PLATAFORMA DE COMUNICACIÓN EN TIEMPO REAL PARA PLATAFORMAS MÓVILES

Your Name*, Second Name†, Third Name‡, Fourth Name‡ and Fifth Name§
*School of Electrical and
Computer Engineering
Institute of Technology
99999 Testcity
Email: test@test.tes
†Ecole Superieure
Nantes, France
Email: second@second.fr
‡Star Academy
San Francisco, California 99999-9999
Telephone: (800) 555–5555
Fax: (888) 555–5555
§Rückwärts GmbH
Niemandsweg 73
99999 Musterstadt, Germany

***Abstract*—En este trabajo se describe como desarrollar un servidor que brinde un canal de comunicación en tiempo real, que será utilzado para controlar plataformas móviles, además de crear un cliente web que servirá como visualizador de datos que esten interactuando entre el cliente y servidor. Se menciona el protocolo de comunicación a utilizar, la forma en el que se va a desarrollar, el esquema del cliente, esquema del servidor y resultados obtenidos.**

***Index Terms*—Protocolos, websocket, Json, cliente, servidor**

I. INTRODUCCIÓN

La comunicación en tiempo real entre varios dispositivos móviles es una necesidad que va creciendo constantemente al realizar tele operación entre plataformas móviles, debido a que en la comunicación se va a minimizar la pérdida de datos, fallos de conexión, tiempo de respuesta entre cliente y servidor, comunicación full-dúplex.[1]

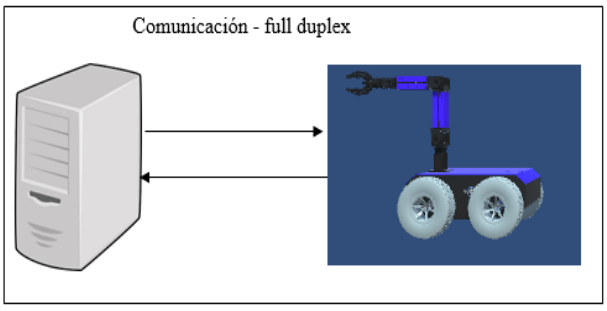

Fig. 1. Comunicación Plataforma

El autor Alan FT Winfield realiza la comunicación entre robots utilizando el protocolo de comunicación TCP-IP[2], que no es un protocolo muy seguro ya que podria haber pérdida de datos y las plataformas móviles no trabajarian de forma correcta. Mientras que los autores H. G. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, utilizan radiofrecuencia para la comunicación[3], teniendo como inconveniente las interferencias que se producen debido a que hay varias señales dentro de la misma frecuencia.

Se propone utilizar en la comunicación en tiempo real el protocolo de comunicación denominado websocket, debido a que es una tecnología que proporciona un canal de comunicación bidireccional y fullduplex. Además utilizar el formato Json para el empaquetamiento de los datos de los datos, de esta manera tener el control absoluto de los datos que llegan o se envian en el servidor.

Como resultado de esta investigación se obtendrá un servidor web que brindará un canal de comunicación estable que permitirá la comunicación fullduplex entre los clientes y el servidor, además de un cliente web que servira para mostrar los datos que se esten enviando en tiempo real. Logrando con esto la comunicación bidireccional de varios dispositivos moviles que esten conectados, enviando y recibiendo datos del servidor.

Es presente trabajo esta estructurado de la siguiente manera: en la sección 1 Introducción, la sección 2 se realiza la selección del protolo de comunicación a utilizar, en la sección 3 se detalla un análisis previo al desarrollo del cliente y servidor, en la sección 4 se muestra como se desarrollo el servidor que utiliza websocket para el canal de comunicación,

en la secion 5 el desarrollo del cliente websocket, en la sección 6 los resultados experimentales y finalmente en la sección 7 las conlcusiones.

II. PROTOCOLOS DE COMUNICACIÓN

En el mundo de la informática se conoce bajo el nombre de protocolo, al lenguaje que es un conjunto de reglas formales, que permiten la comunicación de distintas computadoras entre sí. Dentro de las distintas redes, como Internet, existen numerosos tipos de protocolos. Aquí forman parte los distintos protocolos y modelos de redes que proveen servicios de comunicación en sistemas distribuidos[4].

