UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

ARQUITECTURA PERVASIVA CON TECNOLOGÍAS WebRTC HÍBRIDAS PARA EL DESARROLLO DE UN FRAMEWORK MODELO VISTA CONTROLADOR DE TIEMPO REAL

PRESENTADA POR:

RAMIRO PEDRO LAURA MURILLO

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2019

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

ARQUITECTURA PERVASIVA CON TECNOLOGÍAS WebRTC HÍBRIDAS PARA EL DESARROLLO DE UN FRAMEWORK MODELO VISTA CONTROLADOR DE TIEMPO REAL

PRESENTADA POR:

RAMIRO PEDRO LAURA MURILLO

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE .....
Dr. JUAN REYNALDO PAREDES QUISPE

PRIMER MIEMBRO .....
Dr. JUAN WALTER TUDELA MAMANI

SEGUNDO MIEMBRO .....
Dr. LEONEL COYLA IDME

ASESOR DE TESIS .....
Dr. BERNABÉ CANQUI FLORES

Puno, 05 de setiembre de 2019

AREA : Ciencias de la Computación
TEMA : Teoría de Sistemas y administración de sistemas
LINEA : Desarrollo de un Framework MVC de tiempo real

DEDICATORIA

A mis padres **Pedro y Adela**, que han dedicado su vida en cuidar de mis hermanas y de mi con sus peculiares ejemplos, cariño y formas no ortodoxas de educación, pero en final efectivas para criar unos hijos de bien y este grado esta dedicados a ellos como un agradecimiento por su sacrificio y enseñanzas.

A Julia, mi novia y esposita, y aunque algunas veces mi dolorcito de cabeza yo la quiero mucho, fue con ella con quien hemos comenzado a crear y compartir una vida de pareja unida y perdurable mirando siempre hacia el futuro, ella me ha enseñado que el cariño puede fluir con total compromiso y entrega aunque debemos lidiar siempre con los problemas cotidianos, pero estoy para apoyarla y quererla mucho estoy para quererla mucho y no para tratar de comprenderla, pero en suma y sobretodo me he encontrado enamorado de su forma de quererme y ponerme siempre en primer lugar.

A mis "queridas hermanas" Docio y Mary por su apoyo cuando he estado en problemas, eso tiene un valor incalculable gracias mis niñas, sin el sentido de responsabilidad de Mary unida con el desenfado de Docio no hubiéramos podido apoyarnos y avanzar juntos.

A mis amigotes de toda la vida a Romel y Vlady, sin ustedes no estaría concretando esta meta, mis más sinceros agradecimientos pues ustedes en particular nunca siempre me han dado la mano y confiado en mis a veces cabildantes habilidades.

A mis grandes amigas que llevaré con cariño en el día a día a las Monchis, Yojis, Lin Vic, Dorca, Sandrita, Paola, la coleguita Edith que logramos una sintonía de sentidos y metas personales con el camino de la vida siempre es difícil pero los amigos puros, siempre se apoyan, gracias.

AGRADECIMIENTOS

- A la Universidad Nacional del Altiplano por darnos la oportunidad de formarnos bajo el entorno del conocimiento pleno de la sociedad académica en nuestras respectivas áreas.

- A mis jurados Dr. Reynaldo Paredes, Dr. Walter Tudela, Dr. Leonel Coyla y especialmente a mi asesor Dr. Bernabe Canqui por apoyarme durante la elaboración de esta tesis, así como por los alcances y correcciones.

- Al Dr. Wenceslao Medina Espinoza y Dr. Walter Tudela Mamani por la confianza y la comprensión del caso en algunas ausencias del trabajo justamente para los trámites para la así consolidación de este grado.

ÍNDICE GENERAL

Pág.

CAPÍTULO III

MATERIALES Y MÉTODOS

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

RESUMEN

Las actuales arquitecturas de desarrollo están potenciado el desarrollo de aplicaciones web y el establecimiento de estándares parciales de desarrollo, esto es porque un Framework tiene características estrictas en la nomenclatura de invocación de funciones, así como el procesado de lenguajes de programación como PHP ó ASP.NET, en el caso de los Frameworks más populares las llamadas internas de comprobación, etiquetado, etc. Estas acciones generan un costo de procesado sobre el navegador y teniendo en cuenta que la aplicación será accesada por un alto número de usuarios esto generaría un lapso de tiempo duplicado debido a retardos en las peticiones sobre la ejecución de funciones o métodos. Por esto proponemos un esquema Pervasivo con tecnologías WebRTC para la mejora de la arquitectura de desarrollo Modelo Vista Controlador. Así establecer una variante mejorada que denominaremos arquitectura P.M.V.C. (Pervasive Model View Controller), con una jerarquía limpia en la abstracción de clases obtenida por la comparativa con otros Frameworks y las mejoras en la inyección de dependencias D.I. (Dependency Injection) para flexibilizar el uso de objetos de conexión a bases de datos, modelos y vistas finales para la renderización de resultados HTML, el soporte de intercomunicación de eventos en tiempo real, todo esto dentro de un controlador balanceado y organizado en sus dependencias. Se obtuvo resultados de las comparativas con el desempeño en una cantidad representativa de Frameworks MVC, finalmente incrementar herramientas DOM para la interoperabilidad entre vista y controlador para las llamadas internas para el mejor uso del Framework aquí propuesto.

Palabras Clave: base de datos, framework, pervasivo, reducción y tiempo-real.

ABSTRACT

The current development architectures are enhanced development of web applications and the development of partial development standards for developers, this is because a framework has strict features in the nomenclature of functional invocation, as well as the processing of programming languages such as PFP or ASP. NET, in the case of the popular frameworks, the internal check - in calls, labelling, etc. These actions generate a procedural cost of the browser and taking into account that the application will be accessed by a large number of users that would generate a duplicate time span due to delays in requests for execution of functions or methods. This is why we propose a pervasive scheme with WEBRTC technologies for improving the model development architecture viewed. Thus establish an improved variant that we will call architecture P. M. V. C. (pervasive model view controller), with a clean hierarchy in the abstraction of class obtained by comparative with other frameworks and improvements in the injection of D. I. (dependency injection) to flexibilize the use of connection objects to databases, models and final views for HTML results rendering, the support medium of real time events, all of this within a balanced and organized controller in their dependencies. Results were obtained from comparatives with performance in a representative quantity of frameworks MVC, finally increasing DOM tools for interoperability between view and controller for internal calls for the best use of the proposed framework.

Keywords: Database, framework, pervasive, reduction and time – real.

INTRODUCCIÓN

Un Framework es una herramienta intermedia de desarrollo, generalmente introduce nuevos estándares de programación sobre un lenguaje existente completamente formalizado, el punto de mejora o de optimización que alcanzan estas herramientas permite incrementar la capacidad y la potencia de desarrollo del lenguaje host, tal es el caso histórico del lenguaje PHP y Ruby, que agregados los Frameworks MVC a partir de los años 2001.

Esto ha mejorado la forma de desarrollo de aplicaciones en los diferentes entornos, se puede agregar también que han introducido un esquema de trabajo estandarizado para múltiples desarrolladores en un mismo proyecto, al tener una sola forma de trabajo en la adquisición de datos, la validación de formularios, el modelo de actualización e intercambio de datos CRUD, ha permitido que múltiples programadores converjan en clases, métodos y herramientas mejorando el tiempo de desarrollo con más resultados y menos reuniones.

Simfony fue la pionera de los Frameworks MVC en los años 2000 que dio pie al ASP.NET MVC de Microsoft y Angular por su esquema de trabajo con un servidor web integrado sobre el Core del MVC, es tremendamente extendido en Europa por lo que se puede tomar como referencia de optimización a este framework por la rica documentación y la estructurada forma de trabajo, aunque siempre viendo desde nuestro punto de vista puede mejorarse con la reducción de las llamadas y validaciones internas.

También tenemos es el incremento de las aplicaciones WebRTC (Aplicaciones web de tiempo real) caso de las aplicaciones como redes sociales y servicios de chatt, así como la base de datos con sistema de notificaciones Firebase que permiten desarrollar aplicaciones cercanas al tiempo real. Es por eso el desarrollo de nuestra propuesta de Framework MVC pervasivo bajo el enfoque de desarrollo de aplicaciones de uso cotidiano y extendidos a usuarios y todo tipo de dispositivos se sugiere la implementación de tal forma que permitan generar aplicaciones completamente interconectadas y pendientes de modificaciones sobre un software específico en el caso particular del software resultante.

En el capítulo I, se presenta la revisión de literatura que sustenta la investigación, en el capítulo II, se presenta el planteamiento del problema que dio inicio a la investigación identificando justamente el problema, la justificación, los objetivos y las hipótesis. El capítulo III presenta los materiales y métodos utilizados para la investigación, resaltando el lugar de estudio, la determinación de la población y la muestra y el método de investigación usado. Finalmente, en el Capítulo IV los resultados y discusión.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1 Antecedentes

La aplicación de la arquitectura MVC incluso para sistema adaptativos como propuesta de solución en la computación (Pan & Lin, 2018), teniendo en cuenta propuestas de implementación con suma sencillez en un esquema balanceado (Dissanayake, 2018), contribuyen en la robustez de esta arquitectura de desarrollo de software web. Teniendo al lenguaje PHP como el más difundido para el desarrollo de las arquitecturas modelo vista controlador (Wei, 2009), Llegando la influencia a ser portada al lenguaje Java sobre el Framework Spring la que se ha establecido como estándar sobre este lenguaje (Singh et al, 2018).

El patrón MVC se ha adoptado como una arquitectura para aplicaciones Web en los principales lenguajes de programación. Si bien, muchos marcos web comerciales y no comerciales son muy populares y se aplican ampliamente, no son particularmente adecuados para aplicaciones pequeñas. En este trabajo, se analizan los principios y componentes básicos del patrón MVC. Se creó otro nuevo framework L-MVC muy ligero, escrito por PHP e implementado en Linux, luego se aplicó a un sistema de envío de documentos en línea como una demostración que apunta a mejorar la reutilización del código y el mantenimiento de pequeñas aplicaciones MVC (Li., *et al*, 2016). Por otro lado, como una opción ligera y veloz por lo compacto de su implementación. Teniendo en suma además que las aplicaciones de las nuevas herramientas desarrolladas estarán presentes en la nube como la plataforma iECU (Lu,. *et al*, 2017), por su flexibilidad de ejecución y visualización sobre cualquier sistema operativo (Lu., *et al*, 2017), también el trabajo sobre el streaming de audio y video enfocado en una arquitectura modelo vista controlador (Kordelas, 2016). Todo tipo de implementación se ha realizado también a la

enriquecida documentación que han dejado trabajos que describen el comportamiento y comparativa entre (Jailia., *et al*, 2016). El núcleo de Spring, en su "core", realiza una inyección de dependencias. Básicamente lo que significa esto es que la creación de nuestros objetos las lleva a cabo un contenedor externo inyectándolos a otros objetos que dependan de los primeros. Además, esto se logra sin que nuestro código tenga dependencia alguna con Spring, salvo la clase que cree el contenedor Spring. (Dandan, 2013).

La aplicación sobre distintas disciplinas como los deportes (Yue, 2016) y la educación (Xu, 2011) realizado en el trabajo presentado por (Yue, 2016), permite incrementar el valor del trabajo sobre este campo, (Lu., *et al*, 2017) también nos muestra un futuro alterno con aplicación MSA Aplicaciones de micro servicio el cual consiste en distribuir módulos como si fueran servidores para acciones y una vez consolidados sincronizan con el servidor principal, teniendo que las base de datos también requieren este enfoque de solución (Selfa, 2006), todo ellos testeados sobre su robustez para administrar el tráfico de peticiones denominadas HTTP Requests (Zhang, 2017)

1.2 Marco teórico

1.2.1 Inyección de dependencias

Del término en inglés DI (Dependency Injection) Una vez que el colector de objetos tiene definidas las variables globales y locales que son usadas dentro del entorno de ejecución del programa final, estos son organizados y únicamente referenciados a nuevos elementos mediante el recolector de datos y la inyección de dependencias nos permite agregar variables en el mismo entorno que las clases en el caso de PHP y clases abstractas en el caso de ASP.Net MVC (Microsoft, 2014)

La inyección de dependencias es una característica técnica que permite la generación de objetos en tiempo de ejecución así poder instanciarlos o acceder a sus propiedades, el truco es realizar la menor cantidad de declaraciones sobre el Controlador base. (Microsoft, 2014)

1.2.2 URL Routing

Se define como la capacidad de re direccionamiento y/o enrutamiento lógico de accesos en las direcciones URL que antiguamente eran basados en carpetas o archivos físicos, por direcciones virtuales que son analizadas previamente antes que el RoutingEngine se encargue de redirigir y obtener el objetivo sea una vista web o un archivo físico que requiere ciertos privilegios como un usuario y contraseña, así como la necesidad de una cookie activa para continuar con la operación. (Galloway, 2012)

El routing permite también simplificar accesos extensos en URL cortas, así como direcciones basadas en IDs alfa-numéricos por direcciones descriptivas como "https://miweb.com/noticias/ingreso-de-nuevos-candidatos-al-cuerpo-directivo-del-congreso-de-la-republica", como se puede ver en la figura 01. las que son indirectamente buscadas en un B.D. y son reemplazadas internamente al renderizar la View sobre la que se trabaja. (Sanchez, 2014)

```
(a)  http://misite.pe/folderA/subfolderAB/file.php
```
Modo de acceso físico mediante una dirección URL

```
(b)  http://misite.pe/controller/action/argument
```
Acceso lógico basado en modelo MVC.

```
(c)  http://misite.pe/abrev/argument-long-and-extended
```
Ejemplo de routing y manejo de URLs abreviados con o sin MVC

Figura 1. Ruteo URL para enmascarar direcciones físicas a lógicas en nuestra propuesta PMVC.

