UNIVERSIDAD DE NAVOJOA
Ingeniería en Sistemas Computacionales

RAQUISFW: FRAMEWORK DE DESARROLLO PARA EL DEPARTAMENTO DE SISTEMAS Y TECNOLOGÍA DE LA UNIVERSIDAD DE NAVOJOA.

Proyecto presentado como requisito para obtener el título de
Ingeniero en Sistemas Computacionales

Por
Daniel Lozano Carrillo
Mayo de 2016

RAQUISFW: FRAMEWORK DE DESARROLLO PARA EL DEPARTAMENTO DE SISTEMAS Y TECNOLOGÍA DE LA UNIVERSIDAD DE NAVOJOA.

Proyecto

Presentado en cumplimiento total de los requisitos
para obtener el título de
Ingeniero en Sistemas Computacionales

APROBADO POR LA COMISIÓN

_____

Presidente: Mtro. Misael Martínez Guadarrama

_____

Miembro: Ing. Amado Víctor Fragoso Vázquez

_____

Miembro: Ing. Gustavo Ordoñez

_____

Fecha de Aprobación

RESUMEN

UNIVERSIDAD DE NAVOJOA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

TITULO: RAQUISFW: FRAMEWORK DE DESARROLLO PARA EL DEPARTAMENTO DE SISTEMAS Y TECNOLOGÍA DE LA UNIVERSIDAD DE NAVOJOA.

Nombre del investigador: Daniel Lozano Carrillo

Asesor principal: Mtro. Misael Martínez Guadarrama

Fecha de terminación: Mayo 2016

Problema

Desarrollar un Framework para el departamento de Sistemas y Tecnología de la Universidad de Navojoa en el lenguaje PHP, para mejorar la calidad de los sistemas de desarrollo.

Método

Para la elaboración del framework se analizaron los problemas más comunes en el desarrollo de software, de los cueles se obtuvieron requerimientos. Con estos requerimientos y siguiendo la metodología ICONIX, se elaboran casos de uso, para posteriormente convertirse en código fuente. Para garantizar la calidad de software se estudian y analizan arquitecturas de software, seguridad, convenciones y mejores prácticas usadas en un ámbito profesional. Al realizar este estudio se opta por la implementación de la arquitectura MVC. El código fuente junto con el historial de desarrollo se puede encontrar en GitHub, el proyecto es de código abierto y es posible participar en su continuo desarrollo.

Conclusiones

Al concluir la primera etapa de desarrollo del framework este cumple todos los requisitos especificados en los requerimientos con sus respectivos casos de uso, y es de importancia mencionar que este framework tiene un desarrollo continuo, su implementación siempre tendrá una mejora continua, conforme se esté usando, este es uno de los propósitos principales del framework.

DEDICATORIA

A DIOS
Por permitirme alcanzar una meta más en vida.

A MI PAPA
Por hacer el más grande sacrifico para lograr que terminara mi carrera, por creer en mí y apoyarme incondicionalmente.

A MI MAMA
Por desvelarse cada noche conmigo, por velar para siempre estuviera bien, por todas tus oraciones.

A MIS COMPAÑEROS
Jorge, Daniel, Hogilber, Guendoly, Sergio y Elida.

AGRADECIMIENTOS

Alejandro Gámez: Nunca podre pagar los consejos que me llego a dar, gracias por ser mi amigo y un gran maestro. Gracias por compartir toda su sabiduría con migo.

Daniel Carrada: Brother, todo hubiera sido diferente sin ti, nunca existiría Danny^2. Siempre me apoyaste y siempre me cuidaste, gracias bro.

Fernando López: Posiblemente el Ing. que más clases me dio, le doy las gracias por la gran paciencia que siempre tuvo conmigo, y por todos sus consejos.

Gustavo Ordoñez: Gracias por ser mi amigo, por retarme a siempre ser mejor, por ayudarme a ver las cosas diferentes, por creer en mí, por todos sus consejos.

Javier Hernández: Mi primer jefe, yo siempre he sabido que puedo contar con su apoyo, y le doy las gracias por eso y por todo lo que hiso por mí.

Jorge Toloza: Tolozin, mi gran amigo, no tienes idea del aprecio y el respeto que te tengo. Gracias por siempre ser un buen amigo y por el apoyo que siempre me has dado.

Lizeth Torres: Aun recuerdo como si fuera ayer el día que nos dijo que se marchaba un día muy triste. Gracias por sus consejos, por preocuparse por mí, por ser una excelencia como maestra y como persona, porque usted no dejaba de ser maestra al salir del aula de clases.

Maria Luisa Monarrez: La mejor maestra de Matemáticas en la historia, nunca olvidare la forma en que creía en mí y me defendía a pesar de que me dormía en su clase. Gracias por todo el tiempo que dedico a aconsejarme, soy un mejor ingeniero por sus consejos.

Misael Martínez: Podría hacer mi tesis tan solo de agradecimientos a usted, pero gracias por todo su apoyo, por abrirme las puertas de su casa cuando lo necesite, por las tantas veces que me extendió la mano, por darme su amistad.

Silvia Cristel Gutiérrez: Kristelitha, siempre te preocupaste por mí, siempre viste que nada me faltara. Gracias por tu comprensión, por tu cariño, por tu paciencia, por empujarme a ser mejor, por nunca dejarme rendir. Por hacerme sonreír, por cuidar mi salud, todo lo que hiciste por mí, por siempre lo llevare en mi corazón.

Victor Fragoso: El papa Fragoso, gracias por el apoyo, la confianza y por siempre creer en mí. De usted aprendí muchas cosas y no solamente de ingeniera.

Zabdi Alvarado: Gracias por su amistad Ing, por su paciencia, por apoyarme y escucharme, por confiar en mí.

Todos: A todas las personas que a lo largo de mi carrera me apoyaron, me escucharon y creyeron en mí, aquellas personas que dedicaron algo de tiempo para conocerme, a todos los maestros que aportaron de su conocimiento para mi formación, a las personas que me brindaron su amistad, a las personas que me aconsejaron, a las personas que oraron por mí y a todas las personas que me ayudaron a convertirme en lo que ahora soy.

Gracias.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO I

INTRODUCCIÓN Y DECLARACIÓN DEL PROBLEMA

Introducción

El objetivo de este proyecto es la elaboración de un conjunto de herramientas y utilerías necesarias para agilizar, facilitar, y economizar el desarrollo de software, conocido como framework. La mayor parte del código que constituye un proyecto de software es código reutilizable, es decir código que funciona en un sistema completamente diferente, muchos desarrolladores lo ignoran o es desconocido para ellos.

El uso de un framework no es algo absolutamente necesario, pero es una herramienta que puede ayudar a un desarrollo mejor y rápido, mejor porque cumple con calidad empresarial y seguridad de un buen desarrollo, rápido porque permite enfocarse únicamente en la mecánica de la aplicación y no en módulos generales y reutilizables ayudo a que el software tenga una buena estructura en su desarrollo, pueda ser mantenible y actualizable.

RaquisFW la presente investigación se propone que por medio de un framework el desarrollo de software sea uniforme y de calidad, permitiendo que el trabajo en equipo sea mucho más sencillo, y ordenado también mejora el desarrollo, mantenimiento, y actualización de aplicaciones para el Departamento Tecnología y Sistemas de la Universidad de Navojoa (UNAV).

Antecedentes

Los sistemas informáticos que existen en la UNAV están a cargo del departamento de tecnología y sistemas el cual da mantenimiento a dos sistemas principales, el Sistema Académico y Financiero. Existe la necesidad de nuevos módulos necesarios para el funcionamiento de distintos departamentos, los cuales no son desarrollados debido a que los sistemas utilizados son desarrollados fura de la UNAV por terceros y no se cuenta con documentación necesaria para integrar nuevo código a los sistemas fácilmente.

La situación presente obliga a generar sistemas por separado, desarrollados por estudiantes, programados en el lenguaje de programación PHP, extrayendo datos de los sistemas principales. Se propone un framework para facilitar el desarrollo de estas aplicaciones. HostDime (2014) menciona que el desarrollo es un camino largo y complejo, el cual requiere tiempo. Sin embargo aclara que la utilización de un Framework ayuda a desarrollar proyectos con mayor rapidez, reutilizando componentes y módulos genéricos y trabajando bajo una base estructural unificada.

Al eliminar el tiempo y esfuerzo para construcción de componentes genéricos, es posible que el desarrollador dedique más tiempo a las tareas y funciones específicas, y centrarse en el código de alta calidad.

Definición del Problema

El departamento de sistemas y tecnología no cuenta con un estándar o herramienta global para estandarizar el desarrollo de software, al no contar con un estándar es causa fácil la pérdida de calidad en el código fuente del producto final. Para suplir la necesidad

del desarrollo de sistemas informáticos para diferentes procesos de la Universidad de Navojoa.

Los resultados de un proyecto sin los parámetros adecuados puede terminar en fracaso, difícil o costoso mantener. La UNAV cuenta con diversas aplicaciones: promoción, calculadora de costos, búsqueda de libros y catálogo de tesis, dichas aplicaciones programadas en el lenguaje PHP, cada uno de ellos con diferente estructura, estilo, modelo y versión de PHP. RaquisFw, propone resolver el problema guiando a los desarrolladores a un mejor desempeño en la estructura del desarrollo de software.

Propósito

Desarrollar un framework que mejore el desarrollo de software, mantenimiento y actualización de aplicaciones para el departamento de sistemas y tecnología de la Universidad de Navojoa, bajo el nombre de RaquisFW.

Objetivos

Se propone por medio de RaquisFw los siguientes objetivos:

1. Agilizar el desarrollo de software mediante el uso de herramientas y librerías de uso común.
2. Tener calidad en el software proponiendo mejores prácticas y estilos de programación para el de desarrollo.
3. Facilitar la actualización y mantenimiento del software.

4. Proveer la habilidad de agregar módulos o librerías alojando el proyecto en GitHub y haciéndolo Open Source.

Justificación

La importancia de mejorar el proceso de desarrollo de software en el departamento de Sistemas y Tecnología en la Universidad de Navojoa es fundamental. Por lo tanto RaquisFW aportara librerías de uso común, una estructura común de desarrollo, seguridad, un patrón de desarrollo robusto; permitiendo al desarrollador escribir menos líneas de código, y facilitando el desarrollo en equipo. Es importante mencionar la experiencia de aprendizaje que el desarrollo de RaquisFW facilita los complejos temas de seguridad, arquitectura de software y mejores prácticas en el desarrollo. Poncier G. (s.f) menciona tres ventajas de utilizar un framework. Primero un framework simplifica el desarrollo de aplicaciones utilizando patrones para resolver tareas comunes, segundo el framework proporciona estructura al código fuente, forzando al desarrollador a crear código, leíble y fácil de mantener, por ultimo facilita la programación de aplicaciones encapsulando operaciones complejas en instrucciones sencillas.

Limitaciones

El uso de las tecnologías adecuadas para el desarrollo del framework por los múltiples cambios de versiones y lenguajes de programación.

Delimitaciones

RaquisFW será usado en el departamento de Sistemas y Tecnología de la Universidad de Navojoa, pero su código será de tipo abierto, es decir podrá ser usando y modificado libremente por cualquier persona.

Metodología del Proyecto

La metodología a utilizar en el desarrollo del framework es ICONIX (Rosenberg y Scoot, 2005) y consiste en cuatro etapas, Análisis de requisitos, Análisis y diseño preliminar, Diseño e Implementación ver figura 1

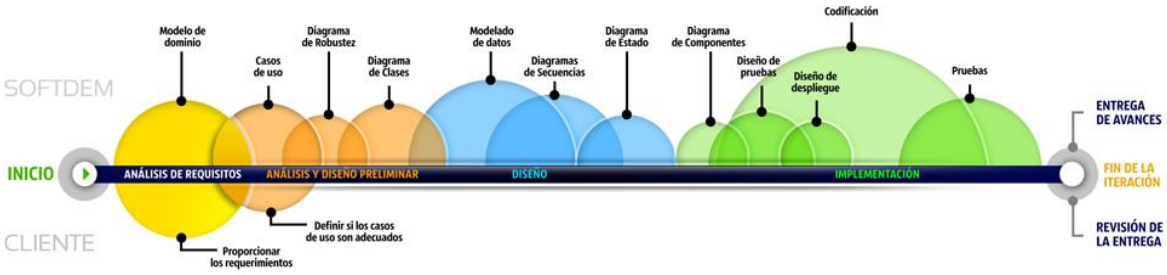


Figura 1: Tareas de metodología ICONIX

Estructura del Proyecto

La estructura del proyecto está formado por nueve capítulos los cuales son descritos a continuación:

- **Capítulo 1** Introducción. Contiene los antecedentes, la declaración del problema, la justificación, objetivo general, objetivos, delimitaciones, limitaciones y supuestos.
- **Capítulo 2** Marco teórico. Se realiza una investigación de la literatura de los temas fundamentales.
- **Capítulo 3** Metodología del proyecto. Describe la metodología a usar para el desarrollo del proyecto.