1) Comunicación Cliente - Servidor (Polling): El servidor generalmente recibe las peticiones de los clientes, las procesa y envía posteriormente los resultados a través de la conexión que existe entre ellos. La comunicación entre el cliente y el servidor es de tipo Polling (preguntar y recibir) y genera un esquema Half Duplex, similar a un sistema PTT (Push To Talk) donde ambas partes pueden enviar y recibir mensajes pero solo una a la vez, la figura 2 muesra el esquema de funcionamiento de Polling[5].

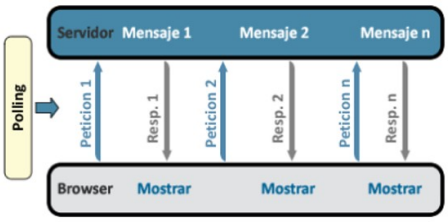

Fig. 2. Esquema Polling

Este gráfico permite comprender la mecánica de asincronismo de estas tecnologías. En primer lugar el cliente (browser) envía una petición, el servidor la recibe, procesa y envía la respuesta que el navegador muestra en pantalla. Este esquema de trabajo hace que el cliente siempre tenga que "pedir" para recibir la información. Si el lado cliente no solicita información entonces esta no será enviada y esto acarrea algunos problemas

A. Rest

Es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web. Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aún asi Rest está basado en estándares[6]:

- HTTP
- URL
- Representación de los recursos XML, HTML

Rest dicta que los mensajes HTTP deberían ser tan descriptivos como sea posible. Esto hace posible que los intermediarios interpreten los mensajes y ejecuten servicios en nombre del usuario. Uno de los modos que HTTP logra esto es por medio del uso de varios métodos estándares, muchos encabezamientos y un mecanismo de direccionamiento. Por ejemplo, las cachés Web saben que por defecto el comando GET es cacheable (ya que es side-effect-free) en cambio POST no lo es. Además saben como consultar las cabeceras para controlar la caducidad de la información. HTTP es un protocolo sin estado y cuando se utiliza adecuadamente, es posible interpretar cada mensaje sin ningún conocimiento de los mensajes precedentes. Por ejemplo, en vez de logearse del modo que lo hace el protocolo FTP, HTTP envía esta información en cada mensaje[6].

Los clientes envían los mensajes basados en los estándares anteriormente mencionados hacia el servidor. El servidor los recibe y los puede manipular reenviándolos a otro cliente o almacenándolos en una base de datos, la figura 3 muestra un esquema del funcionamiento de Rest al momento de envió y recepción de mensajes[6].

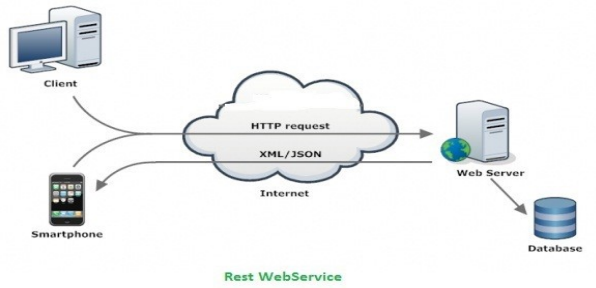

Fig. 3. Esquema Rest

B. Websocket

Es la tecnología que llega para resolver los problemas de comunicación que plantean los esquemas anteriormente descritos. De manera simple, Websockets permite comunicar el cliente y el servidor a través de un canal Full Duplex bidireccional y sin tener que hacer polling por parte del cliente ni acoplarse a estándares predefinidos. La figura 4 muestra el funcionamiento de los websockets[7].

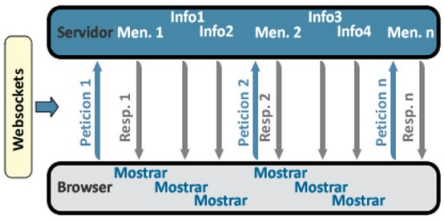

Fig. 4. Esquema Websocket

C. Selección del protocolo

En la tele operación de plataformas móviles se necesita una comunicación fullduplex, estable y con mínimos retardos de tiempo, para eso se produce a comparar los protocolos anteriormente mencionados.