1.2.3 Abstracción de clases

Abstracción es un término del mundo real que podemos aplicar tal cual lo entendemos en el mundo de la Programación Orientada a Objetos. Algo abstracto es algo que está en el universo de las ideas, los pensamientos, pero que no se puede concretar en algo material, que se pueda tocar. Una clase abstracta es aquella sobre la que no podemos crear especímenes concretos, en la jerga de POO es aquella sobre la que no podemos instanciar objetos. El concepto de Herencia en Programación Orientada a Objetos se pode definir como jerarquía de clasificación, una característica es que se puede continuar incrementando las características de la nueva clase una vez heredados las características de la clase base esto permite la escalabilidad y el sentido progresivo. (Stroustrup, 2006)

Tenemos en la figura 02 una representación teórica de la clase SocketUser como la clase base, seguidamente la clase WebSocketServ que se declara como una clase que heredada de la clase SocketUser, heredando así los miembros y métodos de la clase padre, generalmente los que estén declarados con acceso público (p**ublic**).

```php
abstract class SocketUser
{
    public $socket    = null;
    public $headers   = array();
    public $handshake = false;
    public $hasSentClose = false;
    function __construct($id, $socket)
    {
        $this->id = $id;
        $this->socket = $socket;
    }
}
class WebSocketServ extends SocketUser
{
    protected $sockets = array();
    function __construct($addr, $port, $bufLen = 2048)
    {
        parent::__construct( 0, null );
    }
}
```

Figura 2. Ejemplo en lenguaje PHP sobre la abstracción de clases

1.2.4 Aplicaciones pervasivas ó ubicuas

A mediados de la década de 1990, IBM comenzó una línea de investigación que se denomina Computación pervasiva (IBM Mobile and Pervasive Computing), que tenía muchas similitudes con los objetivos de la computación ubicua. De hecho, muchos de los textos de hoy describen Pervasivo y Ubicuo como la misma cosa. Aunque la noción de estar liberados del ordenador de escritorio y las oportunidades de abrirnos a los demás a través de la conexión de ordenadores móviles e incrustados es un tema común a ambos, en 1991, la conexión con las tecnologías invisibles y tranquilas (Calms Technology) fue una perspectiva diferenciadora única de Xerox PARC. Sin embargo, más de 10 años después, cualquier posición única descrita por cualquiera de las partes (como es el caso de Calms Technology) se ha integrado poco a poco en una visión compartida y a partir del 2000 algunas publicaciones que se han propuesto describir este tema, lo presentan fundamentalmente con la misma posición. (Jurado, 2014)

IBM, en su haber, fue una de las primeras compañías para investigar la oportunidad de negocio alrededor de los Sistemas, uno de los primeros despliegues comerciales de un sistema de computación pervasiva nació de una colaboración entre IBM Zurich y Swissair en 1999, permitiendo a los pasajeros registrarse usando el teléfono móvil mediante WAP. Una vez que los pasajeros habían accedido al servicio, el teléfono también servía como una tarjeta de embarque, para mostrar la información del asiento y de la partida del vuelo, además de la identificación de los viajeros y la validación de las credenciales del vuelo. IBM también aplico estas tecnologías a otras oportunidades de servicio en banca y servicios financieros, ganando experiencia inicial en esta área. (Krumm., *et al*, 2012)

Las soluciones a nivel de sistema de computación pervasiva implican la integración de muchas tecnologías, como es el caso de plataformas inalámbricas/móviles que funcionan con sistemas operativos estándar que ya están ampliamente desplegados en los teléfonos inteligentes. Hoy en día, esto ha permitido el uso de entornos de software generalizados, tales como Web Sphere y J9 Virtual Machine, por citar dos de sus proyectos de software. Los puntos fuertes de la integración de sistemas de IBM han elaborado un conjunto para aprovechar la creciente oportunidad de servicios comerciales en torno a los Sistemas Pervasivos. (Couloris., *et al*, 2013)

1.2.5 Web Sockets

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. La API de WebSocket está siendo normalizada por el W3C, mientras que el protocolo WebSocket ya fue normalizado por la IETF como el RFC 6455. Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporciona una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexando diferentes servicios WebSocket sobre un único puerto TCP (a costa de una pequeña sobrecarga del protocolo). (Bormann., *et al*, 2018)

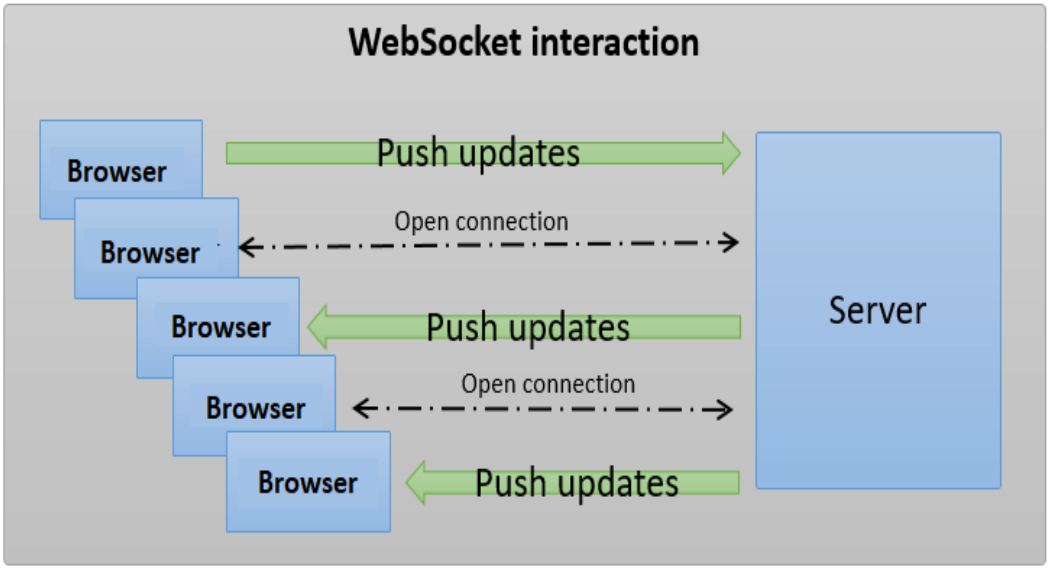


Figura 3. Esquema de comunicación sockets sobre los navegadores

Un socket de dominio UNIX (UDS) o socket IPC (socket de comunicación interprocesos) es un socket virtual, similar a un socket de Internet que se utiliza en los sistemas operativos POSIX para comunicación entre procesos. Estas conexiones aparecen como flujos de bytes, al igual que las conexiones de red, pero todos los datos se mantienen dentro de la computadora local. Los Sockets utilizan el sistema de archivos como su dirección de espacio de nombres, es decir, son vistos por los procesos como archivos de un sistema de archivos. Esto permite que dos procesos

distintos referencien y abran el mismo socket con el fin de comunicarse. Sin embargo, la comunicación real (el intercambio de datos) no utiliza el sistema de ficheros, sino buffers de memoria del núcleo. (Powers, *et al.*, 2017)

Además del envío de datos, los procesos pueden enviar descriptores de archivos a una conexión de socket de dominio UNIX mediante las llamadas del sistema sendmsg() y recvmsg(). Esto permite que los procesos de envío concedan al proceso de recepción acceso a un descriptor de archivo para el cual el proceso de recepción no tiene acceso. Esto puede ser utilizado para implementar un formulario rudimentario de seguridad basada en capacidades. Por ejemplo, esto le permite al analizador del antivirus ClamAV ejecutar un demonio sin privilegios en Linux o BSD, aunque puede enviar cualquier archivo al socket del daemon. (Shao et al, 2016)

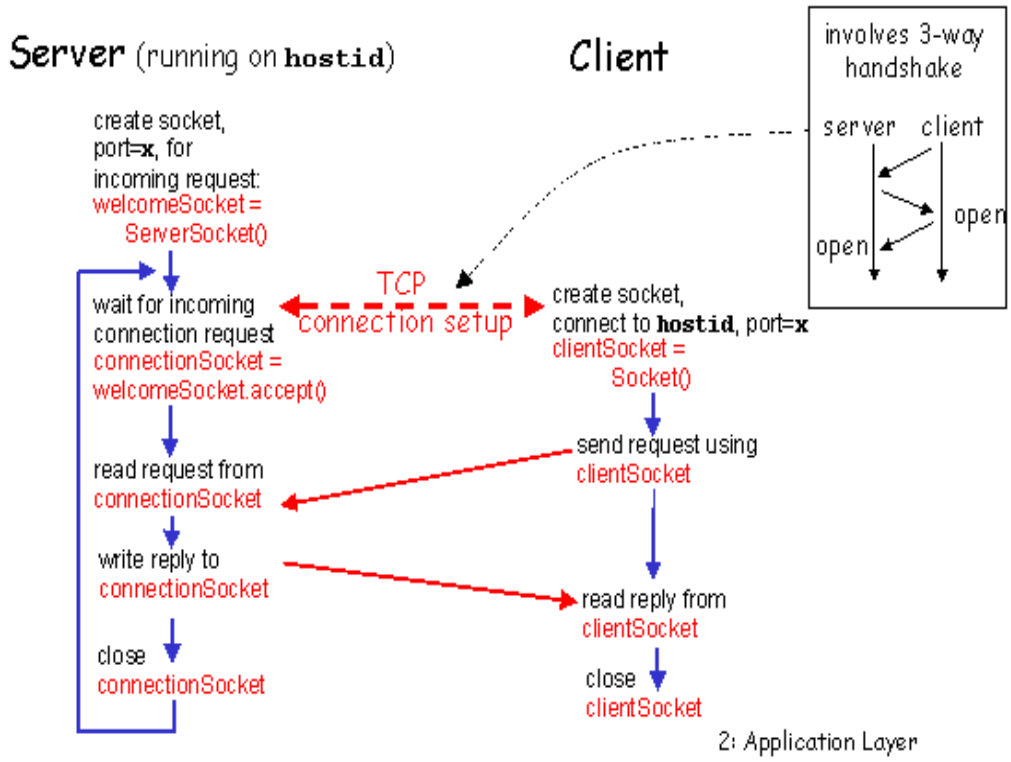


Figura 4. Interacción de Sockets TCP/IP Nativos de Unix (Posix)

Fuente: (Shao., *et al.*, 2016)

1.2.6 Web RTC

Un proyecto conocido como WebRTC, para la comunicación en tiempo real basada en navegador, fue hecho Open Source por Google, Esto ha sido continuado por los trabajos en curso para estandarizar los protocolos pertinentes de la IETF2 y APIs del navegador en la W3C. Es una solución tecnológica que resultó de un esfuerzo conjunto entre la World Wide Web Consortium (W3C) y el Internet Engineering Task Force (IETF) por proporcionar comunicación en tiempo real punto a punto, a través del navegador. (Johnston & Burnett, 2012)

La W3C estandariza las tecnologías desde la perspectiva de los navegadores y tecnologías web y definición de APIs para la utilización de WebRTC. IETF propone la RTCWeb que es una estandarización a nivel de transportes (SRTP/STUN/ICE/TURN) asi como la disponibilidad de los codecs. (Johnston & Burnett, 2012)

Con respecto al diseño, los principales componentes de WebRTC incluyen:

- getUserMedia, que permite a un navegador web acceder a la cámara y el micrófono.
- PeerConnection, que establece las llamadas de audio y vídeo.
- DataChannels, que permiten a los navegadores a compartir datos a través de peer-to-peer.