- **Capítulo 4** Análisis de requerimientos. Describe el proceso de obtención de los requerimientos del sistema.
- **Capítulo 5** Análisis y diseño preliminar**.** Describe las primeras dos etapas de ICONIX análisis de requisitos y diseño preliminar
- **Capítulo 6** Diseño. La etapa de diseño de RaquisFW, describe como se cubren todos los requerimientos
- **Capítulo 7** Implementación, describe la implantación y trabajo final.
- **Capítulo 8** Convenciones y Buenas Prácticas. Describe las buenas prácticas y convenciones propuestas para el uso de RaquisFW.
- **Capítulo 9** Conclusiones y Trabajos Futuros
- **Anexos**

Definición de Términos

API: Application Program Interface es conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. (Carrero, 2011)

Framework: La palabra inglesa "framework" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. (Riehle, 2000)

GitHub:una página para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. (GitHub, 2014)

Open Source: Software de código abierto es software que se puede utilizar libremente, cambiado y se comparte por cualquier persona. Software de código abierto es hecho por muchas personas, y se distribuye bajo licencias que cumplen con la definición de Open Source. (Opensource.org, 2014)

PHP: Lenguaje de scripting embebido en HTML. Gran parte de su sintaxis es tomada de C, Java y Perl con un par de características únicas específicas de PHP. El objetivo del lenguaje es permitir a desarrolladores web escribir rápidamente páginas generadas dinámicamente. (The PHP Group, 2014)

Modulo: Un módulo es una pieza independiente de código que proporciona una funcionalidad específica y fuertemente acoplado, módulos de definir y hacer cumplir los límites lógicos en el código. (Martí, 2012)

HTML: es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web. (Álvarez, 2001)

Base de Datos: Una base de datos es una herramienta para recopilar y organizar información. En las bases de datos, se puede almacenar información sobre personas, productos, pedidos, o cualquier otra cosa. (Microsoft, 2007)

UML: es un popular lenguaje de modelado de sistemas de software. Se trata de un lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software. (Alegsa, s,f)

Código Fuente: Texto escrito en un lenguaje de programación específico y que puede ser leído por un programador. Debe traducirse a lenguaje máquina para que pueda ser ejecutado por la computadora o a bytecode para que pueda ser ejecutado por un intérprete. (Alegsa, s,f)

Herramientas: Programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo. Se puede decir que una herramienta es cualquier programa o instrucción que facilita una tarea. (Ecured, 2015)

Utilerías: En informática, una utilidad es una herramienta que sirve de soporte para la construcción y ejecución de programas, en donde se incluyen las bibliotecas de sistema, middleware, herramientas de desarrollo. (Alegsa, 2009)

Sistema: Un sistema informático es un conjunto de partes que funcionan relacionándose entre sí con un objetivo preciso. (Alegsa, 2009)

Estructura: El concepto, que procede del latín structura, hace mención a la disposición y el orden de las partes dentro de un todo. (Definicion.de, 2015)

CMS: son las siglas de Content Management System, que se traduce directamente al español como Sistema Gestor de Contenidos. es un sistema que permite gestionar contenidos, un CMS permitiría administrar contenidos en un medio digital. (Alvarez, 2008)

Clase: Una clase es una definición de un objeto, en ella se encuentran todas las propiedades que componen a los diferentes objetos. (Alegsa, 2009)

Objeto: Una instancia de una clase, pueden existir muchos objetos de una clase. (Alegsa, 2009)

Actor: Un actor es una entidad externa al sistema que se modela y que puede interactuar con él. Puede ser una persona o un grupo de personas homogéneas, otro sistema, o una máquina. (Jojooa, s.f.)

RaquisFW: El nombre del proyecto presentado en esta tesis. Del latin rachis, y del griego. ῥάχις ráchis o columna vertebral (Real Academia Española, s.f.). El nombre es significante a la columna vertebral porque este le da soporte a todo el software como la columna vertebral al cuerpo.

Capítulo II

Marco Teórico

Este capítulo estudia los conceptos para comprender el desarrollo y funcionamiento de RaquisFW. En primer lugar se describe que es un framework, el funcionamiento básico, tipos de framework y las herramientas necesarias para su desarrollo.

Programa Informático

Según EcuRed (2015) menciona que los programas informáticos son un conjunto de instrucciones que una vez ejecutadas realizan una o varias tareas en una computadora y al conjunto general de programas se lo denomina software.

El comportamiento de un programa depende del tipo de lenguaje usado para escribirlo. Donde el interpretador ejecuta los comandos a medida en que va leyendo el archivo. Los programas informáticos se desarrollan basados en arquitecturas de software muy bien conocidas. Scott (s,f) menciona algunas de las arquitecturas universales y mayores reconocidas: Descomposición Modular, Cliente-Servidor, Arquitectura de tres niveles, Modelo Vista Controlador, Pipeline, Orientada a Servicios, Dirigida por Eventos y Máquinas Virtuales. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto.

Cliente-Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida donde las tareas se reparten entre los proveedores de recursos o servicios llamados servidores, y los

demandantes, llamados clientes. El cliente hace peticiones al servidor, el cual procesa dicho requerimiento y retorna los resultados al cliente apropiado. Por clientes y los servidores se comunican entre sí a través de una red, pero también pueden residir ambos en un mismo sistema . Wiley (s.f.) menciona las principales características de la arquitectura cliente-servidor.

- Servicio: Unidad básica de diseño. El servidor los proporciona y el cliente los utiliza.
- Recursos compartidos: Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos.
- Protocolos asimétricos: Los clientes inician “conversaciones”. Los servidores esperan su establecimiento pasivamente.
- Transparencia de localización física de los servidores y clientes: El cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.
- Independencia de la plataforma hardware y software que se emplee.
- Sistemas débilmente acoplados. Interacción basada en envío de mensajes.
- Encapsulamiento de servicios. Los detalles de la implementación de un servicio son transparentes al cliente.
- Escalabilidad horizontal y vertical.
- Integridad: Datos y programas centralizados en servidores facilitan su integridad y mantenimiento.

Alegsa (2015) menciona en las ventajas y desventajas de la arquitectura Cliente – Servidor. Ver tabla 1.

Tabla 1. Ventajas y Desventajas de Arquitectura Cliente-Servidor

Ventajas	Desvestajas
Centralización del control de los recursos, datos y accesos.	Si el número de clientes simultáneos es elevado, el servidor puede saturarse.
Facilidad de mantenimiento y actualización del lado del servidor.	Frente a fallas del lado del servidor, el servicio queda paralizado para los clientes.
Toda la información es almacenada en el lado del servidor, que suele tener mayor seguridad que los clientes.	

Aplicaciones Web

Kosher (2012) menciona que en el año 2012 existían aproximadamente 360 millones de páginas web y básicamente todas ellas no son más que un montón de documentos ligados, pero en 10 años todas serán una aplicación. Consumidores experimentaran marcas como una experiencia integral a lo largo de todos sus dispositivos y las páginas web lentamente se unirán a AOL y al tono de marcación como reliquias de eras pasadas.

Una aplicación web es una aplicación a la que se accede a través de internet u otras redes similares como intranet, que no requiere instalación para los usuarios. eNubes (2014) describe que las aplicaciones web se crean en respuesta a diversas necesidades o problemas. La aplicación web es un sitio web que contiene páginas con contenido sin determinar, parcialmente o en su totalidad. El contenido final de una página se determina sólo cuando el usuario solicita una página del servidor web. Dado que el contenido final de la página varía de una petición a otra en función de las acciones del visitante, este tipo

de página se denomina página dinámica. Adobe (2014) refiera las principales ventajas de las aplicaciones web:

- Reducción de costes.
- Ahorro de tiempo.
- Evita los problemas de compatibilidad entre sistemas.
- No ocupan espacio en disco duro porque se aloja en la Nube.
- Están siempre actualizadas.
- No consume recursos.
- Multiplataforma.
- Portables.
- Alta disponibilidad.
- Desarrollo de aplicaciones web.
- Seguridad frente a virus y hackers.
- Colaboración.
- Mejores funcionalidades para creación de aplicaciones web
- Permiten compartir información con otros usuarios.
- Fáciles de usar gracias a las herramientas de formación online.
- Constituyen un nuevo canal de publicidad.
- Si son de pago, se pueden pagar online con tarjeta de crédito u otro medio.
- Ofrecen información sobre el comportamiento del usuario que se puede utilizar para las estrategias de marketing y publicidad.

MVC

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello el MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

De acuerdo con Conningham (2014) el MVC fue inventado por Xerox Parc en los años 70's, fue desarrollado para ser usado por un lenguaje de programación llamado smalltalk-80. Por años no existió ningún tipo de documentación publica hasta que el primer documento significativo publicado sobre MVC fue realizado por Glenn Krasner y Stephen Pope en Agosto/Septiembre de 1988 y fue titulado "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk -80".

AmSalihefendic (s.f) menciona que Smalltalk es un lenguaje de programación puramente orientado a objetos que es muy interactivo y MVC es su característica principal. El objetivo principal de usar MVC en Smalltalk era separar la lógica de la aplicación de la lógica de la interfaz permitiendo la reutilización del modelo para formar diferentes interfaces de usuarios. Una aplicación capaz de poder dividir, la lógica de la aplicación y la interfaz permite, poder controlar el modelo de diferentes interfaz, un

ejemplo podría ser una interfaz capas de ser controlador mediante una interfaz gráfica, consola, y remotamente de un navegador. Ver figura 2.

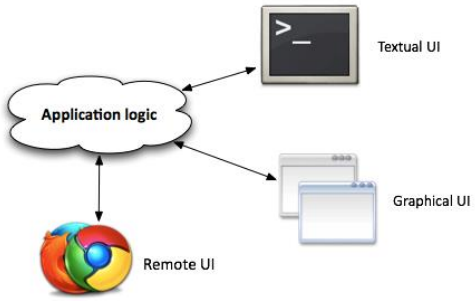


Figura 2: Separación de lógica de la aplicación e interfaz de usuario

PHP

The PHP Group (2001- 2014) expresa que el acrónimo de PHP: (Procesador de Hipertexto) es un lenguaje para hacer scripts de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, facilitando el aprendizarlo. La meta principal de este lenguaje es permitir a los desarrolladores web escribir dinámica y rápidamente páginas web generadas; aunque se puede hacer mucho más con PHP.

PHP creció rápidamente desde su primera versión nombrada originalmente Herramientas de Páginas Personales (Personal Home Page Tools,PHP Tools) liberada al público por Rasmus Lerdorf en 1995.

NetCrasft (s.f) expresa que para enero del año 2013, PHP estaba usado por la cantidad impresionante de 244 millones de páginas, siendo esto el 39% de todas las páginas. Ide (2013) argumenta que algunas de las aplicaciones web más populares son gestores de contenido tales como Wordpress, Joomla y Droopal al igual que algunas soluciones de comercio en línea tales como Zencart, osCommerce y Magento. En enero

del 2013 estas seis aplicaciones solas se encontraron corriendo en más de 32 millones de sitios.

Sintaxis de PHP

La forma más sencilla de mostrar la sintaxis de un lenguaje de programación es ver el clásico ejemplo de Hola Mundo. En la figura 3 se muestra el programa de Hola Mundo embebido en HTML.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Hola Mundo</title>
5      </head>
6      <body>
7          <?php echo '<p>Hola Mundo</p>'; ?>
8      </body>
9  </html>
```

Figura 3: Estructura de “hola mundo” en PHP embebido en HTML.

PHP no tiene que ser forzosamente embebido en HTML así que la forma más sencilla de obtener este resultado sería como se muestra en la fugura 4

```
1  <?='Hola Mundo';?>
```

Figura 4: Hola mundo en PHP

Delimitadores

The PHP Group (2001- 2014) expresa que PHP puede reconocer y ejecutar el código escrito es necesario que el interpretador pueda identificarlos, para ello estos deben de estar dentro de un delimitador. Los delimitadores más comunes se muestran en la figura 5

```
<?php echo '#1 si se quiere mostrar documentos XHTML o XML, debe
hacerse así'; ?>
<script language="php">
        echo '#2 algunos editores (como FrontPage) no les gusta
              las instrucciones de proceso';
</script>
<?='#3 esta es la forma más simple, una instrucción de procesado
SGML'; ?>
<?=$expresión?> Esto es una forma abreviada de <?php echo
$expresión;?>
<% echo '#4 Quizá use de forma opcional etiquetas de estilo ASP';
%>
```

Figura 5: Ejemplo de delimitadores en PHP.

Variables

W3Shools (2014) expresa que en algebra se usan letras como x para almacenar valores como en la figura 5. En PHP estas letras se llaman variables. Como en algebra, variables de PHP pueden ser usados para almacenar valores como (x=5) o expresiones como (z=x+y), estas variables pueden tener nombres cortos (x e y) o nombres más descriptivos (edad, nombre_carro, volumen_total).

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas. Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (underscore), seguido de cualquier número de letras, números y caracteres de subrayado como se muestra en la figuara 6.

```
<?php
$var = 'Roberto';
$Var = 'Juan';
echo $var, $Var;      // imprime "Roberto, Juan"
$4site = 'aun no';      // inválido; comienza con un número
$_4site = 'aun no';     // válido; comienza con un carácter de
subrayado
$täyte = 'mansikka';    // válido; 'ä' es ASCII (Extendido) 228
?>
```

Figura 6: Variables validos en PHP.

PHP ofrece otra forma de asignar valores a las variables: asignar por referencia. Eso significa que la nueva variable simplemente referencia la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Para asignar por referencia, simplemente se antepone un signo ampersand (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, en la figura 7 se muestra la salida 'Mi nombre es Bob' dos veces:

```php
<?php
$foo = 'Bob';               // Asigna el valor 'Bob' a $foo
$bar = &$foo;               // Referenciar $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $bar;
echo $foo;                  // $foo también se modifica.
?>
```

Figura 7: Variables por referencia

Variables con variables

En ocasiones es conveniente asignar variables con nombre de otras variables. Un nombre que pueda ser movido dinámicamente. Una variable, variable toma el valor de una variable y automáticamente la toma como el nombre de la variable. En la figuras se muestra un ejemplo.

```
<?php
$a = “hello”;
$hello=”world”;
echo $$a; // es igual a echo $hello;
¿>
```

Figura 8: Variables, Variables

Esto puede ser de gran benéficio y útil mientras se programa, otra forma en que se pueden usar las variables variables es cuando los nombres de las variables siguen algún patrón, la Figura 3 muestra un ejemplo.