- Websocket - Polling

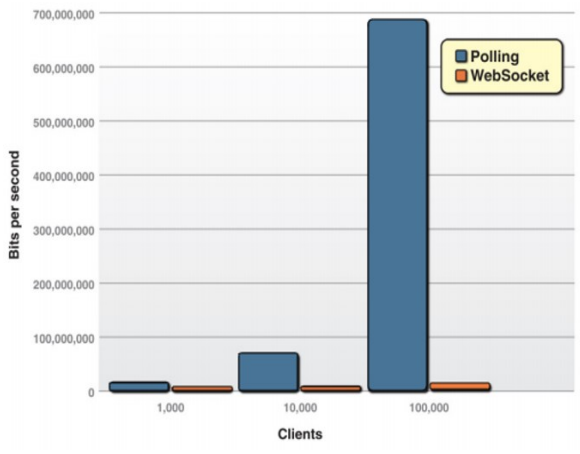

Fig. 5. Websocket - Polling

En la figura 5 puede verse la reducción bits para encapsular los datos y enviarlos a la red, gracias a que no hay que implementar polling enviando paquetes de datos que solo "preguntan" al servidor si hay información para que este envíe al cliente. Eso sin contar los paquetes enviados por el cliente que terminan siendo inútiles porque el servidor no tiene información para enviar.

- Websocket - Rest

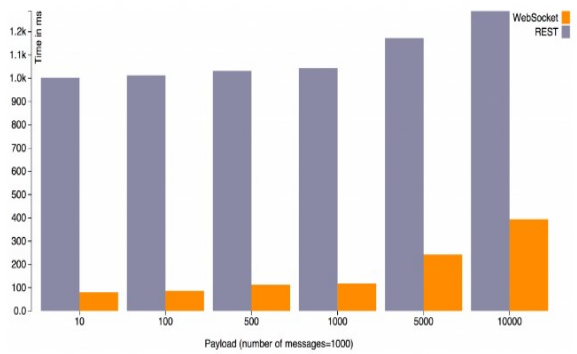

Fig. 6. Websocket - Rest

La figura 6 muestra la reducción de tiempo de respuesta al utilizar websockets en la transmisión de mensajes. Demostrando que es el protocolo más adecuado para cumplir el objetivo principal de esta investigación que es lograr la comunicación bidireccional de plataformas móviles.

III. ANÁLISIS EN PARA EL DESARROLLO DE LA PLATAFORMA DE COMUNICACIÓN

Antes de realizer cualquier Proyecto de software es necesario realizer ciertas tareas previas para el correcto desarrollo.

A. Definir forma de programación

Existe varias formas o métodos para poder programar o desarrollar algún proyecto de software, esto va a depender del funcionamiento y objetivos que se vayan a cumplir en el proyecto. De las más conocidas son Modelo Vista Controlador (MVC), Por Capas, Modelo Vista Presentador (MVP).

- El Modelo Vista controlador (MVC), es un patrón de arquitectura de software que separa los datos y la lógica del negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario[8].
- Programación por Capas: La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario[9].
- Modelo Vista Presentador: El Patrón Modelo-Vista-Presentador (MVP) surge para ayudar a realizar pruebas automáticas de la interfaz gráfica, para ello la idea es codificar la interfaz de usuario lo más simple posible, teniendo el menor código posible, de forma que no merezca la pena probarla. En su lugar, toda la lógica de la interfaz de usuario, se hace en una clase separada (que se conoce como Presentador), que no dependa en absoluto de los componentes de la interfaz gráfica y que, por tanto, es más fácil de realizar pruebas[10].

El modelo que se va a utilizar es MVC, debido a que presta los servicios necesarios para poder cumplir los objetivos trazados en el proyecto.

B. Definir forma de programación

El momento de desarrollar algún proyecto de software es necesario definir que tecnologías se va a utilizar, las mismas se detallan a continuación:

- Lenguajes de Programación: Al tratarse de aplicaciones y servicios web tanto para el servidor como para los clientes, se decidió utilizar PHP para el lado del servidor, y javascript, html, para el lado del cliente[11].
- Tecnologías: Al tratarse de una aplicación web MVC, lo más recomendable es utilizar el framework Codeigniter, que es un framework PHP para la creación rápida de aplicaciones web[12].

IV. DESARROLLO DEL SERVIDOR

El servidor web está desarrollado en php, utilizando la librería web PHPwebsocket.php, que es una librería que contiene los métodos, funciones y variables que utiliza el protocolo websocket para poder crear el canal de comunicación[13]. La figura 7, muestra cómo se realiza el proceso de funcionamiento del servidor.