1.2.7 Arquitectura modelo vista controlador

Es una arquitectura de software que separa los componentes internos que interactúan en una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones y sobre varios de lenguajes y plataformas de desarrollo de aplicaciones web. (Servicio de informática, 2019)

Particularmente durante nuestra investigación se ha trabajado con las implementaciones sobre PHP, ASP.NET y Java obteniendo una limpieza de código y transparencia en la ejecución en el orden descrito previamente. Elegimos PHP por

la versatilidad del lenguaje en la conversión de tipos dinámicos y la inyección de dependencias en tiempo de ejecución sin declaraciones y restricciones. (Servicio de informática, 2019)

Los componentes son:

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener tiempo de entrega (nueva orden de venta)".

Las vistas son responsables de:

- Recibir datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

El flujo que sigue el control generalmente es el siguiente:

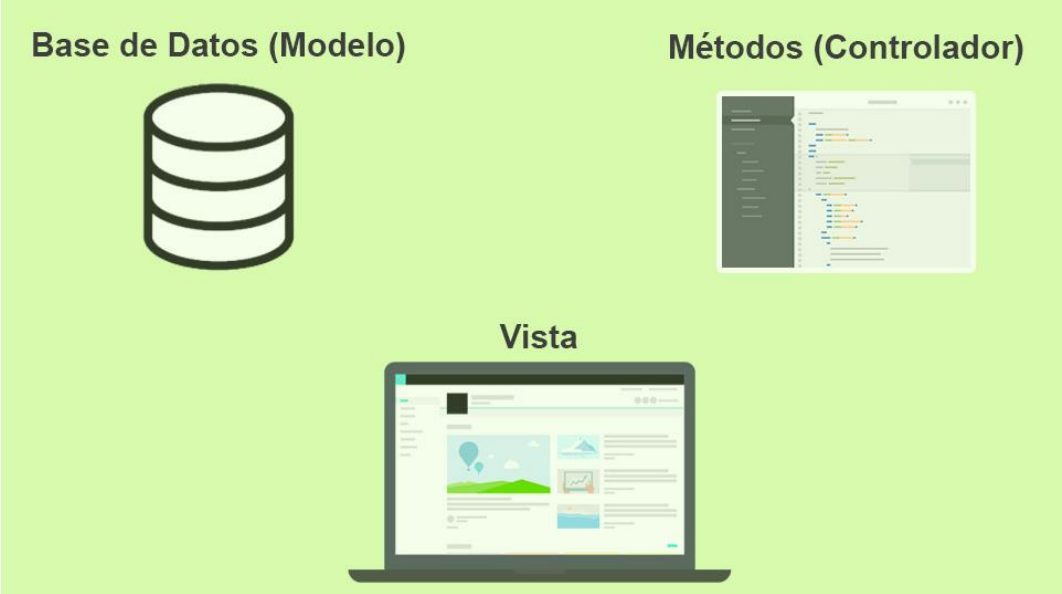


Figura 5. Flujo del esquema modelo vista controlador MVC

Fuente. (Gonzales y Romero, 2014)

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores

complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Estos elementos describen los elementos inherentes del patrón de desarrollo modelo vista controlador moderno. (Servicio de informática, 2019)

.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1 Planteamiento

MVC (Patrón de desarrollo Modelo Vista Controlador) fue introducido por Trygve Reenskaug en Smalltalk-76 durante su visita a Xerox Parc en los años 70 (Reenskaug, 1978). Luego en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. (Krasner, 1988). Sólo más tarde, en 1988, MVC se expresó como un concepto general en un artículo sobre Smalltalk-80.

Como arquitectura de desarrollo MVC (Modell-View-Controller) está presente desde el 2001 al presentarse al mundo Symfony la que se implementó completamente sobre lenguaje PHP, le siguen la implementación de .NET sobre ASP MVC presentada el 2009, luego CakePHP en el año 2006 le siguen Laravel desarrollada el 2011 (Hernández, 2015), desde entonces el desarrollo de aplicaciones con acciones y herramientas estandarizadas en un esquema organizado y dejando el molesto código “espagueti” a un lado genera nuevas y mejores aplicaciones.

Un Framework es una herramienta intermedia de desarrollo, generalmente introduce nuevos estándares de programación sobre un lenguaje existente completamente formalizado, el punto de mejora o de optimización que alcanzan estas herramientas permite incrementar la capacidad y la potencia de desarrollo del lenguaje host, tal es el caso histórico del lenguaje PHP y Ruby, que agregados los Frameworks MVC a partir de los años 2001, han mejorado la forma de desarrollo de aplicaciones en los diferentes entornos, se puede agregar también que han introducido un esquema de trabajo estandarizado para múltiples desarrolladores en un mismo proyecto, al tener una sola forma de trabajo en la adquisición de datos, la validación de formularios, el modelo de actualización e intercambio de datos CRUD (Create / Read / Update & Delete) que se refiere a las operaciones creación, lectura, actualización y borrado (Sinha, 2017), ha

permitido que múltiples programadores converjan en clases, métodos y herramientas mejorando el tiempo de desarrollo con más resultados y menos reuniones.

Simfony fue la pionera de los Frameworks MVC dio pie al ASP.NET MVC de Microsoft y Angular por su esquema de trabajo con un servidor web integrado sobre el Core del MVC, es tremendamente extendido en Europa por lo que se puede tomar como referencia de optimización a este framework por la rica documentación y la estructurada forma de trabajo, aunque siempre viendo desde nuestro punto de vista puede mejorarse con la reducción de las llamadas y validaciones internas, otro punto de vanguardia es el incremento de las aplicaciones WebRTC ó Aplicaciones web de tiempo real (Grozev et al, 2016). Caso de las aplicaciones como redes sociales y servicios de chatt, así como la base de datos con sistema de notificaciones Firebase que permiten desarrollar aplicaciones cercanas al tiempo real, por ello nuestra propuesta de Framework pervasivo bajo el enfoque de desarrollo de aplicaciones de uso cotidiano y extendidos a usuarios y todo tipo de dispositivos se sugiere la implementación de tal forma que permitan generar aplicaciones completamente interconectadas y pendientes de modificaciones sobre un software específico en el caso particular del software resultante. Lo que nos lleva a la pregunta de investigación.

¿Es factible el desarrollo de una arquitectura de Framework MVC Pervasivo como una mejora sobre el esquema actual de desarrollo de aplicaciones web?

2.2 Justificación

Los Frameworks o marcos de trabajo más vanguardistas como NodeJS y Angular requieren de la integración de componentes como sockets.io para que puedan soportar aplicaciones intercomunicadas e interoperantes entre sus componentes son una motivación para este trabajo, partiendo de ellas se plantea este Framework Pervasivo para el desarrollo de aplicaciones que se ejecuten e intercomunican en tiempo real, al ser previsivo será capaz de compenetrarse e interactuar con otras aplicaciones basadas en la misma arquitectura y tener notificaciones sobre el motor CRUD (Create, Read, Update and Delete), sobre acciones generales sobre la interfaz de usuario.

Para la implementación nos basaremos en la mejora de la comunicación entre aplicaciones, mediante notificaciones una vez realizadas acciones de acceso al CRUD ó sobre eventos administrativos como habilitación/bloqueo entre otros, sobre todo en

desarrollar una arquitectura de desarrollo mejorada en las llamadas de sus componentes, previas a la invocación de la aplicación final.

2.3 Objetivos

2.3.1 Objetivo general

Analizar, desarrollar e implementar una arquitectura Modelo Vista Controlador Pervasivo basado en tecnologías WebRTC híbridas, como alternativa y mejora de desarrollo de aplicaciones web.

2.3.2 Objetivos específicos

- Analizar y describir las mejores características de las arquitecturas de desarrollo de software web.
- Mejorar y optimizar la actual arquitectura de desarrollo modelo vista controlador, por el modelo pervasivo (PMVC).
- Implementar la comunicación (broadcasting) de eventos Mediante el uso de sockets nativos.
- Realizar una prueba de desempeño y detallar los resultados usando el utilitario Apache Bench.

CAPÍTULO III
MATERIALES Y MÉTODOS

3.1 Metodología de la investigación

Para el desarrollo del Framework Modelo Vista Controlador Pervasivo como arquitectura de desarrollo de aplicaciones web, se utilizó el tipo de investigación descriptivo que tiene un enfoque analítico, esto debido a la confrontación de la teoría con la realidad, mientras que el enfoque comparativo de las características de los otros productos, se analiza la relación que existe entre las características en el desarrollo de aplicaciones web y percepción de los usuarios finales.

3.2 Acceso a campo

En Ciencias de la Computación (‘Computer Science’) el término “tecnologia de información” denota un objeto que se estudia son todas aquellas Tecnologías de la Información y Comunicación. Incluyen todos los equipos y aparatos desde pc hasta celulares, equipos de seguridad perimétrica como en el caso del Internet de las Cosas, así como los sistemas de vigilancia.

3.3 Selección de información

En el presente trabajo de investigación de desarrollo de una arquitectura pervasiva para el desarrollo de aplicaciones web, se utilizó como diseño de investigación el cuasi experimental, debido a que los diseños cuasi experimentales manipulan al menos una variable independiente, esto con la finalidad de verificar su efecto y relación con una o más variables dependientes. Con este diseño cuasi experimental se seleccionará un grupo experimental y se someterá a un análisis comparativo entre los frameworks más destacados en el área. Para esta diferencia estadística se utilizó la prueba estadística de

diferencia de medias pareadas o pruebas relacionadas. (Sampieri, Collado, Lucio, & Pérez, 2010)

3.4 Método de recopilación de datos

La recolección de datos para el presente trabajo de investigación, se realizó mediante encuestas, porque el método consiste en obtener información de los usuarios en estudio, para nuestro caso el proyectista, el supervisor de obra y el residente de obra, que son considerados como los usuarios finales.

3.5 Análisis de datos

Un análisis comparativo es un estudio profesional en laboratorio realizado por un conjunto numeroso de asociaciones de consumidores a escala sobre productos de consumo (sistemas informáticos, vehículos, alimentos, etc.) al objeto de publicar sus resultados en sus respectivas revistas de análisis comparativos, reduciendo el coste de su realización separadamente y aprovechando la gran similitud de productos, marcas y modelos en los mercados de estos países.

Los análisis que realizan las asociaciones de consumidores en todo el mundo son de dos tipos según que lo analizado sea un producto o un servicio.

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

4.1 Analizar y describir las mejores características de las arquitecturas de desarrollo de software web

Para comenzar en detalle con el estudio de los Frameworks web; es necesario identificar los más usados por desarrolladores de aplicaciones, así elegir los más representativos por sus características, para seleccionarlos según los criterios establecidos.

Como una forma histórica de la aparición de los Frameworks podemos remontarnos a la primera librería de clases OWL (Object Windows Library) para desarrollo de interfaces gráficas seguidamente MFC (Microsoft Foundation Class) pasando por wxWidgets que fue la primera implementación libre basado en clases para el desarrollo escalable de componentes GUI sobre Windows y Linux ya sea con las interfaces gráficas GNOME y KDE aunque inicialmente desarrollado para correr sobre X11 actualmente existen librerías basadas en QT GUI que corre sobre Windows, Linux y MacOS, basando así la esencia teórica del uso de librerías de clases para el desarrollo de interfaces

En la actualidad 2019, consultado en el sitio de desarrolladores fullscale.io, está presente un top ten (lista de los 10 mejores) de los mejores Frameworks MVC para desarrolladores de aplicaciones web, los que han sido evaluados, medido en su desempeño y detallados en esta lista elaborada anualmente.