```php
<?php
// Teniendo las siguientes variables
$tiposNombres    = array("primero", "segundo", "empresa");
$nombre_primero   = "John";
$nombre_segundo    = "Doe";
$nombre_tercero = "PHP.net";

// El loop seria
foreach($tiposNombres as $tipo)
  print ${"nombre_$tipo"} . "\n";

// ... este seria su equivalente.
print "$name_primero\n$name_segundo\n$name_empresa\n";
?>
```

Figura 3: Ejemplo de Variables Variables

Framework

La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de carácter similar.

Sandobal (s.f.) menciona que en el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Un framework es una aplicación reutilizable y semi-completa que puede ser especializada para producir aplicaciones individualizadas- ya que es una abstracción de software que proporciona una funcionalidad genérica que debe ser cambiada de forma selectiva por el código adicional escrito por el usuario, lo que genera el software de aplicación específica.

Los frameworks incluyen programas de apoyo, compiladores, bibliotecas de código, juegos de herramientas e interfaces de programación de aplicaciones que reúnen a todos

los diferentes componentes para permitir el desarrollo de un proyecto o solución. Un framework resuelve alguna situación repetitiva que quitan mucho tiempo de programación o que complique la implementación, de forma tal que el desarrollador pueda concentrarse en implementar la lógica de negocio particular de la aplicación, permitiendo que el framework se encargue del enrutamiento y las tareas comunes de toda aplicación.

Tipos de Framework

Prakash (2008) Menciona que los frameworks se categorizan en dos tipos los frameworks GLUE y los FullStack. Los frameworks tipo GLUE no están intensamente ligados y proveen mayor flexibilidad durante el desarrollo. Permiten a los desarrolladores avanzados usar código reutilizable existente dentro del framework. Los frameworks Full-Stack están muy acoplados, dificultando el desarrollo de ideas propias, pero permite menos trabajo para generar aplicaciones.

Frameworks existentes

Laravel

Laravel (2014) menciona que, es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC. Gran parte de Laravel está formado por dependencias, especialmente de Symfony, esto implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias.

Github (2014) expresa que Laravel es el proyecto más popular y más seguro en el año 2014 por otra lado SitePoint (2015) en su encuesta anual encontró que Laravel es el framework más popular de PHP en el 2015, seguido por Symfony y CodeIgniter.

Maxoffsky (2013) menciona que desde su primera versión, Laravel contaba con muchas funcionas como autentificación, localización, Modelos relaciones, un enrutamiento sencillo, caching, sesiones, vistas que podían ser extendidas mediante módulos y librerías. Lamentablemente aun no era un framework completo por que no incluía el uso de controladoros, expresa que en sus siguientes versiones dos y tres se introdujo muchas mejores como integración completa de MVC, integración completa de controles para sesiones y bases de datos, eventos. En su versión 4 laravel fue reescrita desde el piso como una colección de componentes trabajando en conjunto para formar un framework. Esta versión incluyo componentes que no se encontraban en ningún otro framework como sembrado de base de datos, cola de mensajes, manejador de correos, entre otros.

TutsPlus (2014) describe el código de Laravel como una artesanía por su sencillez y belleza. Menciona que permite hacer código muy elegante y legible.

Phalcon

Sipos (2014) describe a Phalcon como un framework que promueve el uso de la arquitectura Modelo, Vista, Controlador, menciona que Phalcon fue lanzado al mercado en el año 2012 bajo la licencia de código abierto BSD. Una de sus características principales es que utiliza una extensión en C para poder procesar peticiones más rápido que aquellos frameworks escritos en PHP.

SitePoint (2014) menciona que una de las principales desventajas de PHP es que cada petición, todos los archivos son leídos del disco duro, traducidos en bytecode y después ejecutado. Con phalcon todo el framework se encuentra en RAM, así que cualquier archivo del framework no se tiene que cargar solo se lee y no tienen que ser procesados, esto permite un rendimiento superior al de cualquier otro framework en PHP como se muestra en las

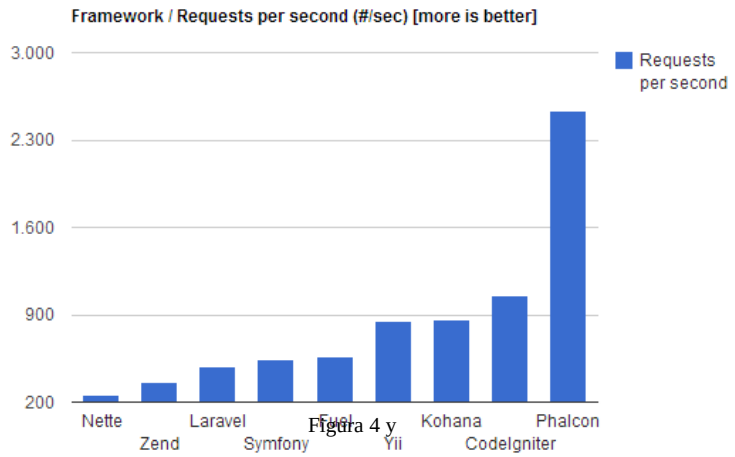


Figura 4 y

Figura 5.

Figura 4: Peticiones por segundo contestadas.

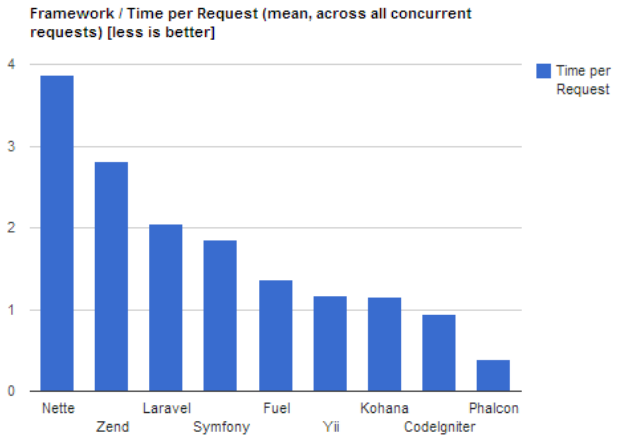


Figura 5: Respuesta de los frameworks por segundo.

Symfony

Symfony (2011) proporciona herramientas y clases encaminadas a reducir tiempo de desarrollo de una aplicación web compleja. Symfony (2014) menciona que la primera versión fue lanzada en octubre del año 2005, por Fabien Potencier. Originalmente fue creado para el desarrollo de las aplicaciones de Sensio. Luego, tras el éxito que tuvo en el desarrollo de una página web para comercio electrónico y algunos otros proyectos, decidió liberarlo bajo una licencia código abierto. En el año 2012 el CMS Drupal decidió empezar a usar algunos componentes de Symfony en la versión Drupal 8.

Una característica importante de Symfony es que en lugar de crear sus propias librerías hace uso de algunas de código abierto, al igual que algunos de sus propios componentes. Algunos de los módulos más populares de symfony son: Propel, Doctrine,

PDO, PHPUnit, Twig, Swift Mailer, Symfony YAML, Symfony Event Dispatcher, Symfony Dependency Injector, Symfony Templating, Prototype, jQuery, Script.aculo.us, lessphp, TinyMCE, CKEditor y TCPDF.

CodeIgniter

EllisLab (2015) menciona que CodeIgniter fue creado por Rick Ellis en el año 2006, fue basado en ExpressEngine, un CMS robusto. Se tomó el CMS y se removieron todo el código específico de la aplicación, y se obtuvo una fuerte y robusta Liberia de clases que permiten el rápido desarrollo de páginas y aplicaciones web. En el año 2014 EllisLab transfirió los derechos del framework al BCIT (British Colombia Institute of Technology) donde se ha transformado de una simple librería de clases a un laboratorio viviente de innovación.

Quora (2013) menciona a ActiveRecords y Hooks como las características originales de CodeIgniter. Hooks en una funcionalidad del framework que permite reescribir parte del framework sin afectar al resto de la aplicación. Permite tener mejor control de lo que está sucediendo en el framework, sin tener que tocar el código original. ActiveRecord es una idea donde cada tabla en una base de datos es representado con un modelo el cual tiene el control de esta tabla puede editar, borrar y buscar. Permite ejecutar código de SQL sin escribirlo, algo parecido a un CRUD.

UML

Grandes aplicaciones empresariales, las que ejecutan las aplicaciones fundamentales del negocio, y permiten que la empresa siga funcionando deben de ser más que un

montón de módulos de código. Deben de ser estructuradas de una manera que permita escalabilidad, seguridad, y robustez en la ejecución ante las peores condiciones. Su estructura, frecuentemente referida como su arquitectura debe de ser suficientemente clara para que un programador pueda tener un mejor entendimiento y de esa forma arreglar errores en el software. Los programas deben de ser diseñados para trabajar perfectamente en diversas áreas, y la funcionalidad del negocio no es lo único importante, aunque si esencial.

OMG.org (2014) menciona que UML (Modelo de Lenguaje Unificado) ayuda a especificar, visualizar y documentar modelos de un sistema de software, incluyendo su estructura y diseño, en una manera que cumpla con todos los requerimientos. Usando cualquiera de las muchas herramientas de UML en el mercado, se puede analizar los futuros requerimientos y diseñar una solución que cumpla los requerimientos, representando los resultados usando cualquier de los diagramas estándares de UML 2.0.

Modelo de Clase

Un diagrama de clase sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de consentimiento. Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: herencia, composición, agregación, asociación y uso.

Es la unidad básica que encapsula toda la información de un Objeto. A través de ella podemos modelar el entorno en estudio.

En UML, una clase es representada por un rectángulo que posee tres divisiones como se muestra en la Figura 12:

- Superior: Contiene el nombre de la clase
- Intermedio: Contiene los atributos (o variables de instancia) que caracterizan a la clase (pueden ser privado, protegido o público).
- Inferior: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: privado, protegido o público).

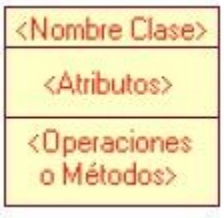


Figura 12 Estructura de una clase en UML

Modelo de Casos de Uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

Actor:

Un actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, mas la labor que realiza frente al sistema.

Caso de Uso:

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Relaciones:

Asociación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Dependencia

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia. Dicha relación se denota con una flecha punteada.

Generalización

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de uso (<<user>>) o de herencia (<<extends>>). Este tipo de relación está orientado exclusivamente para casos de uso la representación de la diagramación de estas propiedades de UML se muestran en la figura 13.

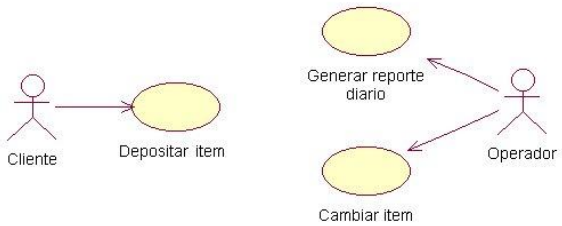


Figura 13: Estructura de un diagrama de Casos de Uso.

Diagrama de Interacción

El diagrama de interacción, representa la forma en como un cliente u objetos se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. Dicho diagrama puede ser obtenido de dos partes, desde el diagrama estático de clases o el de casos de uso. Los componentes de un diagrama de interacción y sus representaciones gráficas se muestran en las figuras 14 al 16.

Un objeto o actor.

El rectángulo representa una instancia de un objeto en particular, y la línea punteada representa las llamadas a métodos del objeto ver figura 14

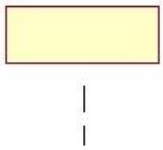


Figura 14: Figura usada para representar un objeto o actor en UML.

Mensaje de un objeto a otro objeto.

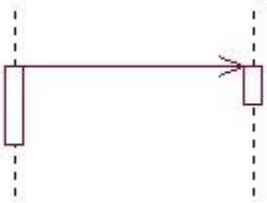


Figura 15: Mensaje de un objeto a otro

Se representa por una flecha entre un objeto y otro, representa la llamada de un método de un objeto en particular ver figura 15.

Mensaje de un objeto a sí mismo.

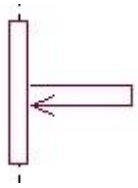


Figura 6: Mensaje de un objeto a si mismo

No solo llamadas a métodos de objetos externos pueden realizarse, también es posible visualizar llamadas a métodos desde el mismo objeto en estudio ver figura 16.

Ejemplo:

En las figuras 17 y 18 se representa una aplicación que posee una ventana gráfica, y ésta a su vez posee internamente un botón. Entonces el diagrama de interacción para dicho modelo es:

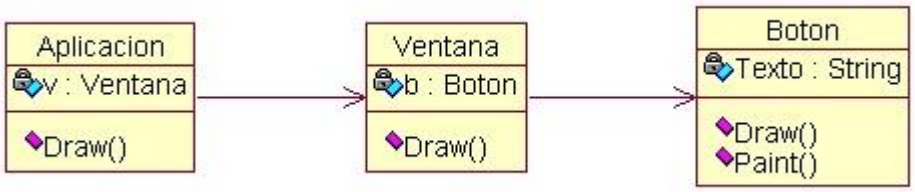


Figura 17: Diagrama de clase de aplicación con una ventana gráfica.

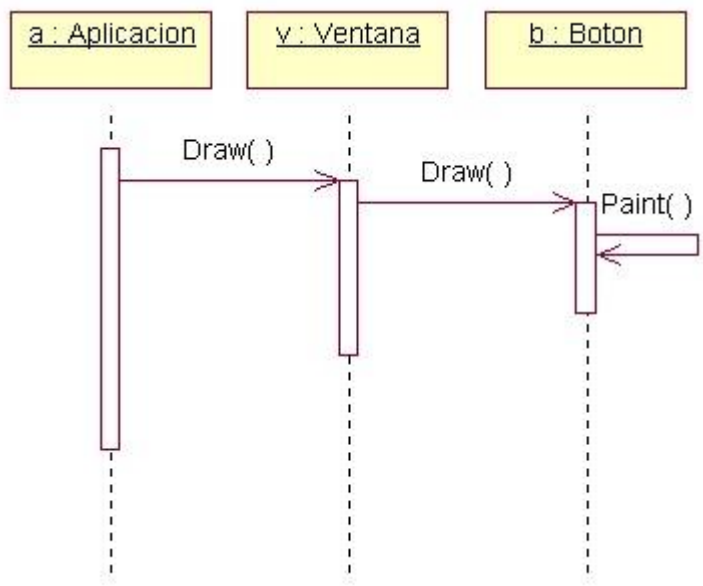


Figura 18: Diagrama de secuencia de aplicación con una ventana gráfica.

Base de Datos

Microsoft (2014) define una base de datos como una herramienta para recopilar y organizar información. En las bases de datos se puede almacenar información sobre personas, productos, pedidos o cualquier otra cosa. Valdés (2007) menciona que el término base de datos fue escuchado por primera vez en el año 1963 durante un simposio celebrado en California, Estados Unidos.

Desde el punto de vista informático la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Una base de datos a fin de ordenar la información de manera lógica, posee un orden que debe ser cumplido para acceder a la información de manera coherente. Cada base de datos contiene una o más tablas que cumplen la función de contener los campos.

Tabla

Las tablas es donde se almacena la información en la base de datos es simplemente una representación en dos dimensiones de la información con columnas y filas como se muestra en la figura 19.

John	Smith	jsmith@huh.com
Paul	McCartney	paul@beatles.com
Bill	Murray	gopher@caddyshack.com

Figura 19: Tabla simple

Cada base de datos se le da un nombre único, para que el gestor de base de datos las pueda identificar fácilmente. Así mismo las columnas en las tablas tendrían un nombre único, tomando en cuenta la figura 20, los nombres podrían ser primer_nombre, apellido ó correo. Es importante tomar en cuenta que no se pueden usar espacios ni símbolos en los nombres de las bases de datos, es recomendable usar "_" en lugar de un espacio.

nombre	primer_apellido	correo
John	Smith	jsmith@huh.com
Paul	McCartney	paul@beatles.com
Bill	Murray	gopher@caddyshack.com

Figura 20: Identificadores de tablas.

Clave primaria

La clave primaria es una columna en nuestra base de datos que garantiza ser única. Esta es usada para el propósito de indexar la tabla, la cual la hace más eficiente las búsquedas, la organización y el enlace de las tablas. La clave primaria generalmente es

una columna que la mayoría de los desarrolladores llaman ID (identificador), generalmente esta columna se llena sola al insertar datos.

MySql

Oracle Corporation (2014) es el gestor de bases de datos de código abierto más popular, es desarrollado, distribuido, y soportado por Oracle Corporation. W3Schools (2014) registra una lista de lo que es MySQL:

- Es un sistema de base de datos que corre en un servidor.
- Es muy rápida, confiable y fácil de usar.
- Usa SQL estándar.
- Es compilado en un número de plataformas.
- Es gratis de descargar y usar.
- Es nombrado en honor a la hija del co-fundador Monty Widenios.

Mysql mejores prácticas

Guzel (2009) expresa que las operaciones de la base de datos casi siempre son el cuello de botella de la mayoría de las aplicaciones web hoy en día. Afirma que no solo es el administrador de bases de datos el que se tiene que preocuparse por problemas de rendimiento, sino también los programadores deben de optimizar los querys y escribir mejor código. Graville (2011) proporciona una lista de las mejores prácticas de MySQL:

1. Indexar campos de búsqueda
    - Los índices no son solo para las claves primarias y las claves de unicidad. Si hay alguna columna en la tabla por la cual realizará una búsqueda, casi siempre debería ser indexada. Esto optimiza las búsquedas que realiza MySQL al generar la consulta.

2. Evitar usar “SELECT *” en los querys
    - Mientras más datos se leen de las tablas, más lenta será la consulta. El tiempo que tardan las operaciones en el disco duro aumenta. También, cuando el servidor de la base de datos está separado del servidor web, existirá más retrasos de red debido a que los datos tienen que ser transferidos entre los servidores. Es un buen hábito especificar siempre las columnas que se necesitan en el SELECT.
3. Siempre usar un campo id
    - En cada tabla es importante tener un campo ID que sea clave primaria con auto incremento y entero. Incluso si se cuenta con una tabla usuarios que tiene un campo nombre_de_usuario con clave de unicidad, es recomendable no usarla como clave primaria.
4. Evitar asignar columnas valores nulos.
    - A menos que tengas una razón específica para usar un valor NULL, siempre debes designar tus campos como NOT NULL. Los campos NULL requieren un espacio adicional y pueden añadir complejidad a la comparación de sentencias. MySQL (2014) expresa que las columnas NULL requieren espacio adicional en el registro sin importar si su valor es NULL o no. Para las tablas MyISAm, cada columna NULL toma un bit extra, redondeado por encima al byte más cercano”.
5. Particionamiento vertical

- Esto consiste en dividir la estructura de la tabla de una forma vertical por razones de optimización.
- Si se tiene una tabla de usuarios que contenga la dirección del hogar, eso no se lee a menudo. Se puede dividir la tabla y almacenar la información de la dirección en una tabla aparte. De esta manera la tabla usuarios será encogida. Las tablas más pequeñas se desenvuelven más rápido.
- Es común tener un campo "ultima_visita" en la tabla de usuarios. Se actualiza cada vez que un usuario inicia sesión en el sitio web. Pero cada actualización en una tabla hace que la caché de consultas de esa tabla sea desechada. Es posible poner ese campo en otra tabla para mantener las actualizaciones de la tabla de usuarios al mínimo.

6. Escoger bien el tipo de motor de almacenamiento
    - Los dos motores de almacenamiento principales de MySQL con MyISAM e InnoDB. Cada uno tienes sus propios ventajas y desventajas. MyISAM es bueno para aplicaciones de con mucha lectura, pero no es muy amplio cuando hay muchas escrituras. Incluso mientras se actualiza un campo en una fila, toda la tabla se bloquea, y ningún otro proceso puede ser procesado hasta que la consulta haya finalizado. MyISAM es muy rápido calculando consultas de tipo SELECT COUNT(*).
    - InnoDB tiende a ser un motor de almacenamiento más complicando y puede ser más lento que MyISAM para las aplicaciones más

pequeñas. Pero soporta el bloqueo basado en filas, el cual escala mejor. También soporta algunas características más avanzadas como las transacciones.

7. Usar “LIMIT 1” cuando ocupamos solo un resultado
    - A veces cuando se consulta las tablas, se sabe que se está buscando sólo una fila. En estos casos, añadir LIMIT 1 a la consulta puede incrementar el desempeño. De esta manera el motor de la base de datos dejará de buscar registros luego de encontrar 1, en vez de ir a través de toda la tabla o índice.

MYSQLI

En las versiones de PHP5 en adelante **mysqli** es usado para comunicarse con la base de datos, en su anteoridad era usado **mysql,** el cual fue deprecado. Antes de acceder a la base de datos necesitamos generar una conexión a ella como se muestra en la figura 21.