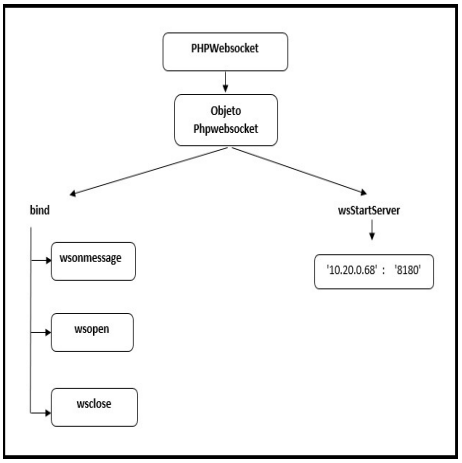

Fig. 7. Esquema del Servidor

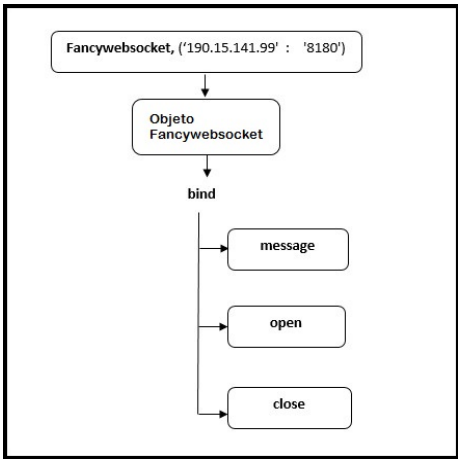

Fig. 8. Esquema del Cliente

El funcionamiento de forma detallada es el siguiente:

- Crear una instancia de la librería PHPWebSocket, para poder utilizar los métodos que contiene esta librería.
  - $this->socket= new PHPWebSocket();
- Enlaza la función wsOnMessage con la librería PHP-Websocket mediante "message", permitiendo utilizar las variables y funciones referentes a los mensajes que se están enviando.
  - $this->socket->bind('message', 'wsOnMessage');
- Enlaza wsOnOpen con la librería PHPWebsocker mediante "open". Permitiendo utilizar las funciones y variables que permiten abrir la comunicación.
  - $this->socket->bind('open', 'wsOnOpen');
- Enlaza wsOnClose con la librería PHPWebsocker mediante "close". Permitiendo utilizar las funciones y variables que permiten cerrar la comunicación.
  - $this->socket->bind('close', 'wsOnClose');
- Inicializa el socket en la ip y puerto especificados.
  - return $this->socket->wsStartServer ('192.168.1.102',9300);

V. DESARROLLO DEL CLIENTE

Se creó una página web que mostrará los datos enviados por los clientes, esta página web también es un cliente más con la diferencia que es un cliente de visualización de datos, se utilizó html y css para la creación de la misma. La aplicación cliente, es decir el websocket cliente está escrito en javascript y jquery, en esta parte también se utiliza una librería que contiene los métodos para conectarse a un websocket server[14]. Esta librería se llama fancywebsocket, de la misma manera tiene los métodos open, message, close, el esquema se muestra en la figura 8.

El funcionamiento es el siguiente:

- Crear una instancia de la librería fancywebsocket, especificando la dirección ip y el puerto al que se debe conectar.
  - Server = new FancyWebSocket('ws://192.168.1.102:9300');
- Abrir la conexión para poder conectarse al servidor.
  - Server.bind('open', function(){..... });
- Envia el mensaje hacia el servidor enpaquetado en formato json.
  - Server.send('message', JSON.stringify(mensaje));
- Cierra la conexión con el servidor.
  - Server.bind('close', function(data) -
- Obtiene mensajes de los clientes para poder mostrarlos en pantalla.
  - Server.bind('message', function(payload); var res = jQuery.parseJSON(payload); console.log(res);

VI. RESULTADOS

A. Comparación

El primer resultado que se obtuvo fue la demostración que el protocolo utilizado es la mejor opción para realizar teleoperación.

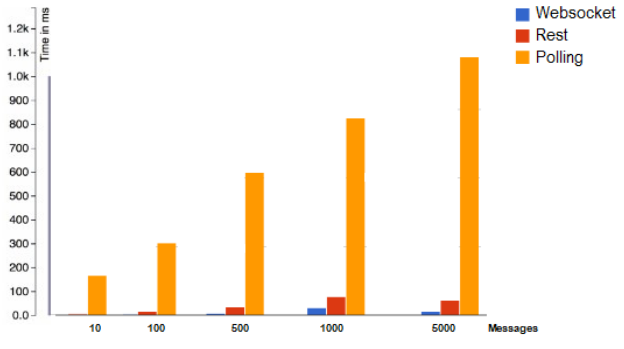

Fig. 9. Websocket vs Rest vs Polling

En el gráfico anterior se puede observar la cantidad de datos enviados por tiempo en milisegundos, demostrando que al utilizar websocket se obtiene un mayor numero de mensajes enviados.