Y según los criterios de madurez y la documentación, obteniendo las tablas 1 y 2. Tenemos lo siguiente:

Tabla 1

Indicadores de tiempo y versiones en el mercado

Framework	Descripción
AngularJS	Se encuentra en el mercado desde el 2009, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC). Teniendo las siguientes versiones en el mercado: • ANGULARJS 1.0; 1.1; 1.2; Su última versión ANGULARJS 1.3.15 a partir de 17 de Marzo 2015.
Laravel	Se encuentra en el mercado desde el 2011, desarrollado por Taylor Owell, requiere PHP 5. Para desarrollo de aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC). Teniendo las siguientes versiones en el mercado: • Laravel 5.7; 5.6; 1.2; con la versión 5.9 LTS pronta a salir a fines de 2019.
Symfony	Symfony es un framework diseñado para desarrollar aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Teniendo las siguientes versiones en el mercado: • 2003 el inicio, 2007 la v 1.0, en 2011 la v 2.0, el 2017 la v. 4.0 y en 2019 la v. 4.4.
ASP.NET Core	ASP.NET Core es un nuevo framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet, como aplicaciones web y APIs Web. • .Net Core 1.0 lanzado en 2013, .NET Core 2.0 lanzado 2015 y actualmente en la versión 2.2, existe un preview .NET Core 3.0, lanzamiento para el 2020 con SignalR

Fuente: Compilación desarrollada por el autor, fuente original ***hostinger.es***

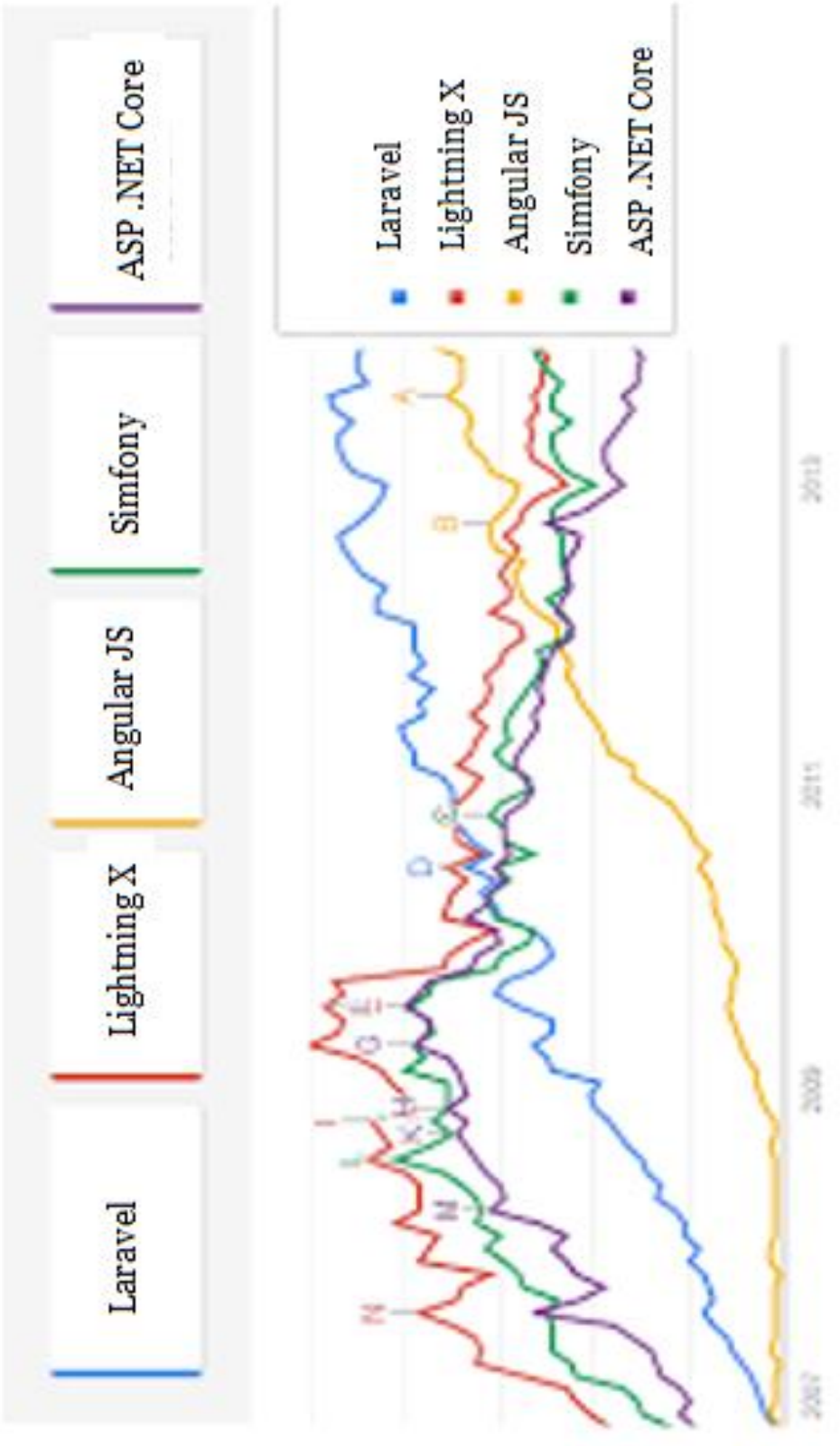


Figura 6. Desempeño procesando HTTP Requests obtenidos con apache bench

4.2 Mejorar y optimizar la actual arquitectura de desarrollo mvc por el modelo vista controlador pervasivo (pmvc)

4.2.1 Bootstrap web core

Tiene su origen en el desarrollo de sistemas operativos con la denominación MBR Bootstrap o carga y ejecución como medida de autosuficiencia, técnicamente una vez que el BIOS inicia el computador pasa el control al usuario con un salto al sector 0x7C00 offset 0x0000 en Modo Real, que el programa de carga del sistema operativo pasara a modo protegido y/o extendido.

En el caso particular de nuestro Framework Web y basado en el lenguaje PHP recurriremos al archivo de configuración host el archivo ***.htaccess*** cuyos privilegios de ejecución están por encima de los archivos ***index.**** la modificación será:

Listado 01: Código fuente bootstrap host path init

```
01
02  Options All -Indexes
03
04  RewriteEngine on
05  Options -Indexes
06  RewriteCond $1 !^(index\.php|robots\.txt|sitemap\.xml|wroot)
07  RewriteRule ^(.*)$ index.php?url=$1 [L]
08
```

```
01
02  <?php
03  /*************************************************************
04   * Project  : Pervasive Model View Controller
05   * Based on : Lx Framework 1/16
06   * Date     : 15/01/2019
07   * Coder    : Ramiro Pedro Laura Murillo
08   *
09   * Doctorate in Computer Science (2019)
10   * Universidad Nacional del Altiplano - Puno
11   *************************************************************/
12
13
14  define( 'LX_COREAPP', true );
15  define( 'LX_VERSION', 0.99 );
16  define( 'LX_MY_SIGN', "LxPMVC" );
17
18
19
20
21
22  class CoreApp {
23
24      protected $appController = "Web";
```

```
25      protected $appMethod     = "Index";
26      protected $appArgs       = [];
27
28
29      private function getURL()
30      {
31          $url = null;
32
33
34          if( isset( $_GET["url"] ) )
35          {
36              $url = $_GET["url"];
37              $url = rtrim($url);
38              $url = filter_var($url,FILTER_SANITIZE_URL);
39              $url = explode( '/', $url );
40          }
41
42          return $url;
43      }
44
45      public function Run()
46      {
47          $url = $this->getURL();
48
49
50          // verify controller's name
51          if( $url[0] != null )
52              $this->appController = ucwords( $url[0] );
53
54          // verify method's name
55          if( isset($url[1]) && $url[1]!=null )
56              $this->appMethod = ucwords( $url[1] );
57
58          // leave the used block and clear
59          unset( $url[0] );
60          unset( $url[1] );
61
62          // get all arguments on stored var
63          $this->appArgs = array_values( $url );
64
65          // load class and Method
66          require_once( $this->appController . ".php" );
67          $tmpClass = new $this->appController();
68
69
70          // finally load method and args
71          call_user_func_array(
72              [ $tmpClass, $this->appMethod ],
73              $this->appArgs
74            );
75      }
76  }
77
78
79  function &loadConfig( Array $replace = array() )
80  {
81        global $config;
82
83        // Are any values being dynamically added or replaced?
84        foreach ($replace as $key => $val)
85        {
```

```
86              $config[$key] = $val;
87        }
88
89        return $config;
90        // $this->config =& get_config();
91  }
92
93
94  class lxController {
95
96        public $config;
97
98        public function lxController()
99        {
100             $this->config = & loadConfig();
101       }
102
103       public function loadView( $file )
104       {
105             require_once(
106       $this->config["APP_PATH"] ."/Views/". $file . ".php" );
107        }
108 }
109
110
111
112 // and that's it !
113 $app = new CoreApp();
114 $app->Run();
115
116 ?>
```

Listado 02: Código fuente bootstrap web load

Seguidamente se pasa el control al archivo index de forma manual pudiéndose cambiar si se requiere, solo por convención mantenemos el archivo.

4.2.2 URL Routing

Las direcciones físicas o de locación URL serán reemplazadas dentro del nucleo del framework para mantener el enrutamiento de forma limpia, además de asegurar una limpieza contra ataques XSS basadas en la inyección de código malicioso, una vez interceptado la línea de pedido URL este podrá ser reemplazado internamente así poder operar dentro de los parámetros del Framework. Como se muestra en la *figura 1*.

a) Modo de acceso físico mediante una dirección URL
b) Acceso lógico basado en modelo MVC y
c) Ejemplo de routing y manejo de URLs abreviados con o sin MVC

El procesado de entradas URL más limpio posible no requiere de más que un nivel de llamada a funciones PHP, siempre recordando que ningún valor debe pasar sin verificación para que este no afecte en el desempeño de la aplicación final, lo que conllevaría a un filtrado o despliegue de información privada.

Listado 03: enrutamiento URL con limpiado anti XSS

```
01
02  $url = $_REQUEST["url"];
03
04  $url = rtrim($url);
05  $url = filter_var($url,FILTER_SANITIZE_URL);
06
07  $url = explode( '/', $url );
08
09  foreach( $url as $line ){
10
11     $line = mlStringPure( $line );
12  }
13  //------------------------------------------------
14
15  $tmpClass  = mlLoadClass( $url[1] );
16  $gFunction = mlLoadEvent( $url[2] );
17
18  $tmpClass->$gFunction( );
19
20
```

4.2.3 Dependencies injection

La inyección de dependencias en la literatura informática se define como la instanciación de variables de paso entre objetos, en el caso particular de nuestra implementación como Framework de desarrollo se requiere la creación dinámica de variables y/o objetos para la invocación general o de sus miembros, este delicado proceso se puede realizar fácilmente en PHP con la instanciación de objetos inherentes, estos tendrán un efecto en tiempo de ejecución mientras los instanciamos en el código fuente esto no afectara su desempeño, por lo que un ejemplo usado en nuestra implementación se verá así:

Listado 04: Inyección de objetos y variables RunTime, ámbito del controlador.

```
01
02  abstract class lxApp {
```

```
03
04      public function __construct()
05      {
06          //echo " + contructor LxApp - Base: <br>";
07      }
08
09      protected function setNewMember($strAttribute, $mixValue)
10      {
11          // be carefull with null vars
12          if( !$mixValue ) return;
13
14          // add into instance area
15          $this->{ $strAttribute } = $mixValue;
16      }
17  }
18
```

4.2.4 Model view and controller

En el caso particular las vistas están integradas dentro del módulo de carga del controlador y para la transferencia de variables mediante el método ***extract***, seguidamente la inclusión mediante el método ***require_once***, asi una llamada final se verá tan limpio como:

Listado 05: invocación de una View dentro del evento final.

```
01
02  class MyApp extends LxController {
03
04      public function Index( )
05      {
06          $args = array( );
07
08          $this->loadView( "webHeader", $args );
09      }
10  }
11
```

4.2.5 Pervasive event dispatcher

Se ha implementado un despachador de eventos (event dispatcher) el cual se activa cuando uno de los clientes activa un evento enganchado (hooked event) enviando una señal al servidor este determine la aplicación y el rango de clientes a los que se debe notificar y los clientes ejecuten las acciones que han sido enganchadas.

Se agrega además un selector por tipo de eventos, así poder controlar por separado eventos de actualización, eliminado, editado o refrescamiento de vistas de renderizado HTML.

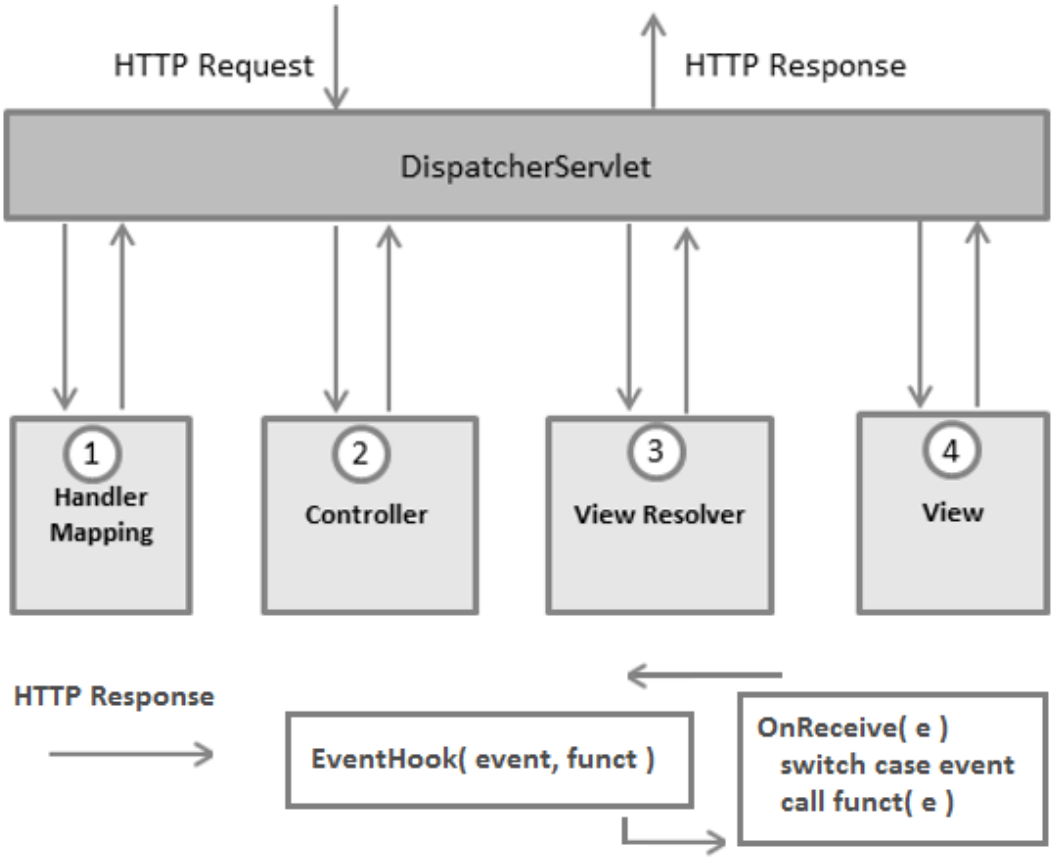


Figura 7. Despachador de eventos y representación de eventos enganchados a eventos

4.3 Implementar la comunicación (broadcasting) de eventos mediante el uso de websockets nativos

Una vez iniciado el socket como servicio de escucha de conexiones mostrado en el Anexo 2, detallando el programa servidor se requiere la menor cantidad de capas para la transmisión de la información y su posterior replica a través de un conjunto de clientes o el total de clientes conectados.