```
<?php
$nombre_del_servidor = "localhost";
$usuario = "usuario";
$contraseña = "contraseña";

// Crear conexión
$conn = new mysqli($nombre_del_servidor, $usuario, $contraseña);

// Verificar conexión
if ($conn->connect_error) {
    die("La conexión fallo: " . $conn->connect_error);
}
echo "Se conectó satisfactoriamente!";
?>
```

Figura 21: Código para conectarse a una base de datos mediante PHP

Un query de SQL puede ser pasado a la base de datos usando la función **query()** la cual regresara algún resultado, este puede ser diferente dependiendo en el tipo de query

que es enviado. En el caso de que el tipo de query sea “SELECT” se usa la función **fetch_assoc()** para ser convertido a un arreglo, ver figura 22.

```
<?php
$sql = <<<SQL
    SELECT *
    FROM `usuarios`
    WHERE `estatus` = 1
SQL;
if(!$result = $db->query($sql)){
    die('Error: [' . $db->error . ']');
}
while($row = $result->fetch_assoc()){
    echo $row['nombre_usuario'] . '<br />';
}
?>
```

Figura 22: Imprimir una tabla de usuarios activos usando query() y fetch_assoc().

La conexión será cerrada automáticamente al terminar el script, pero se puede terminar antes con el código mostrado en la figura 23.