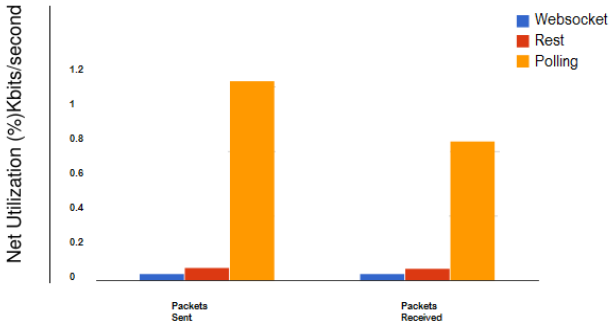

Fig. 10. Paquetes Transmitidos - Recibidos

La figura 10 muestra el porcentaje de consumo de red entre paquetes enviados y recibidos, demostrando que websocket tiene un consumo mínimo de red.

B. Servidor

Se tiene un servidor que cuando inicia el websocket esta en espera de los clientes que se van a conectar y brinda la comunicación full-duplex, La figura 11 describe el servidor. Se inicia atraves de línea de comandos en terminal linux en nuestro caso, para esto se hace lo siguiente:

- Ubicarse en la dirección donde estén los archivos y escribir : /var/www/html/Teleoperacion-Bilateral/.
- Escribir el comando para iniciar el servidor websocker, En la siguiente figura se detalla los servicios que se inician. php index.php Administrador/iniciar_socket.

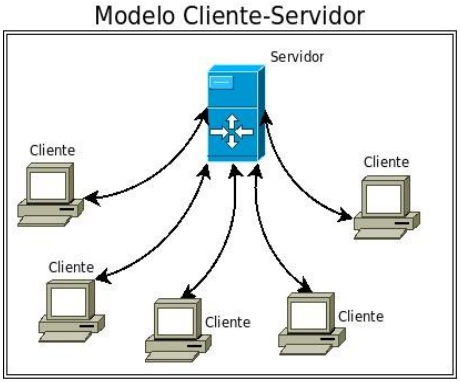

Fig. 11. Comunicación Clientes - Servidor

C. Cliente

En cuanto al cliente web, Conectar el cliente abriendo un navegador web especificando la dirección del servidor, en este caso se trabaja en un servidor local, se muestra en la figura 12, http://127.0.0.1Teleoperacion-Bilateral/Administrador/g

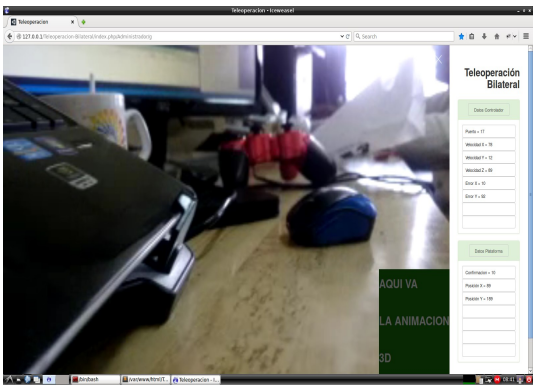

Fig. 12. Cliente Web

D. Comunicación Bidireccional

- La primera conexión permite identificar al servidor que cliente se conecta, como se puede ver en la figura 13.
- El cliente envia el mensaje de la siguiente forma: {"cadena":"c"}
- Desde el cliente se envian los datos en formato JSON {"cadena":"cadena","cadena":"cadena"}

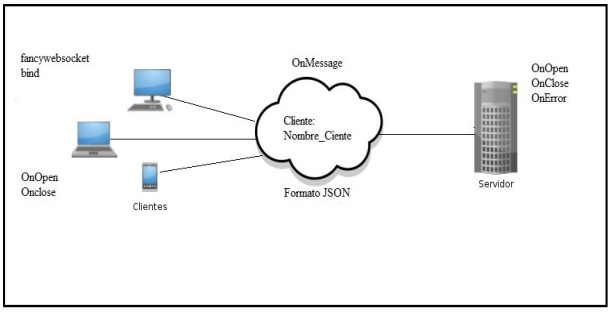

Fig. 13. Identificación de clientes

- Los mensajes llegan al servidor identificando {"cadena":"cadena","cadena":"cadena"}
- El re-direccionamiento de los mensajes va a variar dependiendo de los datos que llegue en la cadena JSON en el campo origen y destino, como se observa en la figura 14.