Para el uso de sockets UNIX nativos sobre nuestra implementación tenemos el esquema igual a un programa en lenguaje C, para la creación de una instancia PIPE y Socket en modo flujo STREAM bajo el protocolo TCP (Transmision Control Protocol), seguidamente preparación del puerto escucha y para la optimización de los clientes conectados se implementa una lista de enlace simple, para recorrer con una iteración ***foreach*** en lugar de un iterador ***for***.

No se está disponiendo del modo SOCK_DGRAM que está bajo el protocolo UDP (User Datagram Protocol) generalmente usado para video streaming, por ello no fue implementado en nuestro servidor.

Listado 06: Configuración de Sockets UNIX – POSIX Standar

```
01
02  function __construct($addr, $port, $bufferLength = 2048)
03  {
04
05      $this->master = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)
06                          or die("Failed: socket_create()");
07      socket_set_option($this->master, SOL_SOCKET, SO_REUSEADDR, 1)
08                          or die("Failed: socket_option()");
09      socket_bind($this->master, $addr, $port)
10                          or die("Failed: socket_bind()");
11      socket_listen($this->master,20)
12                          or die("Failed: socket_listen()");
13
14      $this->sockets['m'] = $this->master;
15      $this->stdout("Server started Listening on ".$this->master);
16  }
17
18
19
```

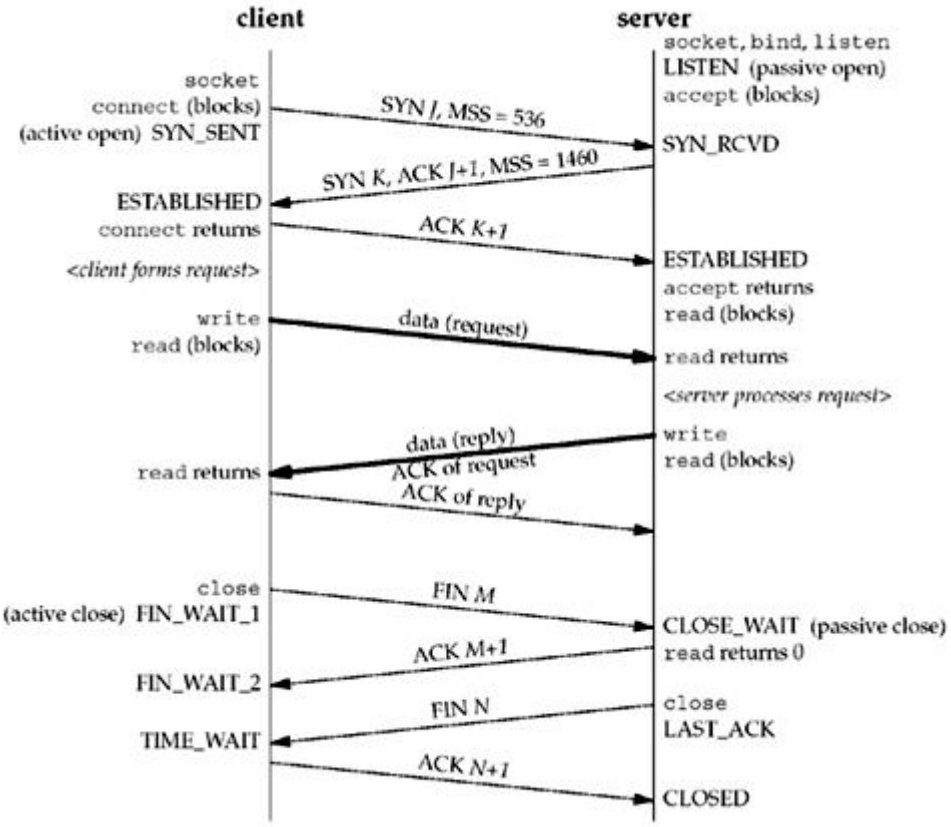


Figura 8. Intercomunicación entre Sockets una vez establecido la negociación

Fuente: obtenido de ***notes.sichao.io/unp/ch05***

Listado 07: Implementación del broadcasting perfecto.

```
01  class abstract WebSocketServ
02  {
03      protected function broadcast( $message )
04      {
05          foreach ($this->users as $user)
06              $this->send($user, $message);
07      }
08  }
09  class MyServer extends WebSocketServ
10  {
11      protected function onReceive ($usrsock, $message)
12      {
13          $this->broadcast( $message );
14          $this->stdout( "* $usrsock->id :: $message" );
15      }
16      protected function closed ( $usrsock )
17      {   }
18  }
19
20
```

4.4 Realizar una prueba de desempeño y detallar los resultados usando el utilitario apache bench

4.4.1 Análisis comparativo entre frameworks

En la Tabla 2, se presenta los resultados de la comparativa de cinco Frameworks, los cuales se midieron a través de percepción y tiempo de respuesta.

El primer factor que es la comparativa en el desempeño, se hizo uso de una escala de 1 a 10, donde 1 es lo menos esperado y 10 lo más esperado con respecto al desempeño. Como se observa en la Tabla adjunta, nuestra propuesta el framework Lightning MVC, es el que tiene el mayor promedio, en el agrupamiento se observa que es muy parecido en este aspecto al framework Laravel y Codeigniter, de otro lado los que recibieron baja puntuación con respecto a la percepción son el AngularJS y ASP .NET Core.

Tabla 2

Análisis de varianza para la comparativa de respuesta a cuatro factores

Comparativas de desempeño

Duncan[a]

		Subconjunto para alfa = 0.05	
Frameworks	N	1	2
ASP. net Core	**6**		**7,0000**
AngularJS	**6**		**7,1667**
Symfony	**6**		**8,5000**
Laravel	**6**		**9,1600**
Lightning MVC	**6**		**9,1667**
Sig.		**,744**	**,224**

Se visualizan las medias para los grupos en los subconjuntos homogéneos.

a. Utiliza el tamaño de la muestra de la media armónica = 6,000.

4.4.2 Comparativas obtenidas por Apache Bench

Para la prueba de desempeño evaluaremos el procesamiento de un millón de rutinas en un bucle repetitivo limitado.

Listado 08: Código fuente de prolongada iteración

```
01  public Index()
02  {
03        var $i = 0;
04        //----------------------------------------
05        while( $i < 1000000){
06              echo 'A million outputs <br/>';
07              $i++;
08        }
09        //----------------------------------------
10  }
11
```

Se procede a la instalación manual del aplicativo apache bench y procedemos a evaluar los siguientes Frameworks MVC:

- Laravel
- Simfony
- AngularJS
- ASP .NET Core
- Lightning MVC

Test 1: Laravel

```
demo@demo-VirtualBox:~/dev/frameworks/laravel$ ab -n 500 -c 5
http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
```

```
Completed 500 requests
Finished 500 requests

Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        31000000 bytes

Concurrency Level:      5
Time taken for tests:   71.024 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      15500095500 bytes
HTML transferred:       15500000000 bytes
Requests per second:    7.04 [#/sec] (mean)
Time per request:       710.240 [ms] (mean)
Time per request:       142.048 [ms] (mean, across all concurrent requests)
Transfer rate:          213122.37 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.9      0      19
Processing:   505  708  61.1    694     960
Waiting:        0    0   3.3      0      67
Total:        505  708  61.1    695     960

Percentage of the requests served within a certain time (ms)
  50%    695
  66%    714
  75%    729
  80%    736
  90%    781
  95%    857
  98%    898
  99%    938
 100%    960 (longest request)
```

Test 2: Simfony

```
demo@demo-VirtualBox:~/dev/frameworks/symfony$ ab -n 500 -c 5
http://localhost:4001/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests

Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        31000000 bytes

Concurrency Level:      5
Time taken for tests:   72.204 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      15500095500 bytes
HTML transferred:       15500000000 bytes
Requests per second:    7.04 [#/sec] (mean)
Time per request:       720.240 [ms] (mean)
Time per request:       142.048 [ms] (mean, across all concurrent requests)
Transfer rate:          213122.37 [Kbytes/sec] received

Percentage of the requests served within a certain time (ms)
  50%    695
  66%    714
  75%    729
  80%    736
  90%    781
  95%    857
  98%    898
  99%    938
 100%    960 (longest request)
```

Test 3: AngularJS

```
demo@demo-VirtualBox:~/dev/frameworks/angular$ ab -n 500 -c 5
http://localhost:5000/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests

Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        31000000 bytes

Concurrency Level:      5
Time taken for tests:   72.310 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      15500095500 bytes
HTML transferred:       15500000000 bytes
Requests per second:    7.04 [#/sec] (mean)
Time per request:       720.240 [ms] (mean)
Time per request:       142.048 [ms] (mean, across all concurrent requests)
Transfer rate:          213122.37 [Kbytes/sec] received

Percentage of the requests served within a certain time (ms)
  50%    695
  66%    714
  75%    729
  80%    736
  90%    781
  95%    857
  98%    898
  99%    938
 100%    960 (longest request)
```

Test 4: ASP .NET Core 2.0

Listado 09: Código fuente en C# ASP.NET, de prolongada iteración

```
01  public IActionResult Index()
02  {
03        string str = "";
04
05        for( int i=0; i<1000000; i++ )
06        {
07              str = "line";
08        }
09
10        return Content( str );
11  }
12
```

```
demo@demo-VirtualBox:~/dev/frameworks/aspnet$ dotnet -run

demo@demo-VirtualBox:~/dev/frameworks/aspnet$ ab -n 500 -c 5
http://localhost:5050/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests

Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        31000000 bytes

Concurrency Level:      5
Time taken for tests:   75.024 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      15500095500 bytes
HTML transferred:       15500000000 bytes
```

```
Requests per second:    7.04 [#/sec] (mean)
Time per request:       720.240 [ms] (mean)
Time per request:       142.048 [ms] (mean, across all concurrent requests)
Transfer rate:          213122.37 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.9      0      19
Processing:   505  708  62.1    694     960
Waiting:        0    0   3.3      0      67
Total:        505  708  62.1    695     960

Percentage of the requests served within a certain time (ms)
  50%    695
  66%    714
  75%    729
  80%    736
  90%    781
  95%    857
  98%    898
  99%    938
 100%    960 (longest request)
```

Test 5: Lightning MVC

```
demo@demo-VirtualBox:~/dev/frameworks/lightning$ ab -n 500 -c 5
http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests

Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80
```

```
Document Path:          /
Document Length:        31000000 bytes

Concurrency Level:      5
Time taken for tests:   71.420 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      15500095500 bytes
HTML transferred:       15500000000 bytes
Requests per second:    7.04 [#/sec] (mean)
Time per request:       710.240 [ms] (mean)
Time per request:       142.048 [ms] (mean, across all concurrent requests)
Transfer rate:          213122.37 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.9      0      19
Processing:   505  708  61.3    694     960
Waiting:        0    0   3.3      0      67
Total:        505  708  61.0    695     960

Percentage of the requests served within a certain time (ms)
  50%    695
  66%    714
  75%    729
  80%    736
  90%    781
  95%    857
  98%    898
  99%    938
 100%    960 (longest request)
```

CONCLUSIONES

- Las arquitecturas Modelo Vista Controlador MVC clásicas requieren en el caso de Laravel y CakePHP la compilación y modificación de la configuración del servidor apache para la completa integración de sus componentes, mientras que el mejor desempeño y flexibilidad lo tienen CodeIgniter y nuestra propuesta PMVC que requieren una configuración sobre la carpeta de trabajo y de la carpeta **/home** en casos particulares como el ruteo y abreviado de direcciones URL.