```
mysqli_close($conn);
```

Figura 23: Cerrar una conexion a la base de datos.

CAPITULO III

Metodología del proyecto

Oliva (s.f) refiere a ICONIX como un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto, después menciona que ICONIX se encuentra entre la complejidad de RUP (Proceso Racional Unificado) y la simplicidad de XP (Programación Extrema). ICONIX fue elaborado por Doug Rosenberg y Kendall Scott a partir de una síntesis del proceso unificado de los "tres amigos" Booch, Rumbaugh y Jacobson y que ha dado soporte y conocimiento a la metodología ICONIX desde 1993. Oliva (s,f) menciona que ICONIX presenta claramente las actividades de cada fase y exhibe una secuencia de pasos que deben ser seguidos. Además que ICONIX está adaptado a los patrones y ofrece soporte de UML, dirigido a casos de uso.

Las tareas de ICONIX

Rosenberg y Scoot (2005) destacan para ICONIX un análisis de requisitos, análisis y diseño preliminar, un diseño y una implementación como las principales tareas. Análisis de requisitos:

- Se identificar en el "mundo real" los objetos y todas las relaciones de agregación y generalización entre ellos. Utilizar un diagrama de clases de alto nivel definido como modelo de dominio.

- Se presentar, un prototipo rápido de las interfaces del sistema, los diagramas de navegación, entre otros, de forma que los clientes puedan comprender mejor el sistema propuesto.
- Se identifican los casos de uso del sistema mostrando los actores involucrados. Utilizar para representarlo el modelo de casos de uso
- Se organizan los casos de uso en grupos, es de decir utilizar los diagramas de paquetes.
- Asociar los requisitos funcionales con los casos de uso y con los objetos del dominio.

Análisis y Diseño Preliminar:

- Se describen los casos de uso, como un flujo principal de acciones, pudiendo contener los flujos alternativos y los flujos de excepción. La principal sugerencia de ICONIX, en esta actividad, es que no se debe perder mucho tiempo con la descripción textual. Debería usarse un estilo consistente que sea adecuado al contexto del proyecto.
- Se realiza un diagrama de robustez donde se debe ilustrar gráficamente las interacciones entre los objetos participantes de un caso de uso. Este diagrama permite analizar el texto narrativo de cada caso de uso e identificar un conjunto inicial de objetos participantes de cada caso de uso.
- Se actualiza el diagrama de clases ya definido en el modelo de dominio con las nuevas clases y atributos descubiertas en los diagramas de robustez.

Diseño:

- Se especificar el comportamiento a través del diagrama de secuencia. Para cada caso de uso identificar los mensajes entre los diferentes objetos. Es necesario utilizar los diagramas de colaboración para representar la interacción entre los objetos.
- Se terminar el modelo estático, adicionando los detalles del diseño en el diagrama de clases.
- Se verifica si el diseño satisface todos los requisitos identificados

Implementación:

- Se utiliza el diagrama de componentes, si fuera necesario para apoyar el desarrollo. Es decir, mostrar la distribución física de los elementos que componen la estructura interna del sistema.
- Se genera el código
- Se realizan pruebas de unidades, de casos, datos y resultados. Pruebas de integración con los usuarios para verificar la aceptación de los resultados.

CAPITULO IV

Análisis de Requisitos

Este capítulo se enfoca en la primera etapa de la metodología ICONIX y el análisis de requerimientos dividido en cuatro partes.

- Primero: Utiliza un diagrama de clases de alto nivel definido como modelo de dominio se realiza una abstracción de los objetos y las relaciones de agregación y generalización que existen entre ellos.
- Segundo: Presenta un prototipo rápido de las interfaces del sistema, diagramas de navegación, para dar a los clientes una mejor comprensión del sistema propuesto.
- Tercero: Se identifican los casos de uso del sistema y los actores involucrados, utilizando el modelo de casos de uso para su representación.
- Cuarto: Realiza una revisión de los requisitos funcionales con los casos de uso y con los objetos del dominio.

Especificación de requisitos

La especificación de requisitos es realizada en un identificador (id), una descripción de requisitos, una necesidad que refleja la importancia del requisito para el usuario y la prioridad que se establece para el requisito.

Requisitos No Funcionales

Olivera (2011) menciona que los requerimientos no funcionales, son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo,

sobre el proceso de desarrollo y estándares, estos requerimientos no se refieren directamente a las funciones específicas que proporcion el sistema, si no a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamietos, de forma alternativa, definen las restricciones del sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivo de entrada/salida y las representaciones de daos que se utilizan en las interfaces del sistema. Estos requerimientos a menuno se aplican en el sistema en su totalidad. La tabla 2 presenta el listado de los requerimientos no funcionales de RaquisFw

Tabla 2: Requerimientos no funcionales de RaquisFw

ID	Descripción	Necesidad	Prioridad
R001	Framework orientado a MVC	Esencial	Alta
R002	Facilitar el desarrollo de aplicaciones	Esencial	Alta
R003	Agilizar el desarrollo de aplicaciones	Esencial	Alta
R004	Sugerir estándares y convenciones	Esencial	Alta
R005	El framework debe de funcionar modularmente	Esencial	Alta
R006	Debe de ser programado para un público de habla español	Esencial	Alta
R008	Incluir Herramientas que ayuden al desarrollador	Esencial	Alta

Requisitos Funcionales

Olivera (2010) describe que los requerimientos funcionales son declaraciones de los servicios que se deben proporcionan en el sistema, la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones especificas, también puede declarar explícitamente lo que el sistema no debe hacer. Las siguientes tablas presentan los requisitos funcionales de cada área como lo son los usuarios, sesiones, base de datos entre otros.

Tabla 3: Requisitos funcionales del RaquisFw

ID	Descripción	Necesidad	Prioridad
R009	Utilizar URL Rewrite para hacer los URL de las aplicaciones amistosas a los buscadores	Esencial	Alta
R010	usar una configuración global en las aplicaciones	Esencial	Alta
R011	Cargar clases automáticamente	Esencial	Alta
R012	Usar Logs	Esencial	Alta
R013	Mantener sencillo del enrutamiento de la aplicación	Esencial	Alta
R014	Clase madre de control con funciones básicas	Esencial	Alta
R015	Clase madre de modelo con funciones básicas	Esencial	Alta
R016	Clase de vista o plantilla con funciones básicas	Esencial	Alta
R017	Controlador de base de datos	Esencial	Alta
R018	Controlador de Usuarios	Esencial	Alta
R019	Control de Sesiones	Esencial	Alta
R020	Control de Formas	Esencial	Alta
R021	Control de Plantillas	Esencial	Alta
R022	Control de Herramientas	Esencial	Alta

Tabla 4: Requisitos funcionales de usuario

ID	Descripción	Necesidad	Prioridad
R023	Crear usuarios	Deseada	Baja
R024	Eliminar usuarios	Deseada	Baja
R025	Editar usuarios	Deseada	Baja
R026	Autentificar Usuarios	Esencial	Alta
R027	Verificar Usuario	Esencial	Alta
R028	Obtener Información de Usuario	Esencial	Alta

Tabla 5: Requisitos funcionales de sesiones

ID	Descripción	Necesidad	Prioridad
R029	Iniciar sesión	Esencial	Alta
R030	Terminar sesión	Esencial	Alta
R031	Verificar sesión	Esencial	Alta

Tabla 6: Requisitos funcionales de base de datos

ID	Descripción	Necesidad	Prioridad
R030	seleccionar todos los campos en una tabla	Esencial	Alta
R031	seleccionar un solo campo	Esencial	Alta
R032	actualizar campo	Esencial	Alta
R033	convertir resultados a array	Esencial	Alta
R034	insertar valores	Esencial	Alta
R035	existe campo en base de datos	Esencial	Alta
R036	contar campos	Esencial	Alta
R037	convertir a fecha mysql	Esencial	Alta
R038	insertar por array	Esencial	Alta
R039	actualizar por array	Esencial	Alta

Tabla 7: Requisitos funcionales de herramientas

ID	Descripción	Necesidad	Prioridad
R040	mandar correos por SMPT	Deseado	media
R041	mandar correos simples	Deseado	media
R042	mandar correos con datos adjuntos	Deseado	media
R043	convertir html a pdf	Deseado	media
R044	re direccionamiento seguro	Deseado	media
R045	mandar fax	Deseado	media
R046	mandar mensajes de texto	Deseado	media
R047	generador de números aleatorios	Deseado	baja
R048	generador de códigos de barra	Deseado	baja
R049	generador de códigos QR	Deseado	baja
R050	generador de captcha	Deseado	baja
R051	gestor de archivos	Deseado	baja

Modelo de Dominio

Con los requisitos se construye el modelo del dominio, que representa el modelo estático del sistema mostrado en la figura 24.

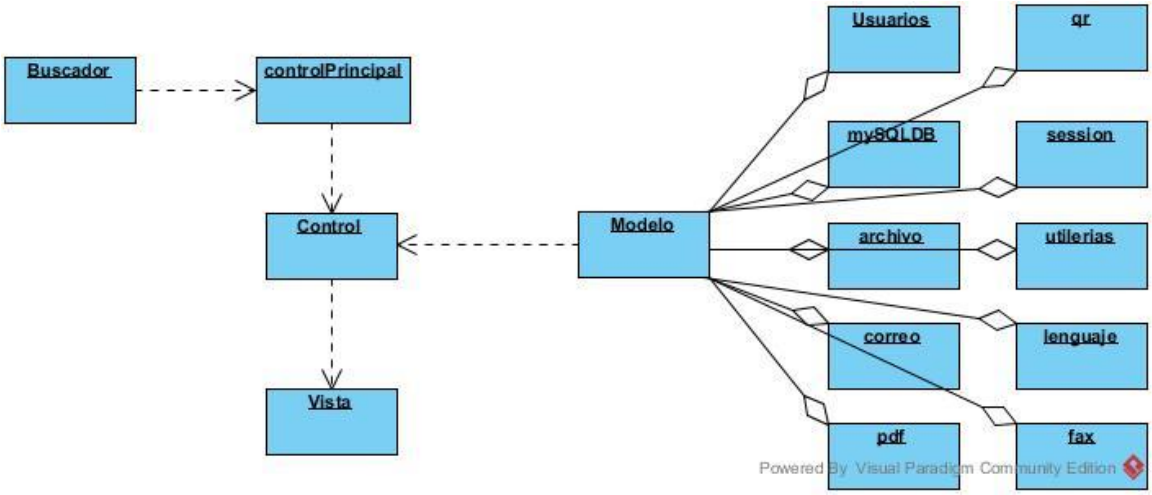


Figura 24: Modelo del Dominio

Modelo de Casos de Uso

Master Magazine (2015) expresa que los casos de uso son una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. El diagrama de caso de uso general de RaquisFw se muestra en la figura 25 y la Tabla 8 se muestra los principales actores del framework.

Tabla 8: Principales actores de RaquisFw

Nombre	Descripción
Control Principal	El control principal se encarga de recibir petición y mandarlas al control de la aplicación.
Control	Interpreta las peticiones y la manda al modelo y la vista.
Modelo	Consulta bases de datos, módulos usando la lógica de la aplicación.
Vista	Encargada de generar una respuesta visual al usuario.

Tabla 8: Principales actores del Framewors RaquisFw y la descripción de los roles que realiza en el framework.

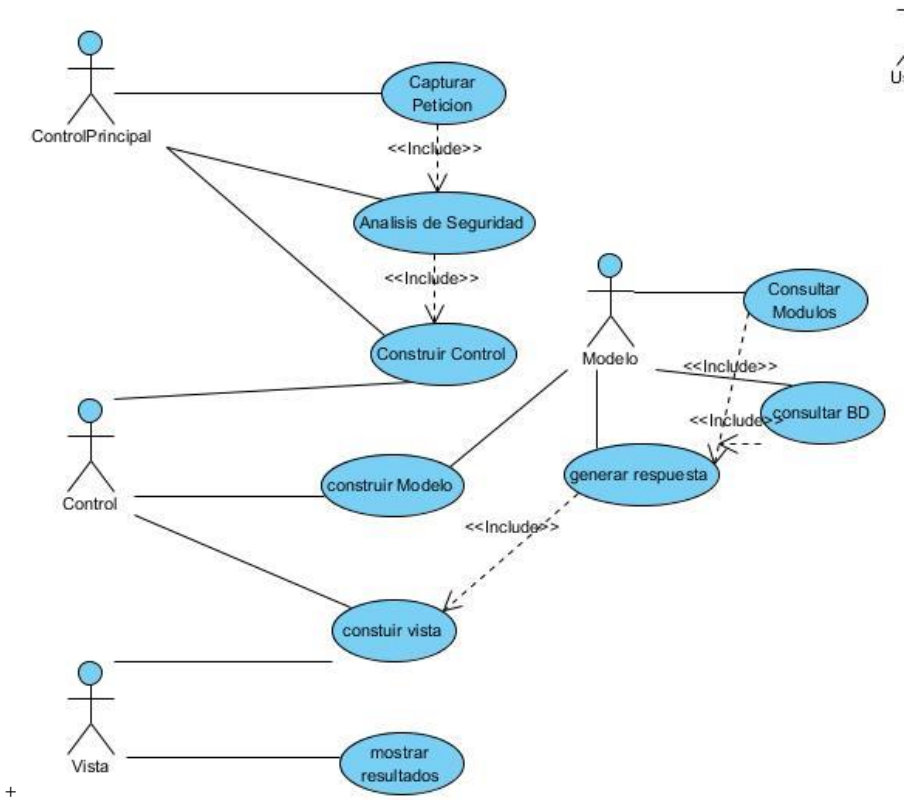


Figura 25: Diagrama de casos de uso general

CAPÍTULO V

Análisis y Diseño preliminar

Este capítulo muestra la documentación obtenida de la segunda etapa de la metodología el análisis y diseño preliminar, el cual cuenta con tres tareas principales. Primero se describen los casos de uso con un flujo principal de acciones y posibles flujos alternos y de excepción. Segundo se realiza un diagrama de robustez, en donde se debe ilustrar las interacciones existentes entre los objetos participantes de un caso de uso. Tercero se actualiza el diagrama de clases definido en el modelo del dominio con las clases y atributos encontrados en los diagramas de robustez. La tabla 9 presenta el listado de las los casos de uso utilizados referenciando a los anexos correspondientes al diagrama y su respectiva descripción.

Tabla 9: Listado de casos de usos de RaquisFw

ID	Nombre	Descripción	Diagrama
CU01	Capturar Petición	Anexo A.1	Anexo A.2
CU02	Análisis de Seguridad	Anexo B.1	Anexo B.1
CU03	Construir Control	Anexo C.1	Anexo C.2
CU04	Construir Modelo	Anexo D.1	Anexo D.2
CU05	Consultar Módulos	Anexo E.1	Anexo E.2
CU06	Consultar BD	Anexo F.1	Anexo F.2
CU07	Generar Respuesta	Anexo G.1	Anexo G.2
CU08	Construir Vista	Anexo H.1	Anexo H.2
CU09	Mostrar Resultados	Anexo I.1	Anexo I.3

CAPÍTULO VI

Diseño

Este capítulo se centra en la etapa de Diseño, enfocándose en las tres tareas principales. Primero Se especifica el comportamiento por medio de un diagrama de secuencia. Segundo se termina el modelo estático, añadiendo los detalles del diseño en el diagrama de clases. Tercero se verifica que el diseño satisface todos los requisitos identificados.

Arquitectura del Sistema

La arquitectura del framework se compone de los siguientes módulos:

- **Buscador:** el buscador es el primer objeto, el cual muestra las vistas, y recibe peticiones del usuario mediante URL.
- **.httacces**: representa el primer interprete, el cual recibe la ruta ingresada en el buscador por el usuario y la pasa a un archivo especifico.
- **Index**: recibe una ruta del .httacces el cual la convierte a un arreglo de strings y es pasada a boostrap. Objeto
- **bootstrap**: bootstrap y shared se encargan de unificar todo el sistema, identifican el controlador el modelo y las clases necesarias para generar la página que fue solicitada por el usuario.
- **Controlador:** el controlador es el que interactúa entre la vista y el modelo para generar la vista del usuario.
- **Modelo**: el modelo contiene la lógica del software,y genera las conexiones a MysqlDB.

- **Vista**: la vista mayormente integrada de HTML se encarga de mostrar contenido al usuario.
- **MysqlDB**: encargado de gestionar todas las conexiones a la base de datos, de forma segura y eficaz.
- **Herramientas**: este módulo contiene una gran cantidad de herramientas y módulos para facilitar el desarrollo de las aplicaciones.
- **Otras Clases**: otras clases y módulos pueden ser agregados, como gestores de plantillas, y formas, los cuales son cargados automáticamente por bootstrap.

En la figura 26 se muestra el diagrama de flujo de los módulos y como son integrados unos con otros, muestra el procedimiento o enrutamiento que seguirá el framework al recibir la entrada del usuario.

Diagrama de Secuencia

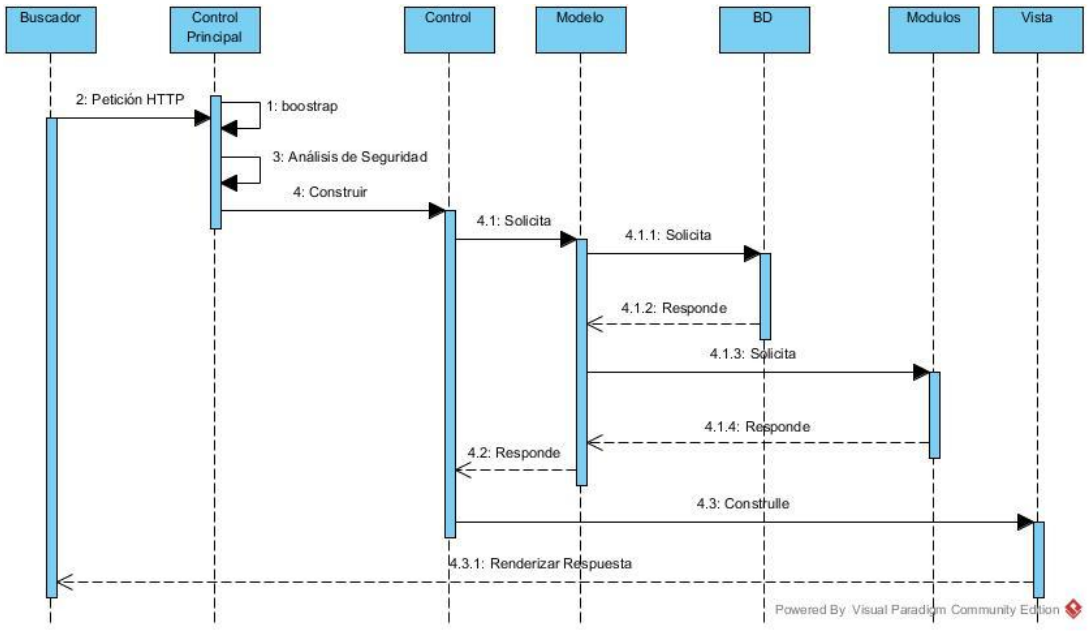


Figura 26: Diagrama de Flujo del framework.

CAPÍTULO VII

Implementación

En este capítulo se describe el funcionamiento obtenido de la implantación del diseño a RaquisFW, se enfoca en el funcionamiento básico del framework y lo implantación de los casos de uso a código.

Modelo Vista Controlador es la vértebra de la aplicación y es usada para separar la lógica de la aplicación y la lógica del negocio. El flujo de datos de una aplicación típica de PHP sin usar algún tipo de framework o MVC se muestra en figura 27 que muestra como PHP funciona como un puente entre el cliente y la base de datos, maneja todo, atiende las entradas de los usuarios, maneja las bases de datos y presenta los resultados. Para reducir la redundancia de la aplicación los programadores generalmente usan la función include() para almacenar código en objetos comunes y archivos externos.

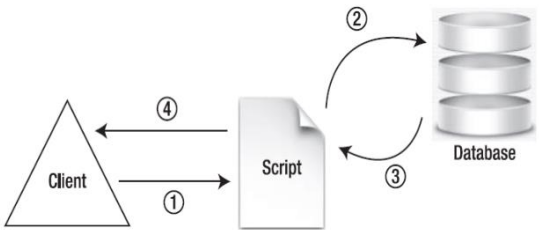


Figura 27: Flujo de una a aplicación típica de PHP

La implementación de MVC mejora el flujo, dividiendo efectivamente este flujo en pequeños pasos y separándolos claramente, asegurándose que cada objeto es escrito una vez y solo una vez. Un diagrama del flujo de la aplicación usando el framework se muestra en la figura 28.

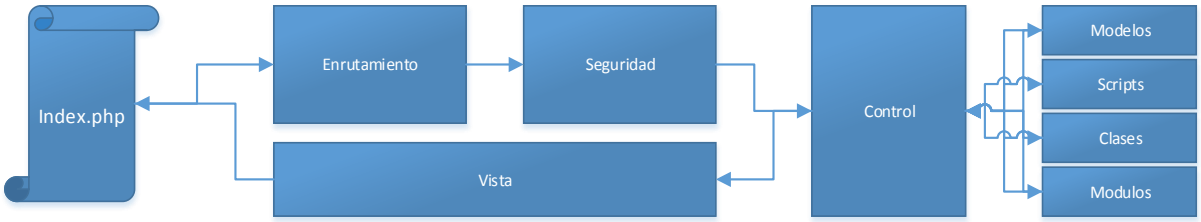


Figura 28: Flujo de Datos del Framework usando MVC

Control Principal

En la figura que muestra el flujo de datos del framework usando MVC, index.php, el control principal el cual inicializa todos los recursos necesitados para correr la aplicación es el primer componente en correr cuando la aplicación inicia. Cuando se recibe una petición de HTTP el router examina y decide que se debe de hacer para poder responder a la petición. Después la petición y cualquier información que esta contenga son examinadas por seguridad antes de que el controlador sea cargado para atender la petición. El controlador carga cualquier recurso necesario para procesar la petición específica y finalmente la vista renderiza una vista en el buscador para el usuario.

Configuración

El archivo de configuración incluye todas las varibles globales necesarias para que la aplicación funcione. Se puede encontrar en primera instancia variables del framework y después de la base de datos, de usuarios, sesiones y los diferentes módulos. Este archivo que se encuentra en app/conf/ es uno de los archivos más importantes de la aplicación ya que sin él, no funcionaría el framework. Se muestra la configuración básica de una aplicación en la figura 29

```php
<?php
/**
* conf.php
* Esta clase es necesaria para especificar todas las
* Diferentes variables para que funcione el programa.
*/


   //-------------------------------------------------\\
  //---------------CONSTANTES GENERALES-----------------\\
//-------------------------------------------------------\\

define('CONTROL_PRINCIPAL',''); //Nombre del controlador principal
define('PLANTILLA',''); // Nombre del Template
define('DOMINIO',''); //nombre de dominio
define('ANVITO_DESARROLLO',false); // si está en desarrollo la aplicación (TRUE o FALSE)
define('RUTA_CLASE', ROOT . DS . 'RaquisFW' . DS . 'classes' . DS);
define('RUTA_MODULOS', ROOT . DS . 'RaquisFW' . DS . 'modulos' . DS);
define('RUTA_TEMPORALES', ROOT . DS . 'app' . DS . 'temporales' . DS);
define('RUTA_PLANTILLA', ROOT . DS . 'RaquisFW'.DS. 'plantillas' . DS . PLANTILLA . DS);
define('RUTA_PLANTILLA_HTML', DOMINIO .'/RaquisFW/plantillas/' . PLANTILLA . '/');

   //-------------------------------------------------\\
  //----------------------SEGURIDAD-------------------\\
//-------------------------------------------------------\\
define('SAL',''); // Sal usada por seguridad.



