



Ficha de Aprendizaje: Práctica de Diseño Básico de una Arquitectura Cliente-Servidor Utilizando Sockets en PHP con CodeIgniter

Objetivo General:

Desarrollar una aplicación cliente-servidor básica utilizando sockets en PHP bajo el framework CodeIgniter, comprendiendo el flujo de comunicación entre ambas partes.

Objetivos Específicos:

1. Configurar un entorno de desarrollo PHP con CodeIgniter.
 2. Implementar un servidor basado en sockets que escuche y responda solicitudes de los clientes.
 3. Desarrollar un cliente PHP que se conecte al servidor mediante sockets.
 4. Integrar la arquitectura de cliente-servidor con el framework CodeIgniter.
 5. Probar la comunicación efectiva entre cliente y servidor.
-

Requisitos Previos:

- Conocimientos básicos de PHP.
 - Familiaridad con el framework CodeIgniter.
 - Conceptos fundamentales sobre arquitectura cliente-servidor.
 - Comprensión básica de la programación de sockets en PHP.
-

Materiales:

- **Software necesario:**
 - XAMPP o WAMP para entorno local de desarrollo.
 - Framework CodeIgniter (versión 3 o superior).
 - **Herramientas:**
 - Editor de código (VS Code, Sublime, etc.).
 - Navegador web para probar la aplicación.
 - Terminal o consola para ejecución del servidor PHP.
-
-

Actividades y Pasos:

1. Configuración del Entorno de Desarrollo

- Instalar y configurar XAMPP o WAMP.

- Crear un nuevo proyecto CodeIgniter y configurar el archivo `config.php` para que apunte a la base URL correcta.

2. Creación del Servidor con Sockets

- Crear un archivo PHP para implementar el servidor que escuche en un puerto determinado.
- Usar la función `socket_create()` para inicializar el socket y `socket_bind()` para asociarlo a una IP y puerto.
- Implementar lógica para recibir conexiones y responder a las solicitudes de los clientes.

Ejemplo de código básico para el servidor:

Pasos a seguir

1. Configuración del servidor WebSocket

En este ejemplo, vamos a usar **Ratchet** para manejar los WebSockets en PHP. Sigue estos pasos:

[Instala Composer](#)

Si no tienes Composer instalado, descárgalo desde [Composer](#).

[Instala Ratchet](#)

En tu terminal, navega al directorio de tu aplicación CodeIgniter y ejecuta:

```
composer require cboden/ratchet
```

2. Servidor de WebSocket en PHP

Crea un archivo llamado `ChatServer.php` en el directorio `application/controllers/` con el siguiente contenido:

```
<?php
use Ratchet\MessageComponentInterface;
use Ratchet\ConnectionInterface;

class ChatServer implements MessageComponentInterface {

    protected $clients;

    public function __construct() {
        $this->clients = new \SplObjectStorage;
    }

    public function onOpen(ConnectionInterface $conn) {
```

```
// Almacena la nueva conexión
$this->clients->attach($conn);
echo "Nueva conexión ({ $conn->resourceId})\n";
}

public function onMessage(ConnectionInterface $from, $msg) {
    $numRecv = count($this->clients) - 1;
    echo sprintf('Conexión %d enviando mensaje "%s" a %d otra(s)
conexiones' . "\n",
        $from->resourceId, $msg, $numRecv);

    foreach ($this->clients as $client) {
        if ($from !== $client) {
            // Enviar el mensaje a todos los clientes excepto al
remitente
            $client->send($msg);
        }
    }
}

public function onClose(ConnectionInterface $conn) {
    // Elimina la conexión
    $this->clients->detach($conn);
    echo "Conexión { $conn->resourceId} cerrada\n";
}

public function onError(ConnectionInterface $conn, \Exception $e) {
    echo "Error: { $e->getMessage()}\n";
    $conn->close();
}
}
```