- Se cumple con implementar manualmente las rutinas de configuración e inicio de sockets UNIX nativos para una aceleración en el procesamiento de las peticiones o HTPP Requests así procesar inmediamente y de ser necesario realizar la transmisión dirigida a las aplicaciones clientes (Broadcasting), esto debido que componentes existentes como WebRTC y SocketJS duplicaría el encaminado de las peticiones. Se hace hincapié en el uso de socket nativos como base de la comunicación por la sencillez de implementación final después de una depuración realizada durante varias iteraciones y depuración luego agregar el protocolo de negociación (handshake) para la conversión y uniformización de datos y la interoperabilidad entre sockets UNIX y los WebSockets de los navegadores web al tener en cuenta que el protocolo HTTP no procesado datos binarios en ráfaga o RAWs sino su contenido se desplaza en un contenido de texto y para los datos que requieren datos binarios estos son codificados en base 64 para la trasmisión.

- La propuesta final de esta investigación propuesta de desarrollo Modelo Vista Controlador Pervasivo (PMVC), permitirá a desarrolladores terceros obtener aplicaciones que interactúen entre sus componentes adentrándonos en la masificación de aplicaciones ubicuas que no se auto actualicen e interactúan entre

los clientes y los datos centralizados, ofreciendo así una experiencia de aplicación web de tiempo real.

- ApacheBench (ab) es un aplicativo informático que se ejecuta por línea de comandos de subproceso único para medir el rendimiento de los servidores web HTTP. Originalmente diseñado para probar el servidor HTTP Apache, es lo suficientemente genérico como para probar cualquier servidor web. La herramienta ab viene incluida con la distribución de código fuente estándar de Apache y al igual que el servidor web de Apache, es un software gratuito de código abierto y se distribuye bajo los términos de la licencia de Apache, se realiza la evaluación y tiempos de respuesta de las peticiones HTTP de los distintos frameworks comparados, se obtiene como resultado un desempeño optimo y con ello se valida el framework pervasivo que proponemos en esta investigación.

.

RECOMENDACIONES

- Se recomienda la integración de una arquitectura de microservicios MSA para incrementar las capacidades de un nuevo y mejor Framework, de este modo la actualización de datos podría realizarse localmente en los clientes así al perder la conexión la información se sincronizaría con la central y los clientes afectados. Esto supone desde ya un esquema de alertas e intercambios de prioridad y evitar la pérdida de datos.

- Finalmente se recomienda el uso y masificación de esta arquitectura de desarrollo web para aplicaciones que requieran visualización en tiempo real como una forma de integración e inmersión tecnológica, teniendo que el futuro de los dispositivos es estar interconectados para la flexibilidad de acceso de los usuarios.

BIBLIOGRAFÍA

Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and Raymor, B. (2018). CoAP *(Constrained Application Protocol) over TCP, TLS, and WebSockets (No. RFC 8323)*. Universitaet Bremen TZI, Standards Track Zebra Technologies. https://tools.ietf.org/html/rfc8323

Coulouris, G., Dollimore, J., Kindberg, T. y Blair, G. (2013). *Sistemas Distribuídos - Conceitos e Projeto*. Porto Alegre: Bookman Editora.

Dandan, Z., Zhiqiang, W. & Yongquan, Y. (2013). Research on Lightweight MVC Framework Based on Spring MVC and Mybatis. *Sixth International Symposium on Computational Intelligence and Design 2013.* 350-353. IEEE. doi: https://www.doi.org/10.1109/iscid.2013.94

Dissanayake, N. R. y Dias, K. (2018). Balanced Abstract Web-MVC Style: An Abstract MVC Implementation for Web-based Applications. *GSTF Journal on Computing (JoC)*, 5(3).

Galloway, J., Haack, P., Wilson, B., & Allen, K. S. (2012). *Professional ASP. NET MVC 4*. John Wiley & Sons. ISBN: 978-1-118-34846-8

Grozev, B., Politis, G., Ivov, E., Noel, T., and Singh, V. (2017). Experimental evaluation of simulcast for WebRTC. *IEEE Communications Standards Magazine*, 1(2), 52-59.

González, Y. D. & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. *Revista Telemática*, 11(1), 47-57.

Hernández, M. T. (2015). *Symfony Framework. Desarrollo Rápido de Aplicaciones Web*. IT Campus Academy.

Hernández S., R., Fernández C, y Baptista L., P. (2010). *Metodología de la investigación* (3ra ed). México: McGraw-Hill.

Jailia, M., Kumar, A., Agarwal, M., and Sinha, I. (2016, November). Behavior of MVC (Model View Controller) based Web Application developed in PHP and. NET framework. *In 2016 International Conference on ICT in Business Industry & Government (ICTBIG)* 1-5. IEEE. doi: https://www.doi.org/10.1109/ICTBIG.2016.7892651

Johnston, A. B., y Burnett, D. C. (2012). *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC

Jurado P, L.A. (2014). *Arquitecturas de Computación Pervasiva Basadas en Servicios REST*. Universidad Politécnica de Madrid, Ingeniería de Redes y Servicios Telemáticos. España

Kordelas, A., Politis, I. & Dagiuklas, T.. (2015). Transport analysis and quality evaluation of MVC video streaming. Multimedia Tools and Applications. *ResearchGate*. doi: https://www.doi.org/10.1007/s11042-015-2530-8

Krasner, G., Stephen T. (Aug/Sep de 1988). A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. The JOT (SIGS Publications). Also published as "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System" (Report), *ParcPlace Systems*; Retrieved 2012-06-05

Krumm J., Bardram, A., Bernheim B., et. al. (2010), *Ubiquitous Computing Fundamentals*, C R C Press LLC, ISBN 978-1-4200-9360-5

Li, X., Liu, N. (2016, December). Research on L-MVC Framework. *In 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (P.D.C.A.T.)*, . 151-154. IEEE. doi: https://www.doi.org/10.1109/PDCAT.2016.043

Lu Y., Liu W., Cui H. (2018). MSA vs. MVC: Future Trends for Big Data Processing Platforms. In: Qiu M. (eds) Smart Computing and Communication. SmartCom 2017. *Lecture Notes in Computer Science, vol 10699*. Springer, Cham

Lu, C., Zeng, J., Zeng, Y., Shi, T., Zhang, Y., and Guo, Y. (2017, October). Research and design of APP for new energy vehicles electronic control system based on cloud platform. *In 2017 6th International Conference on Computer Science and Network Technology (ICCSNT)* 179-183. IEEE. doi: https://www.doi.org/10.1007/978-3-319-73830-7_31

Microsoft. (2014). ASP.NET MVC Overview. *Microsoft Developer Network*.

Pan, C. C., Lin, C. C. (2018, April). Designing and implementing a computerized adaptive testing system with an MVC framework: A case study of the IEEE floating-point standard. *In 2018 IEEE International Conference on Applied System Invention (ICASI)* 609-612. IEEE. doi: https://www.doi.org/10.1109/ICASI.2018.8394328

Powers, B., Vilk, J. and Berger, E. D. (2017). Browsix: Bridging the Gap Between Unix and the Browser. *ACM SIGOPS Operating Systems Review*, 51(2), 253-266.

Reenskaug T. (1978) MVC Pattern XEROX PARC 1978-79. http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

Sanchez, F. L., Althmann, M. F. (2014). Desenvolvimento web com ASP.NET MVC. Casa Do Código.

Selfa, D. M., Carrillo, M., and Boone, M. D. R. (2006, February). A database and web application based on MVC architecture. *In 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06)*. IEEE. doi: https://www.doi.org/10.1109/CONIELECOMP.2006.6

Servicio de Informática ASP.NET MVC 3. (2019). *Modelo Vista Controlador*. Recuperado de https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Sinha, S. (2017). *A CRUD Application. In Beginning Laravel*. Apress, Berkeley, CA. 67-79

Singh, A., Chawla, P., Singh, K., and Singh, A. K. (2018, May). Formulating an MVC Framework for Web Development in JAVA. *In 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)* 926-929. IEEE.

Shao, Y., Ott, J., Jia, Y. J., Qian, Z., and Mao, Z. M. (2016, October). The misuse of android unix domain sockets and security implications. *In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 80-91. ACM.

Stroustrup, B. (2013). C++ *Programming Language*. Addison-Wesley. ISBN 0-321-56384-0

Wei, C., Lin, H., LiJing, L., and Jing L. (2009). The Research of PHP Development Framework Based on MVC Pattern. 2009, *Fourth International Conference on Computer Sciences and Convergence Information Technology*. ICCIT 2009.

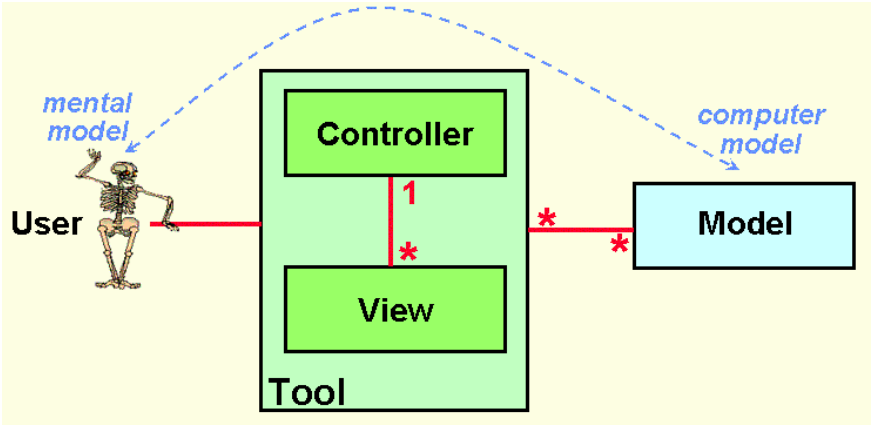
Xu, S., Yang, T. (2011, July). Application of Struts framework based on MVC in Online Countryside Teachers' Training System in China. *In 2011 International Conference on Multimedia Technology*. 6252-6255. IEEE.

Yue, J., Ye, Y., Wei, Z., and Li, Z. (2016, December). The design and implementation of national traditional sports professional teaching resources platform based on MVC. *In 2016 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS).* 277-280. IEEE. doi: https://www.doi.org/10.1109/ICITBS.2016.111

Zhang, S. & Liu, Z. (2017, January). Research on the construction and robustness testing of SaaS cloud computing data center based on the MVC design pattern. *In 2017 International Conference on Inventive Systems and Control* (ICISC). 1-4. IEEE. doi: https://www.doi.org/10.1109/ICISC.2017.8068723

ANEXOS

Anexo 1. Esquema nativo MVC 1978-79



A note on DynaBook requirements

22 March 1979 (Partial scan, 11 pp

http://folk.uio.no/trygver/1979/sysreq/SysReq.pdf

MVC was conceived as a general solution to the problem of users controlling a large and complex data set. The hardest part was to hit upon good names for the different architectural components. Model-View-Editor was the first set:

Thing-Model-View-Editor

12 May 1979 (11 pp)

PDF (11 pp, 312,594 bytes)

After long discussions, particularly with Adele Goldberg, we ended with the terms Model-View-Controller:

Models-Views-Controllers

10 December 1979 (2 pp)

(.PDF (2 pp, 9,696 bytes)