   //-------------------------------------------------\\
  //-----------------------FAX -----------------------\\
//-------------------------------------------------------\\
define('FAX_USUARIO',''); // usuario.
define('FAX_CLAVE',''); // clave.


  //-------------------------------------------------\\
  //--------------------MENSAJES SMS --------------------\\
//-------------------------------------------------------\\
define('SMS_USUARIO',''); // usuario.
define('SMS_CLAVE',''); // clave.
define('SMS_NUMERO',''); // número que se utiliza.



   //-------------------------------------------------\\
  //------------CONSTANTES DE BASE DE DATOS-------------\\
//-------------------------------------------------------\\
```

```php
define('BD_NOMBRE', ''); //nombre de la base de datos para usar.
define('BD_USUARIO', ''); //usuario
define('BD_CONTRASENA', ''); //clave
define('BD_SERVIDOR', ''); //ip o dominio del servidor de base de datos.


   //-----------------------------------------------------\\
  //--------------CONSTANTES DE SESSIONES----------------\\
//-------------------------------------------------------\\

define("TABLA_USUARIOS", '');//tabla de usuarios
define("CAMPO_ID_USUARIO", '');//compo de usuario
define("CAMPO_USUARIO", '');//compo de usuario
define("CAMPO_CONTRASENA", '');//campo de clave
define("CAMPO_PERMISOS", '');//campo de permisos


   //-----------------------------------------------------\\
  //--------------CONSTANTES DE CORREO---------------------\\
//-------------------------------------------------------\\

define("CORREO_EMISOR", "");//Correo que envía
define("NOMBRE_EMISOR", "");//Nombre que envía
define("CONTRASENA", "");//campo de clave
define("HOST",""); // Establece el dominio del host,
                   //"smtp.gmail.com" es el de gmail
define("PUERTO",465);   // Establece el puerto del servidor
                   //465 el de gmail
define("SMTPDEBUG", 0);// Habilita información SMTP (opcional para pruebas)
                   // 0 = deshabilitado
                                    // 1 = errores y mensajes
                                    // 2 = solo mensajes
define("SMTPAUTH", true); // Habilita la autenticación SMTP
                   // true habilidado
                   // false desabilitado
define("SMTPSECURE", "ssl");  // Establece el tipo de seguridad SMTP

?>
```

Figura 29: Ejemplo de Configuración.

La configuración es guardada en variables globales y pueden ser accedidas en cualquier parte de la aplicación simplemente escribiendo el nombre, por ejemplo si se quiere saber y mostrar el nombre de la plantilla simplemente se usa “echo PLANTILLA;”. Es importante destacar que las variables globales no ocupan ‘$’ y son sensibles a mayúsculas y minúsculas.

Enrutamiento

El enrutamiento es una de las partes esenciales de un framework, este asume responsabilidad sobre la efectividad del mismo. Si el enrutamiento es muy complejo, la curva de aprendizaje crecerá. Comienza en el buscador, generalmente en el buscador indicamos que página necesitamos abrir o solicitar y en caso de que se solicite un directorio index.php/html resuelve la petición. Si la página no existe el servidor muestra una página código 404 el cual indica que la página no existe. En el caso del framework, siempre contesta un archivo llamado .htacces el cual se encarga de alterar el funcionamiento normal o default de un servidor. En la figura 30

se muestra el código utilizado para enviar el url, recibido por el httacces a index.php como POST en una variable URL.

```
<IfModule mod_rewrite.c>
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

RewriteRule ^(.*)$ index.php?url=$1 [PT,L]

</IfModule>
```

Figura: 30 codigo de htaccess

Una vez que index las recibe, anexa algunas variables globales al igual que bootstrap. Las variables anexadas son usadas para identificar los niveles dentro de la aplicación al igual que usar el correcto separador (/ o \) dependiendo del sistema operativo como lo muestra el código de la figura 31.

```php
<?php

/**
* index.php se encarga de recoger todas las entradas por url y mandarlas
* a su controlador.
*/

/* obtener variables globales para ser usado en todo include. */
define('DS', DIRECTORY_SEPARATOR);
define('ROOT', dirname(realpath(__FILE__)));

/* Requerir el url, por medio de $GET, (este viene del htaccess). */
$url = $_GET['url'];

/* Requerir bootstrap. */
require_once (ROOT . DS . 'tutzifw' . DS . 'bootstrap.php');
```

Figura 31: Código de index.php

El MVC implementado en PHP siempre funciona con un URL estructurado de tal manera que busca al controlador, a esto se le llama petición. Por ejemplo tomando el URL **misitio.com/usuario/perfil/1** el nombre de la clase del controlador seria **UsuarioControl.** Seguido del nombre del controlador se muestra la acción o función dentro del controlador que en este caso sería **perfil**. También el URL puede contener otros datos que en este caso sería el 1 el cual será pasado a la función perfil() dentro de UsuarioControl mostrado en la figura 32.

```php
<?php
class UsuarioControl extends Control
{
    function perfil($id) {

    /**
    *
    * Mapear a misitio.com/usuario/perfil/$id
    *
    **/

    }

}
```

Figura 32: Estructura de Clase Usuario

Es necesario que una vista o una salida sea regresada al usuario. Cada método o función por controlador tiene un archivo de vista, este archivo tiene el mismo nombre que la función, así que si el nombre de la función es perfil la vista se llama a perfil.php.

Además de la estructura MVC es necesario conocer como los archivos y folders son organizados dentro del framework. Los tres folders básicos dentro de la aplicación son modelos, vistas, y controladores. Todos los controladores entran en el folder de controladores. Similarmente con los modelos, pero las vistas entran dentro de vistas (folder), en un folder con el nombre del modelo, una vista por cada función, o acción. En el folder de vistas también se colocaran un pie y encabezado mostrado en la figura 33.

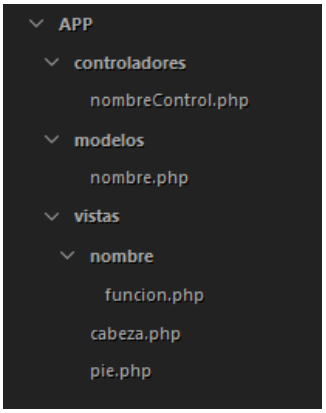


Figura 33: Estructura de una aplicación usando MVC.

Seguridad

La capa de seguridad se encarga de proporcionar seguridad a la aplicación protegiéndola de los ataques y vulnerabilidades más comunes. Esta capa primeramente se encarga de evitar la inyección de código no deseado a la aplicación también conocido como XSS. Todas las peticiones enviadas a la base de datos son verificadas para que no contengan inyección de SQL. De la misma forma las variables de Sesión, Globales, Ámbito, Servidor, y archivos son anuladas, para asegurar que no exista un robo de Sesión.

Controladores

La capa de Control se encarga de responder a todas las peticiones del usuario. Es responsable de renderizar una respuesta con la ayuda de la vista y el modelo. El Control se puede ver como un supervisor que se asegura que todas las respuestas necesarias para cumplir un trabajo son delegadas al trabajador correcto. Espera peticiones de los clientes, verifica su valides de acuerdo a reglas de autentificación y seguridad, delega la obtención de información o proceso de información al modelo, selecciona el tipo de presentación que los clientes solicitan, y finalmente delega el trabajo de renderizacion a la vista.

Despues de que el ruteo es aplicado y el controlador correcto es encontrado, la acción de este controlador es llamada. El control se debe de asegurar de que la información es interpretada, llamar al modelo para procesar la petición, y que la vista muestre una respuesta. Los controles funcionan como un puente entre el Modelo y la Vista. Estos deben de ser tan pequeños como sea posible, y los modelos extensos. Esto ayuda a la reutilización de código y facilitar la resolución de errores.

Control Principal

Cada control creado extiende al control principal, el cual se encuentra en /RaquisFW/class/control.class.php que contiene métodos o funciones que todo control debe heredar para facilitar la comunicación con el modelo y la vista. El código fuente del Control Base o Principal, muestra las funciones básicas que cada control hereda para facilitar la comunicación con el modelo la vista y las plantillas ver figura 34.

```php
<?php
/**
* Esta clase es la base para, todos los controles de la aplicacion.
* @author Daniel Lozano Carrillo <daniel@unav.edu.mx>
* @version 0.2
* @license MIT
*/

class Control {

    protected $_modelo;
    protected $_control;
    protected $_accion;
    protected $_plantilla;

    /**
     * Constructor de la clase, recibe variables y contrullo la aplicacion
     * principal
     * @param String $modelo  el nombre del modelo
     * @param String $control el nombre del control
     * @param String $accion  la accion princial.
     */
    function __construct($modelo, $control, $accion) {

        $this->_accion = $accion;
        $this->_control = $control;
        $this->_modelo = $modelo;
        $this->$modelo = new $modelo;
        $this->_plantilla = new Plantilla($control,$accion);

    }


    /**
     * Set es usado para asignar variables a vistas.
     * @param String $nombre  el nombre de la variable.
     * @param String $valor el valor de la variable.
     */
    function set($nombre,$valor) {
        $this->_plantilla->set($nombre,$valor);
    }


    /**
     * Mandar a renderizar la aplicacion antes de terminar.
     */
    function __destruct() {
        $this->_plantilla->render();
    }

}
```

Figura 34. Código del Control Base del Framework.

Convenciones

Las convenciones forman una parte importante de un framework siguiendo un principio importante convenciones sobre configuración, seguir estas convenciones es importante para el correcto funcionamiento del mismo.

- El nombre del archivo es el nombre del modelo más control, ejemplo: si el modelo se llama Admin el nombre del archivo seria admincontrol.php.
- Todos los controles se encuentran en el directorio /app/controladores/
- El nombre de la Clase es el nombre del modelo más Control en CamelCase.
- Cada clase debe de extender a Control (el control principal).
- El control debe de tener la acción o función index() la cual es la acción principal, es llamada en case de que no exista alguna otra acción.

Las Acciones

Los Controles ofrecen funciones para interpretar las peticiones recibidas. Estas se llaman acciones por default, cada función pública es una acción. Estas funciones pueden ser mapeadas desde una URL. Cada acción es responsable de interpretar la petición y generar una respuesta. Generalmente esta respuesta consiste en la renderizacion de la vista.

Cada controlador tiene por lo menos la acción o función index() la cual es llamada en caso de que solo el controlador sea llamado en la petición de HTTP. En la figura 35 se muestra el código fuente de nombrecontrol.php el cual se encuentra en su forma más simple y tiene solo una acción index.

```php
<?php

class NombreControl extends Control{

    function index(){

          //MAPEO: DOMINIO/nombre/
                    }
    }
```

Figura 35: Código de un control básico

Variables y atributos de acciones

Cada acción puede recibir variables o atributos los cuales son determinados en el URL, estas son recibidas por las acciones y tienen que ser especificadas en la función. La figura 36 muestra el controlador mostrado anterior con una nueva acción llamada bienvenida, la cual recibe un usuario de la petición de HTTP. Si el usuario no es especificado, la acción determinar que el usuario es “Invitado”. La acción recibe el usuario y asigna un mensaje a la variable $mensaje.

```php
<?php

class NombreControl extends Control{

    function index(){

          //MAPEO: DOMINIO/nombre/
    }

    function bienvenida($usuario="Invitado"){
        //MAPEO: DOMINIO/nombre/bienvenida/usuario

        $mensaje="Te damos la bienvenida a nuestro sitio”. $usuario;
    }
}
```

Figura 36: Recibiendo variables desde el URL

Interacción con el Modelo

El propósito del control no es generar la lógica de la aplicación si no conectar y supervisar que se atienda la petición recibida del usuario. Por lo tanto la interacción con

el modelo es importante. El mensaje generado por el control no pertenece en el control y debe de ser mandado al modelo donde existe la lógica de la aplicación ver Figura 36. Para llamar a alguna función del modelo solamente llamamos el nombre del modelo. Ejemplo, si la función para generar el mensaje en el modelo se llama mensajeBienvenida() y recibe un usuario, se mandaria a llamar con $this->Nombre->mensajeBienvenida($usuario), donde nombre es el nombre del modelo. Con $this->Nombre podemos acceder a todas las funciones y variables públicas del modelo. En la figura 37 se muestra de nuevo el Control Nombre, con la acción, bienvenida, pero esta vez permite que el modelo genere el mensaje de bienvenida.