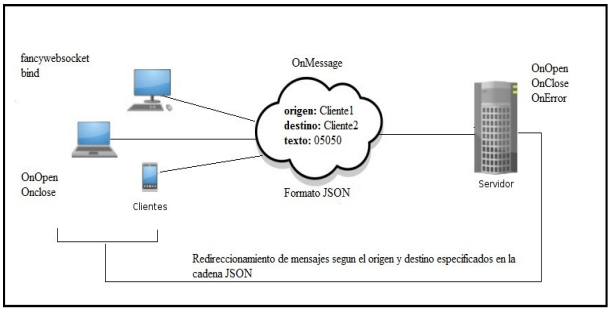

Fig. 14. Redireccionamiento

VII. CONCLUSIONES

- Se utilizó el protocolo de comunicación webocket, debido a que brinda una comunicación estable, full duplex y con un tiempo de respuesta mínimo.
- El framework codeigniter ofrece un modelo vista controlador, de modo que facilitó la programación de las aplicaciones con las que se está trabajando.
- Es necesario empaquetar los datos en formato Json para poder utilizar cada campo de la cadena que se crea, permitiendo tener el control de los datos que entran y salen del servidor.

AGRADECIMIENTOS

Al Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado -CEDIA-, Universidad Técnica de Ambato UTA y a la Universidad de las Fuerzas Armadas ESPE por el financiamiento del proyecto Tele-operación Bilateral Cooperativo de Múltiples Manipuladores Móviles CEPRAIX-2015-05 y el apoyo para el desarrollo del presente trabajo.

REFERENCES

[1] N. H. F. Beebe. (2010, Dec.) TEX user group bibliography archive. [Online]. Available: http://www.math.utah.edu/pub/tex/bib/index-table.html

[2] A. F. Winfield, *Distributed Sensing and Data Collection Via Broken Ad Hoc Wireless Connected Networks of Mobile Robots*. Tokyo: Springer Japan, 2000, pp. 273–282. [Online]. Available: http://dx.doi.org/10.1007/978-4-431-67919-6_26

[3] M. R. A. G. J. M. S. H. G. Nguyen, N. Pezeshkian, "Autonomous communication relays for tactical robots," *The 11th International Conference on Advanced Robotics*, 2003.

[4] A. Álvarez, A. Pitarch, and J. Monferrer, "Protocolos de comunicación en web: el portfolio europeo de las lenguas electrónico y los weblogs," in *III Congreso sobre Lengua y Sociedad, Universitat Jaume I de Castelló*, 2006.

[5] Iskandar, T. Hendrawan, N. Rachmana, and G. F. Ramadhan, "Polling system as a mobile tv interactive application based on dvb and unicast hybrid network," in *Telecommunication Systems Services and Applications (TSSA), 2014 8th International Conference on*, Oct 2014, pp. 1–5.

[6] L. Li, W. Chou, W. Zhou, and M. Luo, "Design patterns and extensibility of rest api for networking applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 154–167, March 2016.

[7] T. Anusas-amornkul and C. Silawong, "The study of compression algorithms for websocket protocol," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014 11th International Conference on*, May 2014, pp. 1–6.

[8] Y. D. González and Y. F. Romero, "Patrón modelo-vista-controlador." *Revista Telem@ tica*, vol. 11, no. 1, pp. 47–57, 2012.

[9] S. D. M. Henríquez, H. V. Huerta, and L. A. G. Grados, "Programación en n capas," *Revista de investigación de Sistemas e Informática*, vol. 7, no. 2, pp. 57–67, 2010.

[10] D. S. Diéguez, "Patrón modelo-vista-presentador (mvp)," *MoleQla: revista de Ciencias de la Universidad Pablo de Olavide*, no. 20, p. 7, 2015.

[11] R. Nixon, *Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites*. " O'Reilly Media, Inc.", 2012.

[12] D. Upton, *CodeIgniter for Rapid PHP Application Development*. Packt Publishing Ltd, 2007.

[13] M. Casario, P. Elst, C. Brown, N. Wormser, and C. Hanquez, "Html5 websocket," in *HTML5 Solutions: Essential Techniques for HTML5 Developers*. Springer, 2011, pp. 241–261.

[14] M. Banchoff Tzancoff, "Websocket: comparación de performance e implementación de aplicaciones web," Ph.D. dissertation, Facultad de Informática, 2011.