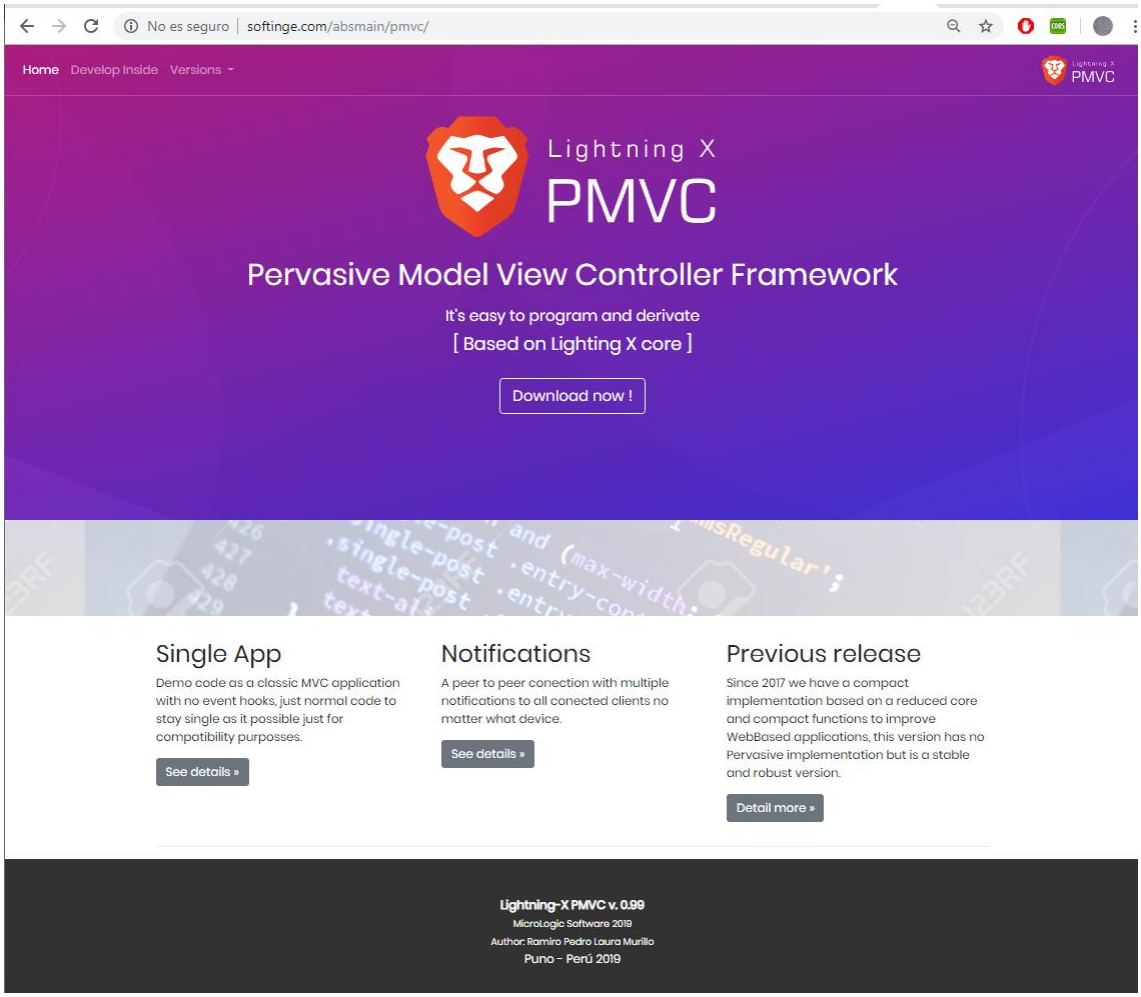
The two papers have been copied together here:

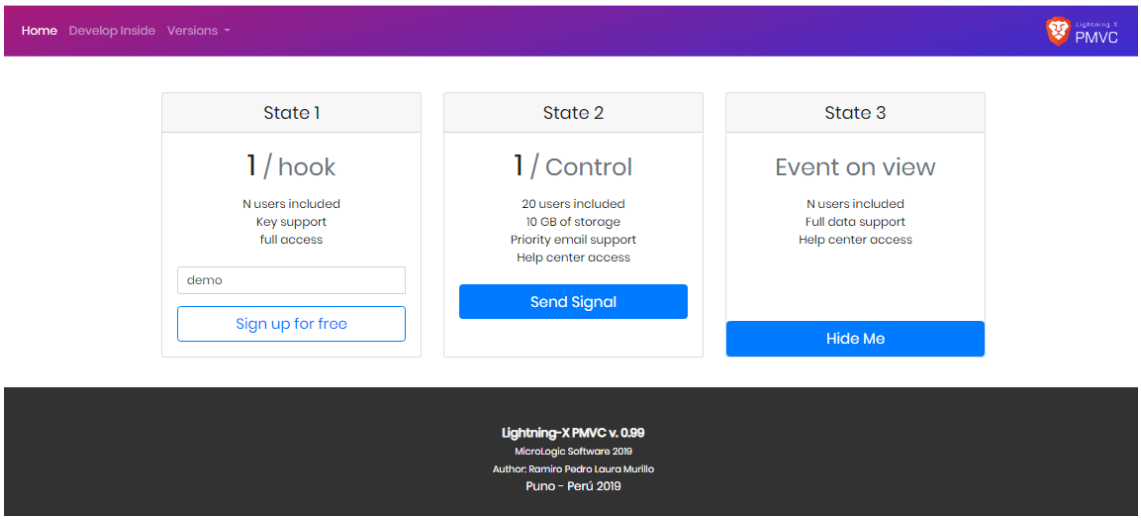
(PDF, 396 567 bytes)

https://codeday.me/es/qa/20190425/567148

Anexo 2. Portal web demostrativo del Framework

http://softinge.com/absmain/pmvc/





Anexo 3. Listado de código fuente del servidor Socket.

```php
<?php
//---------------------------------------------------------------------
//  This Redefined from originally code in: WebSocketServer PHP
//  downloaded from https://github.com/WebSocketServer
//---------------------------------------------------------------------
//
//  :: Lightning-X MVC - 1/16
//
//  Author: Ramiro Pedro Laura Murillo
//          Doctoral Candidate in Computer Science
//          Universidad Nacional del Altiplano - Puno
//
//---------------------------------------------------------------------

class SocketUser
{
    public $socket;
    public $id;
    public $headers    = array();
    public $handshake = false;
    public $handPartialPacket = false;
    public $sendingContinuous = false;
    public $hasSentClose      = false;
    public $partialBuffer     = "";
    public $partialMessage    = "";

    function __construct($id, $socket)
    {
        $this->id = $id;
        $this->socket = $socket;
    }
}

abstract class WebSocketServ
{
    protected $userClass = 'SocketUser';
    protected $maxBufferSize;
    protected $master;
    protected $sockets     = array();
    protected $users       = array();
    protected $interactive = true;

    function __construct($addr, $port, $bufferLength = 2048)
    {
        $this->maxBufferSize = $bufferLength;

        $this->master = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)
                        or die("Failed: socket_create()");
        socket_set_option($this->master, SOL_SOCKET, SO_REUSEADDR, 1)
                        or die("Failed: socket_option()");
        socket_bind($this->master, $addr, $port)
                        or die("Failed: socket_bind()");
        socket_listen($this->master,20)
                        or die("Failed: socket_listen()");