```php
<?php

class NombreControl extends Control{

    function index(){

          //MAPEO: DOMINIO/nombre/
    }

    function bienvenida($usuario="Invitado"){
        //MAPEO: DOMINIO/nombre/bienvenido/usuario

        $mensaje=$this->Nombre->mensajeBienvenida($usuario);

    }
}
```

Figura 37. Interacción del Control con el Modelo

Interacción con la Vista

Una vez se obtiene la respuesta a la petición del usuario el control debe de mandarlas a la vista, está la renderiza en una página con estilos proporcionados por los diseñadores. La comunicación existe mediante una función en el Controlador Principal mostrado en la figura 38. La función se llama set() y recibe dos variables la primera es te tipo String y la segunda puede ser cualquier tipo de dato, String, Int, Double, Array, o un Objeto. La

primera variable determina el nombre de la variable en la vista que contendrá el contenido especificado en la segunda variable. En la Figura 38 muestra como la acción bienvenida que recibe una variable $usuario desde una URL, manda a llamar al Modelo para obtener un mensaje y se lo asigna a una variable mensaje para la vista.

```php
<?php

class NombreControl extends Control{

    function index(){

          //MAPEO: DOMINIO/nombre/
    }

    function bienvenida($usuario="Invitado"){
        //MAPEO: DOMINIO/nombre/bienvenido/usuario

        $this->set("mensaje", $this->Nombre->mensajeBienvenida($usuario));

    }
}
```

Figura 38: Interacción del Control con la Vista

Modelos

Los modelos están encargados de gestionar toda la lógica de la aplicación, por lo tanto, son donde el desarrollador más trabaja. Estos modelos gestionan todas las conexiones a la base de datos. Pueden hacer uso de los diferentes módulos que se encuentran en el framework.

Modelo Principal

El modelo principal es aquel que se encuentra en /RaquisFW/class/modelo.class.php. Este modelo contiene funciones básicas para realizar conexiones a la base de datos y

obtener y mandar información. Cada modelo creado en app/modelos necesita extender o heredar al modelo principal. En la figura 39 se muestra el código fuente del Modelo principal.

```php
<?php
class Modelo extends MySQLDB{
    protected $_modelo;

    function __construct() {
        parent::__construct();
    }

    function __destruct(){


    }
}
```

Figura 39: Modelo principal

Convenciones

- El nombre del archivo es el nombre del archivo es el nombre del control nombre.php.
- Todos los modelos se encuentran en el directorio /app/modelos/
- El nombre de la clase es el nombre del control con la primera letra en mayúscula.
- Cada clase debe de extender a Modelo (el modelo principal).

Conexión a la base de datos

El modelo hereda la Clase MySQLDB y se encarga de generar las conexiones a la base de datos. Al heredar a la clase MySQLDB hereda el funcionamiento de la base de datos, y es posible llamar sus funciones, como si fueran una misma clase. MySQLDB contiene funciones que facilitan la comunicación con la base de datos de MySQL. La

clase MySQLDB tiene funciones que facilitan la obtención de recursos de la base de datos, proveyendo funciones fáciles de entender. Las funciones más importantes son query(), insertarxArray() y actualizarxArray(). La función query(), permite mandar código SQL a la base de datos, para insertar, actualizar, seleccionar o cualquier tipo de proceso permitido por SQL. insertxArray() y actualizarxArray() reciben un array de datos que convierten a código SQL para poderlo usar la función query para ser insertada en la base de datos.

Vistas

La vista contiene la presentación final del contenido, como el HTML, CSS, y JavaScript. Estas vistas deben de contener la menor cantidad de PHP posible ya que debe de ser fácil para los diseñadores de interfaces y diseñadores web trabajar con las vistas sin conocer PHP, esta es una de las ventajas de usar MVC.

Convenciones

- Cada función en los controladores debe de tener una vista con el mismo nombre que la función.
- Las vistas son guardas en la carpeta /app/vistas/NombreDelControl/.
- Dentro del folder /app/vistas/ debe de existir, encabezado.php y pie.php estos archivos son llamados al princio y final de cada vista.
- Se puede crear un encabezado.php y pie.php dentro de /app/vistas/NombreDelControl/ y este sería usado en lugar del que se encuentra en /app/vistas/.

Motor de Plantillas

El framework cuenta con un motor de plantillas, con el pueden crearse plantillas reutilizables para agilizar el desarrollo de aplicaciones, enfocándose solo en la lógica de la aplicación, usando código PHP para generar el HTML, CSS y JavaScript. Para agregar platillas al framework estas tiene que se guardados en /RaquisFW/plantillas/ en un directorio con el nombre de la plantilla y con el contenido de la plantilla adentro. En este mismo directorio se debe de incluir un folder llamado PHP que incluya tres archivos, estructura.class.php, tema.class.php y modulos.class.php. Estos archivos implementaran tres interfaces encontradas en /RaquisFW/plantillas/ las cuales llevan en mismo nombre. Al implementar estas interfaces, crea un estándar, la cual cada nueva plantilla debe seguir para ofrecer una misma funcionalidad.

El motor de plantillas accede desde cualquier parte del framework ya sea la vista, el modelo o el controlador. La figura 40 muestra un ejemplo de cómo se genera, una nueva plantilla. En este caso la plantilla es implementada en el controlador, y se encuentra en el modelo ver figura 41. Para construir la plantilla es necesario especificar las variables de la plantilla y al llamar la función renderizar() la cual regresa el html, css y javascript que el gestor genera. El resultado es mandado a la vista por medio de dos variables $cabeza y $pie, los cuales son impresos antes y después de la vista principal usando encabezado.php y pie.php en /app/vistas/ como se muestra en las figuras 42 y 43.

```
1   <?php
2
3   class NombreControl extends control{
4
5       function index(){
6           $this->Nombre->plantilla->titulo = "Inicio";
7           $this->Nombre->plantilla->ruta = Array(
8                                                   Array("#", "Admin"),
9                                                   Array("#", "Inicio"),
10                                                  Array("#", "Fin")
```

```
                                                );
        $this->Nombre->plantilla->titulo=$this->titulo;
        $this->Nombre->plantilla->ruta=$this->ruta;
        $this->Nombre->plantilla->nombreApp="Registro";
        $this->Nombre->plantilla->skin="skin-blue";
        $this->Nombre->plantilla->piedePagina="Copyright &copy; 2014-2015 <a
href='#'>TutziLAbs</a>.</strong> Todos los derechos reservados.";
        $this->Nombre->plantilla->busqueda="";
        $this->Nombre->plantilla->datos_usuario=Array(
            "imagen"=>RUTA_PLANTILLA_HTML."dist/img/user2-160x160.jpg",
            "usuario"=>"Nombre de Usuario",
            "desc1"=>"descripcion 1",
            "desc2"=>"descripcion 2",
            "link1"=>"link 1",
            "link1url"=>"#",
            "link2"=>"link 2",
            "link2url"=>"#",
            "link3"=>"link 3",
            "link3url"=>"#",
            "perfil"=>"#",
            "logout"=>DOMINIO."/admin/salir"
            );
            //tipos -> separador,sensillo,multimple
        $this->Nombre->plantilla->menu=Array(
                Array("separador","Separador"),//separador
                Array("sensillo","#","fa fa-book","Menu1"), //menu sensillo
                Array("multiple","#","fa fa-book","Menu2", //multinivel
                    Array(
                        Array("ponente","fa fa-book","SubMenu1"),
                        Array("ponencias","fa fa-book","SubMenu2"),
                        Array("#","fa fa-book","SubMenu3"),
                        Array("horarios","fa fa-book","SubMenu4"),
                        Array("lugar","fa fa-book","SubMenu5")
                         )
                    ),
                Array("separador","Menu3"), //separador
                Array("sensillo","boletos","fa fa-
book","Separador2"),//sensillo
);
        $this->Nombre->plantilla->notificaciones=Array(Array("#","Nuevo
mensaje."),
                                                Array("#","Tiene un nuevo
mensaje."),
                                                Array("#","Otro mensaje."),
                                                );
        $this->set("cabeza", $this->Nombre->plantilla->renderizar());
        $this->set("pie", $this->Nombre->plantilla->pie());

    }
```

Figura 40: Creación de una plantilla usando el gestor de plantillas

```
<?php
class Nombre extends Modelo
{
```

```
4       public $plantilla;
5
6       public function __construct(){
7           parent::__construct();
8           $this->plantilla = new Tema;
9       }
10  }
```

Figura 41: Inicializando una plantilla en el modelo.

```
1   <?php
2   echo $pie;
3   ?>
```

Figura 42: pie.php, imprime variable recibida del controlador.

```
1   <?php
2   echo $cabeza;
3   ?>
```

Figura 43: encabezado.php, imprime variable recibida del controlador

4Los módulos son parte fundamental del framework, proveen al desarrollador de distintas librerías organizadas de tal manera que tenga acceso a ellas de cualquier parte de la aplicación. Cada módulo es una libreria estructura para ser reutilizable en cada proyecto. Todos los módulos deben der ir en /fw/modulos/.

Módulos Existentes

El framework cuenta con los siguientes módulos.

- Archivo: dedicado a la gestión de archivos, permite verificar el nombre, tamaño y tipo de archivo, guardarlo o descargarlo.
- Catálogos: gestor de Catalogos, permite generar formularios y tablas para editar, borrar, e insertar automáticamente por cada tabla en la base de datos.
- Correos: se encarga de gestionar los correos, puede mandar correos simples, con datos adjuntos y por medio de SMPT.
- Fax: gestiona el envió de Fax desde PHP.
- Formas: gestor de formularios.

- Lenguaje: permite convertir a la aplicación en aplicaciones multilenguaje mediante el uso de archivos .ini, los cuales pueden ser traducidos globalmente.
- PDF: genera PDF, los cuales pueden ser guardados o descargados puede generarlos recibiendo un HTML o desde un URL.
- QR: genera códigos QR.
- Sesión: gestiona las sesiones de la aplicación.
- SMS: permite mandar mensajes de texto mediante PHP.
- Usuarios: gestiona los usuarios.
- Utilerías: diferentes herramientas útiles para el desarrollador.
- Validar: Validador de información, puede validar, números, ips, macs, correos etc.

Nuevos Módulos

Es recomendable que conforme el desarrollador avance en diferentes proyectos, pueda convertir código reutilizable en módulos para poder ser usados en futuros proyectos. Para crear un nuevo módulo es necesario definir el tipo de clase que se usara, una clase normal o de tipo estática. Las ventajas de las clases estáticas es que no tienen que ser inicializadas y pueden ser llamadas simplemente mediante el nombre del módulo por ejemplo para mandar un correo simple, llamamos a Correo::simple(); esto facilita el uso de los módulos.

Cada módulo está compuesto de la configuración, la clase y los archivos o librerías necesarias para que el modulo funcione. La configuración, debe de ser agregada a /app/configuracion/configuración.php este archivo contiene la configuración de cada

módulo y del framework en general. La clase debe de ser guardad en /RaquisFW/modulos, debe de guardarse como nombredelmodulo.mod.php y por último los archivos deben de ser guardados en una carpeta con el nombre del módulo, y todos los archivos dentro de ella.

CAPÍTULO VIII

Convenciones y Buenas Prácticas

Es importante que el framework, siga convenciones y buenas prácticas para poder mantener una estructura y formato único que cualquier programador pueda entender con facilidad, para que todos estén en la misma página. En este capítulo se proponen convenciones y buenas prácticas en diferentes puntos como es la estructura de los archivos, estilo de programación, documentación, y control de versiones.

> "Drupal es un Administrador de Contenido de Software Libre. En su desarrollo y mantenimiento participan diseñadores, programadores y colaboradores en general, de todas partes del mundo. Para poder lograr que todos los aportes se den en forma ordenada y minimizar los problemas cuando alguien debe tomar el proyecto de otro para corregirlo o mejorarlo, se ha definido ciertas buenas prácticas, que toda la comunidad de Drupal sigue con el fin de facilitar las cosas." (sanchez, 2011)

Estructura de los archivos

Una buena estructura de archivos facilita la igualdad de estos en diferentes ambientes, permitiendo que todos los desarrolladores tengan la misma experiencia sin importar su pantalla, sistema operativo o el contexto en el que trabajan.

- Para los archivos que solo contengan código PHP la etiqueta para cerrar php (?>) nunca es permitida. No es requerida por PHP y evitarla ayuda a evitar la inserción de espacios en blanco.
- La sangría de los archivos deben de ser de 4 espacios, sin usar tabulador.
- Cada línea debe de ser menor de 80 caracteres, el desarrollador debe de hacer todo lo posible para que así sea en caso no ser posible el tamaño máximo, es de 120 caracteres sin excepciones.

- Una sola clase por archivo.
- Declarar visibilidad de cada variable y función.

Convenciones de Nombres

El estilo de escritura que se aplica a frases o palabras compuestas son bajo la siguientes convenciones de nombres.

- Los nombres de las clases siempre tiene que ser en base al estilo CamelCase, con la inicial en mayúscula.
- Los archivos deben de ser nombrados alfanuméricos, en minúsculas, sin espacios.
- Variables, propiedades, funciones y métodos deben de ser en CamelCase, con la inicial en minúscula.
- Las CONSTANTES siempre tiene ser nombrados en mayúsculas.
- Propiedades y métodos privados deben de usar un guion bajo al principio.

Estilo de Programación

Las convenciones para escribir el código fuente es bajo la siguente convención.

- Las funciones y clases deberán tener su llave inicial en la siguiente línea a la declaración y en el mismo nivel de indentación.
- Las estructuras de control abren llaves iniciales en la misma línea de la estructura, y se deja un espacio después del control para diferenciarlo de una función.
- Todas las estructuras de control deben incluir llaves, sentencias inline no son permitidas.

- Strings que no contengan variables deben de ser encerradas en comillas sencillas.
- Si el String contiene comillas sencillas se encierra en doble comillas.
- Strings deben de ser concatenados con el operador "." y un espacio en blanco debe ser dejado.

En la figura figura 44 se muestra un ejemplo de las buenas prácticas, y estilo de programación descrito previamente.

