "change_status",
 "sender":
 "<nombre_usuario>",
 "content": "<nuevo_estado>"
}
```

Servidor:

```
{
 "type":
 "status_update",
 "sender": "server",
 "content": {"user": "<nombre_usuario>", "status": "<nuevo_estado>"},
 "timestamp": "<fecha_y_hora>"
}
```

4.6 Cierre de Conexión

Cliente:

```
{
 "type": "disconnect",
 "sender":
 "<nombre_usuario>",
 "content": "Cierre de sesión"
}
```

Servidor:

```
{
 "type":
 "user_disconnected",
 "sender": "server",
 "content": "<nombre_usuario> ha salido",
 "timestamp": "<fecha_y_hora>"
}
```

5. Manejo de Errores

Mensaje de error estándar:

```
{
 "type": "error",
 "sender": "server",
 "content": "Descripción del error",
 "timestamp": "<fecha_y_hora>"
}
```

6. Ejemplo de Implementación en C

Cliente WebSocket en C

```
#include <stdio.h>
#include <libwebsockets.h>

static int callback_chat(struct lws *wsi, enum lws_callback_reasons reason, void *user, void *in,
size_t len) {
   switch (reason) {
      case LWS_CALLBACK_CLIENT_ESTABLISHED:
         printf("Conexión establecida con el servidor WebSocket\n");
         break;
```

```
        case LWS_CALLBACK_CLIENT_RECEIVE:
            printf("Mensaje recibido: %s\n", (char *)in);
            break;
        case LWS_CALLBACK_CLIENT_WRITEABLE:
            lws_write(wsi, (unsigned char *)"{\"type\": \"register\", \"sender\": \"usuario1\"}", 50,
LWS_WRITE_TEXT);
            break;
        case LWS_CALLBACK_CLOSED:
            printf("Conexión cerrada\n");
            break;
        default:
            break;
    }
    return 0;
}
```

Este código es una base para implementar el cliente WebSocket en C usando libwebsockets.