```

```
58          $this->sockets['m'] = $this->master;
59          $this->stdout("Server started Listening on $addr:$port".$this-
60  >master);
61      }
62
63      abstract protected function onReceive($user,$message); //
64      abstract protected function connected($user);          //
65      abstract protected function closed($user);             //
66
67      protected function connecting($user)
68      {
69          // Override, after User is created, but before handshake ends.
70      }
71
72      protected function send($user, $message)
73      {
74          if ($user->handshake) {
75              $message = $this->frame($message,$user);
76              $result = @socket_write($user->socket,
77                                      $message, strlen($message));
78          }
79          else {
80              // User has not yet performed their handshake.
81              $holdingMessage = array(
82                      'user' => $user, 'message' => $message);
83              $this->heldMessages[] = $holdingMessage;
84          }
85      }
86
87      protected function broadcast( $message )
88      {
89          foreach ($this->users as $user) {
90              $this->send($user, $message);
91          }
92      }
93
94      protected function connect( $socket )
95      {
96          $user = new $this->userClass(uniqid('u'), $socket);
97          $this->users[$user->id] = $user;
98          $this->sockets[$user->id] = $socket;
99          $this->connecting($user);
100     }
101
102     protected function disconnect( $socket,
103                     $triggerClosed = true, $sockErrNo = null)
104     {
105         $disconnectedUser = $this->getUserBySocket($socket);
106
107         if ($disconnectedUser !== null) {
108             unset($this->users[$disconnectedUser->id]);
109
110             if (array_key_exists( $disconnectedUser->id,
111                                   $this->sockets) ) {
112                 unset($this->sockets[$disconnectedUser->id]);
113             }
114
115             if (!is_null($sockErrNo)) {
116                 socket_clear_error($socket);
117
118
```

```
119             }
120             if ($triggerClosed) {
121                 $this->stdout( "Client disconnected.
122 ".$disconnectedUser->socket );
123                 $this->closed( $disconnectedUser );
124                 socket_close( $disconnectedUser->socket );
125             }
126             else {
127                 $message = $this->frame('', $disconnectedUser,
128 'close');
129                 @socket_write( $disconnectedUser->socket,
130                     $message, strlen($message));
131             }
132         }
133     }
134
135
136 /**********************************************************************
137  *
138  *  Main processing loop
139  *
140 **********************************************************************/
141     public function run()
142     {
143         while( true )
144         {
145             if (empty($this->sockets)) {
146                 $this->sockets['m'] = $this->master;
147             }
148
149             $read  = $this->sockets;
150             $write = $except = null;
151
152             @socket_select( $read, $write, $except, 1 );
153
154             foreach ($read as $socket) {
155                 if ($socket == $this->master) {
156                     $client = socket_accept($socket);
157                     if ($client < 0) {
158                         $this->stderr("Failed: socket_accept()");
159                         continue;
160                     }
161                     else {
162                         $this->connect($client);
163                         $this->stdout("Client connected. " . $client);
164                     }
165                 }
166                 else {
167                     $numBytes = @socket_recv($socket, $buffer, $this-
168 >maxBufferSize, 0);
169                     if ($numBytes === false) {
170                         $sockErrNo = socket_last_error($socket);
171                         switch ($sockErrNo)
172                         {
173                             case 102: // ENETRESET    -- Network
174                             case 103: // ECONNABORTED -- Software
175                             case 104: // ECONNRESET   -- Connection
176                             case 108: // ESHUTDOWN    -- Cannot send
177                             case 110: // ETIMEDOUT    -- Connection
178
179
```

```
                                case 111: // ECONNREFUSED -- Connection
                                case 112: // EHOSTDOWN    -- Host is down
                                case 113: // EHOSTUNREACH -- No route to
                                case 121: // EREMOTEIO    -- Rempte I/O
                                case 125: // ECANCELED    -- Operation

                                $this->stderr("Unusual disconnect on
socket " . $socket);
                                $this->disconnect($socket, true,
$sockErrNo);
                                break;
                                default:
                                $this->stderr('Socket error: ' .
socket_strerror($sockErrNo));
                            }

                        }
                        elseif ($numBytes == 0) {
                            $this->disconnect($socket);
                            $this->stderr("Client disconnected. TCP
connection lost: " . $socket);
                        }
                        else {
                            $user = $this->getUserBySocket($socket);
                            if (!$user->handshake) {
                                $tmp = str_replace("\r", '', $buffer);
                                if (strpos($tmp, "\n\n") === false ) {
                                    continue;
                                    // If the client has not finished
sending the header, then wait before sending our upgrade response.
                                }
                                $this->doHandshake($user,$buffer);
                            }
                            else {

                                // send and split packet into frame and
send it to deframe
                                //
                                $this->split_packet($numBytes,$buffer,
$user);
                            }
                        }
                    }
                }
            }
        }

        protected function doHandshake($user, $buffer)
        {
            $magicGUID = "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";
            $headers   = array();
            $lines     = explode("\n",$buffer);

            foreach ($lines as $line) {
                if (strpos($line,":") !== false) {
                    $header = explode(":",$line,2);
                    $headers[strtolower(trim($header[0]))] =
trim($header[1]);
                }
```

```
241             elseif (stripos($line,"get ") !== false) {
242                 preg_match("/GET (.*) HTTP/i", $buffer, $reqResource);
243                 $headers['get'] = trim($reqResource[1]);
244             }
245         }
246 
247         if (isset($headers['get'])) {
248             $user->requestedResource = $headers['get'];
249         }
250         else {
251             // todo: fail the connection
252             $handshakeResponse = "HTTP/1.1 405 Method Not
253 Allowed\r\n\r\n";
254         }
255 
256 
257         // when error happends
258         if ( isset($handshakeResponse) ) {
259             socket_write($user-
260 >socket,$handshakeResponse,strlen($handshakeResponse));
261             $this->disconnect($user->socket);
262             return;
263         }
264 
265         $user->headers = $headers;
266         $user->handshake = $buffer;
267 
268         $rawToken = "";
269         $webSocketKeyHash = sha1($headers['sec-websocket-key'] .
270 $magicGUID);
271 
272         for ($i = 0; $i < 20; $i++) {
273             $rawToken .= chr( hexdec(substr($webSocketKeyHash,$i*2,
274 2)) );
275         }
276 
277 
278         $handshakeToken = base64_encode($rawToken) . "\r\n";
279         $subProtocol    = (isset($headers['sec-websocket-protocol']))
280 ? $this->processProtocol($headers['sec-websocket-protocol']) : "";
281         $extensions     = (isset($headers['sec-websocket-
282 extensions'])) ? $this->processExtensions($headers['sec-websocket-
283 extensions']) : "";
284 
285         $handshakeResponse = "HTTP/1.1 101 Switching
286 Protocols\r\nUpgrade: websocket\r\nConnection: Upgrade\r\nSec-
287 WebSocket-Accept: $handshakeToken$subProtocol$extensions\r\n";
288         socket_write( $user->socket, $handshakeResponse,
289 strlen($handshakeResponse) );
290         $this->connected($user);
291 
292         //echo "HandResp: $handshakeResponse";
293     }
294 
295     protected function getUserBySocket($socket)
296     {
297         foreach ($this->users as $user) {
298             if ($user->socket == $socket) {
299                 return $user;
300 
301 
```

```
302                 }
303             }
304             return null;
305         }
306 
307         public function stdout($message)
308         {
309             if ($this->interactive) {
310                 echo "$message\n";
311             }
312         }
313 
314         public function stderr($message)
315         {
316             if ($this->interactive) {
317                 echo "$message\n";
318             }
319         }
320 
321         protected function frame($message, $user, $messageType='text',
322 $messageContinues=false)
323         {
324             switch ($messageType) {
325                 case 'continuous':
326                 $b1 = 0;
327                 break;
328                 case 'text':
329                 $b1 = ($user->sendingContinuous) ? 0 : 1;
330                 break;
331                 case 'binary':
332                 $b1 = ($user->sendingContinuous) ? 0 : 2;
333                 break;
334                 case 'close':
335                 $b1 = 8;
336                 break;
337                 case 'ping':
338                 $b1 = 9;
339                 break;
340                 case 'pong':
341                 $b1 = 10;
342                 break;
343             }
344             if ($messageContinues) {
345                 $user->sendingContinuous = true;
346             }
347             else {
348                 $b1 += 128;
349                 $user->sendingContinuous = false;
350             }
351             $length = strlen($message);
352             $lengthField = "";
353             if ($length < 126) {
354                 $b2 = $length;
355             }
356             elseif ($length < 65536) {
357                 $b2 = 126;
358                 $hexLength = dechex($length);
359                 //$this->stdout("Hex Length: $hexLength");
360                 if (strlen($hexLength)%2 == 1) {
361 
362 
```

```
363                 $hexLength = '0' . $hexLength;
364             }
365             $n = strlen($hexLength) - 2;
366             for ($i = $n; $i >= 0; $i=$i-2) {
367                 $lengthField = chr(hexdec(substr($hexLength, $i, 2)))
368 . $lengthField;
369             }
370             while (strlen($lengthField) < 2) {
371                 $lengthField = chr(0) . $lengthField;
372             }
373         }
374         else {
375             $b2 = 127;
376             $hexLength = dechex($length);
377             if (strlen($hexLength)%2 == 1) {
378                 $hexLength = '0' . $hexLength;
379             }
380             $n = strlen($hexLength) - 2;
381             for ($i = $n; $i >= 0; $i=$i-2) {
382                 $lengthField = chr(hexdec(substr($hexLength, $i, 2)))
383 . $lengthField;
384             }
385             while (strlen($lengthField) < 8) {
386                 $lengthField = chr(0) . $lengthField;
387             }
388         }
389         return chr($b1) . chr($b2) . $lengthField . $message;
390     }
391 
392     //check packet if he have more than one frame and process each
393 frame individually
394     protected function split_packet($length,$packet, $user)
395     {
396         //add PartialPacket and calculate the new $length
397         if ($user->handPartialPacket) {
398             $user->handPartialPacket = false;
399             $packet = $user->partialBuffer . $packet;
400             $length = strlen($packet);
401         }
402 
403         $fullpacket = $packet;
404         $frame_pos = 0;
405         $frame_id  = 1;
406 
407         while($frame_pos<$length)
408         {
409             $headers    = $this->extractHeaders( $packet );
410             $size       = $this->calcoffset( $headers );
411             $framesize = $headers['length'] + $size;
412 
413             //split frame from packet and process it
414             $frame = substr($fullpacket,$frame_pos,$framesize);
415             if (($message = $this->deframe($frame, $user,$headers))
416 !== FALSE) {
417                 if ($user->hasSentClose) {
418                     $this->disconnect($user->socket);
419                 } else {
420                     if ((preg_match('//u', $message)) ||
421 ($headers['opcode']==2)) {
422 
423 
```

```
                        //$this->stdout("Text msg encoded UTF-8 or
Binary msg\n".$message);
                        $this->onReceive($user, $message);
                    } else {
                        $this->stderr("not UTF-8\n");
                    }
                }
            }

            $frame_pos += $framesize;
            $packet = substr($fullpacket,$frame_pos);
            $frame_id++;
        }
    }

    protected function calcoffset($headers)
    {
        $offset = 2;
        if ($headers['hasmask']) {
            $offset += 4;
        }
        if ($headers['length'] > 65535) {
            $offset += 8;
        } elseif ($headers['length'] > 125) {
            $offset += 2;
        }
        return $offset;
    }

    protected function deframe($message, &$user)
    {
        $headers = $this->extractHeaders($message);
        $pongReply = false;
        $willClose = false;
        switch($headers['opcode']) {
            case 0:
            case 1:
            case 2:
            break;
            case 8:
            // todo: close the connection
            $user->hasSentClose = true;
            return "";
            case 9:
            $pongReply = true;
            case 10:
            break;
            default:
            //$this->disconnect($user); // todo: fail connection
            $willClose = true;
            break;
        }


        if ($this->checkRSVBits($headers,$user)) {
            return false;
        }
        if ($willClose) {
            // todo: fail the connection
```

```
485             return false;
486         }
487         $payload = $user->partialMessage . $this-
488 >extractPayload($message,$headers);
489         if ($pongReply) {
490             $reply = $this->frame($payload,$user,'pong');
491             socket_write($user->socket,$reply,strlen($reply));
492             return false;
493         }
494         if ($headers['length'] > strlen($this-
495 >applyMask($headers,$payload))) {
496             $user->handPartialPacket = true;
497             $user->partialBuffer = $message;
498             return false;
499         }
500         $payload = $this->applyMask($headers,$payload);
501         if ($headers['fin']) {
502             $user->partialMessage = "";
503             return $payload;
504         }
505         $user->partialMessage = $payload;
506 
507         return false;
508     }
509 
510     protected function extractHeaders($message)
511     {
512         $header = array('fin'     => $message[0] & chr(128),
513                         'rsv1'    => $message[0] & chr(64),
514                         'rsv2'    => $message[0] & chr(32),
515                         'rsv3'    => $message[0] & chr(16),
516                         'opcode'  => ord($message[0]) & 15,
517                         'hasmask' => $message[1] & chr(128),
518                         'length'  => 0,
519                         'mask'    => "");
520 
521         $header['length'] = (ord($message[1]) >= 128) ?
522 ord($message[1]) - 128 : ord($message[1]);
523         if ($header['length'] == 126) {
524             if ($header['hasmask']) {
525                 $header['mask'] = $message[4] . $message[5] .
526 $message[6] . $message[7];
527             }
528             $header['length'] = ord($message[2]) * 256 +
529 ord($message[3]);
530         }
531         elseif ($header['length'] == 127) {
532             if ($header['hasmask']) {
533                 $header['mask'] = $message[10] . $message[11] .
534 $message[12] . $message[13];
535             }
536             $header['length'] = ord($message[2]) * 65536 * 65536 *
537 65536 * 256
538                 + ord($message[3]) * 65536 * 65536 * 65536
539                 + ord($message[4]) * 65536 * 65536 * 256
540                 + ord($message[5]) * 65536 * 65536
541                 + ord($message[6]) * 65536 * 256
542                 + ord($message[7]) * 65536
543                 + ord($message[8]) * 256
544 
545 
```

```
546                 + ord($message[9]);
547         }
548         elseif ($header['hasmask']) {
549             $header['mask'] = $message[2] . $message[3] . $message[4]
550 . $message[5];
551         }
552         //echo $this->strtohex($message);
553         //$this->printHeaders($header);
554         return $header;
555     }
556 
557     protected function checkRSVBits($headers,$user)
558     {
559         if (ord($headers['rsv1']) + ord($headers['rsv2']) +
560 ord($headers['rsv3']) > 0) {
561             return true;
562         }
563         return false;
564     }
565 
566     protected function extractPayload($message,$headers) {
567         $offset = 2;
568         if ($headers['hasmask']) {
569             $offset += 4;
570         }
571         if ($headers['length'] > 65535) {
572             $offset += 8;
573         }
574         elseif ($headers['length'] > 125) {
575             $offset += 2;
576         }
577         return substr($message,$offset);
578     }
579 
580     protected function applyMask($headers,$payload) {
581         $effectiveMask = "";
582         if ($headers['hasmask']) {
583             $mask = $headers['mask'];
584         }
585         else {
586             return $payload;
587         }
588         while (strlen($effectiveMask) < strlen($payload)) {
589             $effectiveMask .= $mask;
590         }
591         while (strlen($effectiveMask) > strlen($payload)) {
592             $effectiveMask = substr($effectiveMask,0,-1);
593         }
594         return $effectiveMask ^ $payload;
595     }
596 
597     protected function checkHost($hostName)
598     {
599         return true; // Override and return false if the
600     }
601     protected function checkOrigin($origin) {
602         return true; // Override and return false if the origin
603     }
604     protected function checkWebsocProtocol($protocol) {
605 
606 
```

```
607             return true; // Override and return false if a protocol
608         }
609         protected function checkWebsocExtensions($extensions) {
610             return true; // Override and return false if an extension
611         }
612         protected function processProtocol($protocol) {
613             return ""; // return either "Sec-WebSocket-Protocol:
614         }
615         protected function processExtensions($extensions) {
616             return ""; // return either "Sec-WebSocket-Extensions:
617         }
618     }
619
620
621     $creds = "=======================================================\n"
622            . "== Lightning-X MVC  [loboAlfa]  SocketServer v0.9     ==\n"
623            . "=======================================================\n"
624            ;
625
626     class MyServer extends WebSocketServ
627     {
628         protected function onReceive ($usrsock, $message)
629         {
630             // $this->send($usrsock,$message);
631             $this->broadcast( $message );
632             $this->stdout( "* $usrsock->id :: $message" );
633         }
634
635         protected function connected ( $usrsock )
636         {
637             // Do nothing: This is where cleanup would go
638         }
639
640         protected function closed ( $usrsock )
641         {
642             // Do nothing: This is where cleanup would go
643         }
644     }
645
646     //---------------------------------------------------------------
647     // starts Listen server
648     //---------------------------------------------------------------
649     echo $creds;
650
651     $app = new MyServer("209.126.105.180","1020");
652     $app->run();
653
654
655     //- EOF
656
657
658
659
660
661
662
```

Anexo 4. Listado de código fuente del prototipo de event dispatcher.

```
//---------------------------------------------------------------------
//
// Socket Dipatcher for Pervasive MVC Framework 2019
//
// Coded by: Ramiro Pedro Laura Murillo
// version: prototype 0.001
//
//---------------------------------------------------------------------

var socket;

function createSock( msg, fnHook )
{
    socket = new WebSocket('ws://vriunap.pe:1020');

    socket.onopen = function (){
        // send when it connects
        socket.send( msg );
    };

    socket.onmessage = function (e){

        if( ! isJSON(e.data) ){
            console.log("Error en datos");
            return;
        }

        obj = JSON.parse( e.data );
        console.log( obj );

        fnHook( obj );
    };

    socket.onclose = function (error){
        console.log( "onClose: " + error );
    };

    socket.onerror = function (error){
        // not answer when try connect
        console.log('WebSocket Error: ' + error);
    };
}

function signalSend( txt, fnHook )
{
    if( !txt ) txt ="";

    var msg = { req : { msg: txt, type:"event" } };

    if( socket == null )
        createSock( JSON.stringify(msg), fnHook );


```

```
    if( socket.readyState == 3 )
        createSock( JSON.stringify(msg), fnHook );

    if( socket.readyState == 1 )
        socket.send( JSON.stringify(msg), fnHook );
}


function lxStateOne()
{
    var strval = jLx("#lxEdit").val();

    signalSend( strval, function( obj ){

        if( obj.req.msg == "evEfect" )
            lxStateThree();
        else
            jLx("#lxEdit").val( obj.req.msg );
    } );

} lxStateOne();

//-----------------------------------------------------------------
function lxStateNotif()
{
    signalSend( "evEfect" );
}
//-----------------------------------------------------------------
function lxStateThree()
{
    if ( $( "#bmenu" ).is( ":hidden" ) )
        $( "#bmenu" ).show( "slow" );
    else
        $( "#bmenu" ).slideUp();

    if ( $( "#cmenu" ).is( ":hidden" ) )
        $( "#cmenu" ).show( "slow" );
    else
        $( "#cmenu" ).slideUp();
} lxStateThree();
```