3. Iniciar el servidor de WebSocket

Crea un archivo llamado `runChatServer.php` en la raíz de tu proyecto:

```
<?php
require 'vendor/autoload.php';
use Ratchet\Http\HttpServer;
use Ratchet\WebSocket\WsServer;
use Ratchet\Server\IoServer;
//use application\controllers\ChatServer;
//use ChatServer;

require_once('application/controllers/ChatServer.php');

$server = IoServer::factory(
    new HttpServer(
```

```
        new WsServer(  
            new ChatServer()  
        )  
    ),  
    8080  
);  
  
echo "Servidor WebSocket corriendo en el puerto 8080...\n";  
$server->run();
```

Para iniciar el servidor, ejecuta este script en la terminal:

```
php runChatServer.php
```

4. Frontend: JavaScript y HTML para el chat

Agrega un controlador para la vista del chat en `application/controllers/Chat.php`:

```
<?php  
class Chat extends CI_Controller {  
    public function index() {  
        $this->load->view('chat_view');  
    }  
}
```

Luego, crea una vista en `application/views/chat_view.php`:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Chat en Tiempo Real</title>  
    <style>  
        #chat {  
            width: 100%;  
            height: 400px;  
            overflow-y: scroll;  
            border: 1px solid #ccc;  
        }  
        #message {  
            width: 80%;  
        }  
    </style>  
</head>  
<body>  
    <h1>Chat en Tiempo Real</h1>
```

```
<div id="chat"></div>
<input type="text" id="message" placeholder="Escribe tu mensaje">
<button id="send">Enviar</button>

<script>
  var conn = new WebSocket('ws://localhost:8080');
  var chat = document.getElementById('chat');
  var sendButton = document.getElementById('send');
  var messageInput = document.getElementById('message');

  conn.onopen = function(e) {
    chat.innerHTML += '<div>Conexión establecida</div>';
  };

  conn.onmessage = function(e) {
    chat.innerHTML += '<div>' + e.data + '</div>';
    chat.scrollTop = chat.scrollHeight;
  };

  sendButton.onclick = function() {
    var msg = messageInput.value;
    conn.send(msg);
    messageInput.value = '';
  };
</script>
</body>
</html>
```

5. Enlace de rutas

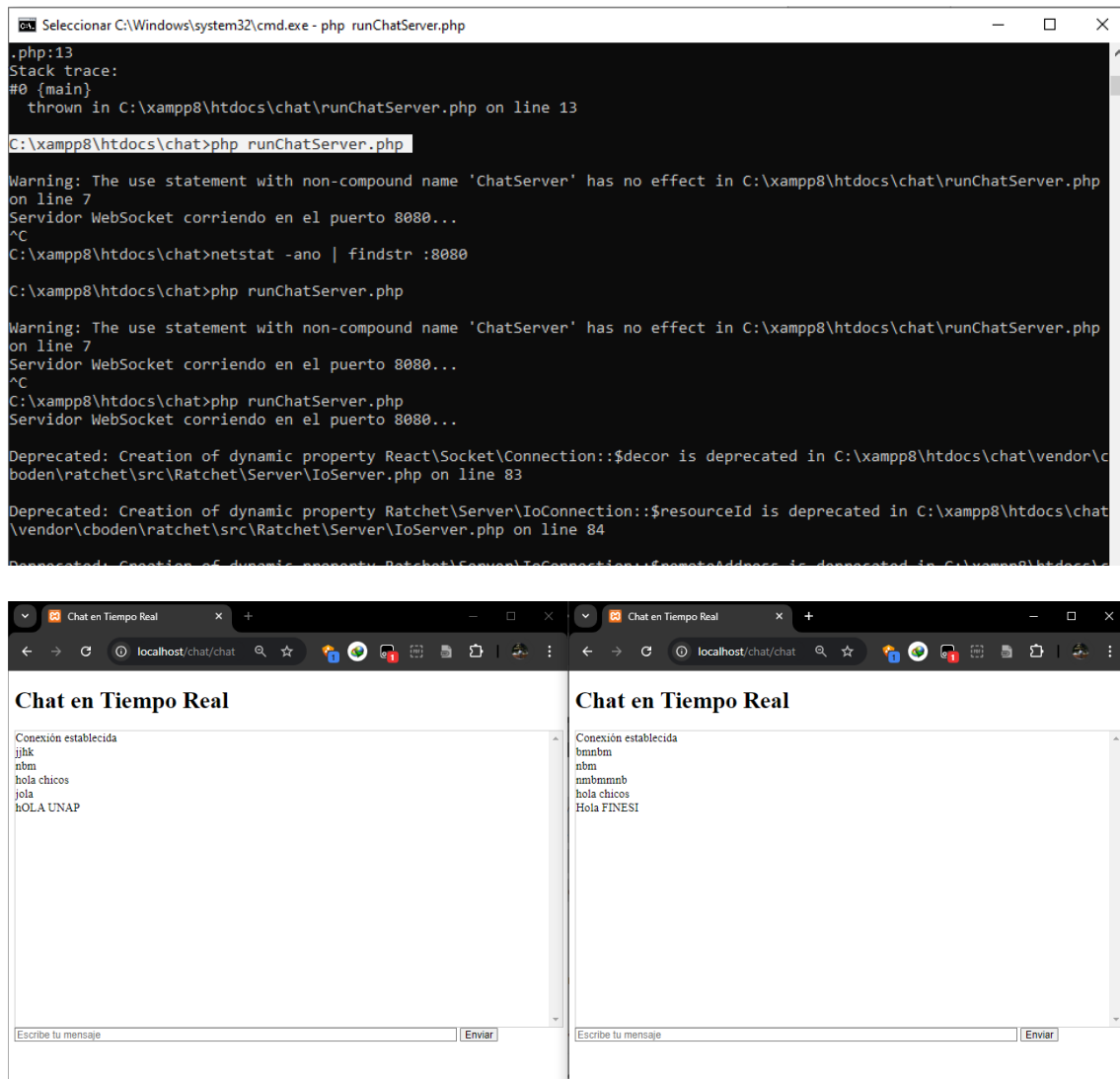
Abre el archivo `application/config/routes.php` y añade la ruta para el chat:

```
$route['chat'] = 'chat/index';
```

6. Ejecutar la aplicación

1. Asegúrate de que tu servidor de WebSocket esté en ejecución (puedes ver los logs en la terminal si todo va bien).
2. Inicia tu servidor web de CodeIgniter y navega a `http://localhost/chat/chat` en tu navegador.
3. Deberías poder enviar mensajes y ver cómo se propagan en tiempo real entre las conexiones abiertas.

EJECUCION DE CHATS



The image shows a terminal window and two browser windows. The terminal window displays the execution of a PHP script named `runChatServer.php`. It shows a stack trace, a warning about a non-compound name, and the server starting on port 8080. The browser windows show the chat interface at `localhost/chat/chat`. The left window shows a list of messages: "Conexión establecida", "jlk", "nbn", "hola chicos", "jola", and "hOLA UNAP". The right window shows a list of messages: "Conexión establecida", "bmnbm", "nbn", "nmbmmnb", "hola chicos", "Hola FINESI", and "HOLA FINESI".

Pruebas y Verificación

- Iniciar el servidor de sockets y ejecutar la aplicación CodeIgniter.
- Acceder al cliente desde el navegador y verificar que la comunicación con el servidor se realiza correctamente.
- Validar la respuesta del servidor en la interfaz gráfica del cliente.

Evaluación:

1. ¿La aplicación cliente puede conectarse y recibir una respuesta del servidor?
2. ¿El servidor maneja múltiples solicitudes de clientes sin fallar?
3. ¿El proyecto se integra correctamente con CodeIgniter para facilitar la interacción con la arquitectura cliente-servidor?



Extensión de la Práctica (Opcional):

- Implementar manejo de errores en la conexión y comunicación de sockets.
 - Crear un sistema de autenticación simple entre el cliente y el servidor utilizando sockets.
 - Probar la aplicación en un entorno remoto o en una red local.
-

Referencias:

- Documentación Oficial de CodeIgniter
- [Sockets en PHP - Manual](#)