```php
<?php //permitido
<?    //no permitido

//STRINGS
$a = 'Texto de Ejemplo'; //permitido
$a = "Texto de Ejemplo"; //no permitido

$universidad = 'Universidad' . ' ' . 'Navojoa'; //permitido
$universidad='Universidad'.''.'Navojoa';        //no permitido

$sql = "SELECT `id`, `name` FROM `people` "
     . "WHERE `name` = 'Susan' "
     . "ORDER BY `name` ASC ";             //permitido

$sql = "SELECT `id`, `name` FROM `people` WHERE `name` = 'Susan' ORDER BY
`name` ASC "; //no permitido

$greeting = "Hola $usuario, bienvenido";   //permitido
$greeting = "Hola {$usuario}, bienvenido"; //permitido
$greeting = "Hola ${usuario}, bienvenido"; //no permitido

//ARRAYS
$sampleArray = array(
    1, 2, 3, 'Tutzi', 'Labs',
    $a, $b, $c,
    56.44, $d, 500,
); //permitido

$sampleArray = array(
    'primerallave'  => 'primervalor',
    'segundallave' => 'segundovalor',
); //permitido

$sampleArray = array(1,2,3,'Tutzi','Labs',$a,$b,$c,56.44,$d,500,); //no
permitido
```

```
34
35
36
37  ?> // no es necesario
```

Figura 44: Estilo de programación de PHP.

Documentación

Todos los bloques de documentación, deben de ser compatibles con el formato de phpDocumentor, el cual se ha convertido en un estándar. Cada archivo debe de contener, un bloque de documentación como se muestra en la figura 45, al igual que cada función y clase los cuales se muestra 46.

```
1   <?php
2   /**
3   * Pequeña descripción del archivo.
4   *
5   * Descripcion larga (si existe)...
6   *
7   * LICENCIA: Informacion sobre la licencia
8   *
9   * @category   Categoria
10  * @package    Paquete
11  * @subpackage sub_paquete
12  * @copyright  Derechos de author. (URL)
13  * @license    URL de la licencia   Nombre de la Licencia
14  * @version    Version
15  * @link       URL del archivo
16  * @since      Cuando se creo
17  */
```

Figura 45: Bloque de documentación de un archivo.

```
1   <?php
2   /**
3   * Pequeña descripción de la clase.
4   *
5   * Descripcion larga (si existe)...
6   *
7   * LICENCIA: Informacion sobre la licencia
8   *
9   * @category   Categoria
10  * @package    Paquete
11  * @subpackage sub_paquete
12  * @copyright  Derechos de author. (URL)
13  * @license    URL de la licencia   Nombre de la Licencia
14  * @version    Version
15  * @link       URL del archivo
16  * @since      Cuando se creo
```

```
17    * @deprecated Funcionamiento depricado.
18    */
```

Figura 46: Bloque de documentación para una clase.

CAPÍTULO IX

Conclusiones y Trabajos Futuros

Conclusiones

El framework cumple con los requisitos funcionales especificados en los requerimientos y está listo para ser implementado en nuevos proyectos de desarrollo. La culminación de este proyecto da el inicio al desarrollo continuo del framework. Es de importancia seguir con el desarrollo conforme se utilice para la creación de nuevo software, siempre tratando de implementar nuevos módulos. La modulación de los proyectos agilizara la construcción de nuevo software en un futuro. La curva de aprendizaje del software puede ser algo alta pero el poder comprender su arquitectura es clave para dominar la curva de aprendizaje, cuando esto tome lugar el framework dará sus frutos y será posible usar y aportar a su desarrollo.

El desarrollo de este proyecto me ayuda a entender conceptos complejos del desarrollo de software como los de arquitectura, seguridad, patrones de diseño, rendimiento, ejecución, calidad, programación orientada a objetos, entre otros. Tener este conocimiento me ayudo a satisfactoriamente completar varias entrevistas de trabajo y poder obtener las posiciones que más me gustaron en algunas de las mejores empresas de México. Cada concepto cubierto en esta tesis es sumamente importante para el desarrollo de software en un ámbito empresarial, haciendo importante su entendimiento para cualquier persona que se quiera dedicarse al desarrollo de software.

Trabajos Futuros

El propósito del framework es crecer y evolucionar conforme sea usado para desarrollar distintas aplicaciones. Conforme fue desarrollo se encontraron algunas funcionalidades que pueden ser agregadas en un futuro, las cuales son listadas a continuación.

- Realizar pruebas para determinar si se cumple el propósito del framework.
- Determinar la curva de aprendizaje, y encontrar formas de reducirla.
- Probar el desempeño de aplicaciones que utilicen este framework, para lograr mejorar el rendimiento.
- Crear un gestor de cache, que sirva para responder peticiones, repetitivas, y estáticas.
- Analizar si es conveniente implementar la nueva arquitectura de desarrollo HMVC (Modelo Vista Controlador Jerárquico) que es una evolución de MVC la cual permite modular los modelos vistas y controladores.
- Incorporar un sistema de excepciones, que sea fácil de utilizar, y mejore la funcionalidad del que ya existe en PHP.
- Implementar ActiveRecord a los modelos para facilitar las consultas a la base de datos.
- Crear un módulo de Ajax, que permita interactuar con PHP dinámicamente.

ANEXOS

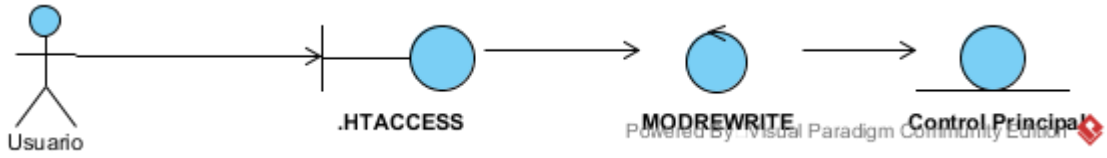
ANEXO A

Caso de Uso : Capturar Petición

Anexo A.1 Capturar Petición Descripción

Caso de uso	Capturar Petición
Resumen	Usando .HTACESS y el control principal, se interceptara la petición de HTT¨P enviada por el usuario, para poder construir el control de la aplicación.
Actor principal	Control Principal
Precondiciones	El usuario debió ingresar una petición en el buscador.
Flujo principal	1. El usuario ingresa una URL en el buscador, 2. .htaccess la recibe. 3. Usa MODREWRITE para mandarla a index.php como string. 4. El control principal la procesa, y la convierte a un array.
Flujo Alternativo	Si el URL es un archivo, no intercepta la petición y deja que APACHE maneje la petición (eje. Dominio.com/stilo.css).

Anexo A.2 Capturar Petición Diagrama



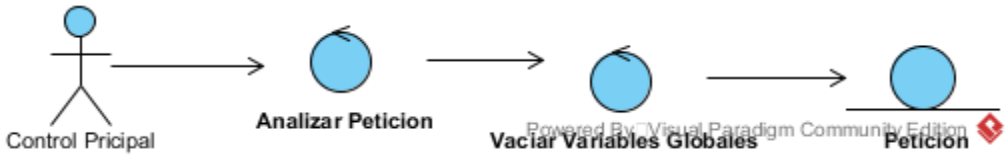
ANEXO B

Caso de Uso : Analisis de Seguridad

Anexo B.1 Análisis de Seguridad Descripción

Caso de uso	Análisis de Seguridad
Resumen	Antes de que el Control Principal envié la petición del usuario al control, analiza la petición para asegurar que no explote alguna vulnerabilidad común de los servidores web.
Actor principal	Control Principal
Precondiciones	
Flujo principal	1. Recibe la petición de HTTP. 2. Analiza su contenido, para verificar que no exista XSS. 3. Vacía las variables globales de PHP para asegurar que no exista un robo de sesión.
Flujo Alternativo	

Anexo B.2 Análisis de Seguridad Diagrama



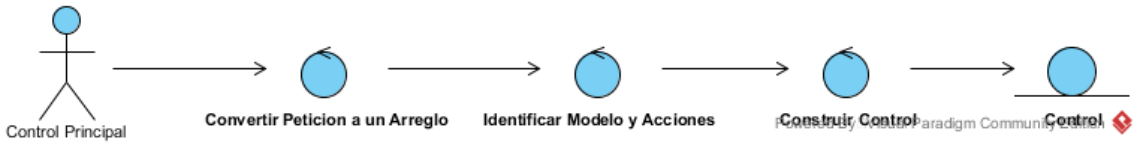
ANEXO C

Caso de Uso : Cosntruir Control

Anexo C.1 Construir Control Descripción

Caso de uso	Construir Control
Resumen	El Control principal convierte la petición en un Array de Strings,y usando los distintos valores identifica el control, el modelo, y las acciones y construye el control necesario para responder a la petición del usuario.
Actor principal	Control Principal
Precondiciones	La petición de HTTP debió de ver pasado la capa de seguridad.
Flujo principal	1. Convertir petición en Array. 2. Identificar Control, Modelo, y Acciones. 3. Construir el Controlador.
Flujo Alternativo	• Si la petición, no contiene un control, se usa el control principal especificado en la configuración. • Si la petición no tiene una Acción, se tomara index, como la acción.

Anexo C.2 Construir Control Diagrama



ANEXO C

Caso de Uso : Contruir Modelo

Anexo D.1 Construir Modelo Descripción

Caso de uso	Construir Modelo
Resumen	El control de la aplicación es construido, y recibe la acción, el modelo, y otras variables. Usando estos datos, el control construye al modelo, necesario para procesar la solicitud del
Actor principal	Control
Precondiciones	
Flujo principal	1. El control identifica el Modelo necesario y lo
Flujo Alternativo	

Anexo D.2 Construir Modelo Diagrama

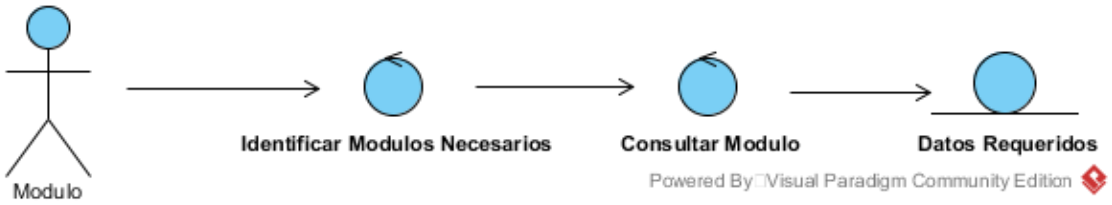
ANEXO E

Caso de Uso : Consultar Módulos

Anexo E.1 Consultar Módulos Descripción

Caso de uso	Consultar Módulos
Resumen	Dependiendo de la necesidad de datos del controlador el Modelo, consulta los diferentes módulos para poder responder a las peticiones.
Actor principal	Modelo
Precondiciones	
Flujo principal	1. El Modelo identifica el modulo necesario para procesar la petición recibida del Control. 2. Consulta al módulo y obtiene los datos requeridos.
Flujo Alternativo	Si el modulo no existe, muestra un error.

Anexo E.2 Consultar Módulos Diagramas



ANEXO F

Caso de Uso : Analisis de Seguridad

Anexo F.1 Consultar Base de Datos Descripción

Caso de uso	Consultar Base de Datos
Resumen	El Modelo consulta la base de datos para obtener o mandar cualquier información necesaria.
Actor principal	Modelo
Precondiciones	
Flujo principal	1. El Modelo hereda al Modelo Principal, el cual hereda al MySQLDB. 2. Identifica el tipo de consulta que va a realizar. 3. Ejecuta el query en la base de datos. 4. Regresa el resultado.
Flujo Alternativo	

Anexo F.2 Consultar Base de Datos Diagrama

ANEXO G

Caso de Uso : Analisis de Seguridad

Anexo G.1 Generar Respuesta Descripción

Caso de uso	Generar Respuesta
Resumen	Usando los datos obtenidos de los módulos y la base de datos generar una respuesta.
Actor principal	Modulo
Precondiciones	• Obtener información de la base de datos. • Obtener datos de los módulos.
Flujo principal	1. Generar respuesta al Control usando la información de los módulos y la base de datos.
Flujo Alternativo	

Anexo G.2 Generar Respuesta Diagrama

ANEXO H

Caso de Uso : Analisis de Seguridad

Anexo H.1 Construir Vista Descripción

Caso de uso	Construir Vista
Resumen	Usando los datos obtenidos de Modelo, asignar variables para que puedan ser accedidas por la Vista.
Actor principal	Control
Precondiciones	• Haber obtenido una respuesta del Modelo.
Flujo principal	1. Control recibe respuesta del Modelo. 2. Asigna Variables con los datos recibidas para que puedan ser accedidas desde la vista.
Flujo Alternativo	

Anexo H.2 Construir Vista Diagrama

ANEXO I

Caso de Uso : Analisis de Seguridad

Anexo I.1 Mostrar Resultados Descripción

Caso de uso	Mostrar Resultados
Resumen	Usando las variables obtenidas del controlador mostrar una respuesta al Usuario.
Actor principal	Vista
Precondiciones	.
Flujo principal	1. Recibe datos del controlador. 2. Carga los CSS, HTML, JavaScript. 3. Muestra un resultado al usuario.
Flujo Alternativo	

Anexo I.2 Mostrar Resultados Diagrama

ANEXO O

CURRICULUM VITAE

Daniel Lozano Carrillo

Personal Data

Place and Date of Birth:	Nogales, Sonora MX | December 02 1992
Address:	Senderos de las Secuoyas 19, Tlaquepaque, Jal.
Phone:	+1 686 288 1684
email:	mail@medanny.com

Work Experience

Current – Apr 16

Full-Stack Software Developer @ Persistent Systems

Rational Doors Next Generation

Develop and maintain components based on requirements. Solve and debug defects and client escalations. Write unit test and documentation for components. Automate and improve build installations after deployment in different environments. Technologies used include Java, JavaScript, Dojo, HTML, CSS, Linux, Docker, Shell Scripting, Eclipse, JUnit, Web Services, RTC, RQM, RDNG.

Jul 15- Mar 16

Software Developer @ IBM

Rational Doors Next Generation - Rational Quality Manager

Develop and maintain components based on requirements. Solve and debug defects and client escalations. Write unit test and documentation for components. Automate and improve build installations after deployment in different environments. Technologies used include Java, JavaScript, Dojo, HTML, CSS, Linux, Docker, Shell Scripting, Eclipse, JUnit, Web Services, RTC, RQM, RDNG.

May 15 - Jul 15

Software Developer @ GameLoft

Develop and maintain Ecommerce shops across the world using PHP, Javascript, MySQL, XML and Ajax.

Nov 2014- May 2015

Freelancer

Accounting, Inventory, and HR Systems for LoneStar Corporation

Design and develop information systems to meet the clients requirements using, PHP, Javascript, HTML, CSS, LINUX, MySQL, Shell Scripting.

Jul 2011- Oct 2014

Software Developer @ Universidad de Navojoa

University Information Systems

Design, develop and maintain information systems across different requirements leading a team of 4 people using Java, JSP, TomCat, Oracle DBA, HTLM, CSS, .

University Home Page

Design and develop the University Website along with many components, using Wordpress, PHP, MySQL, Javascript and JQuery.

University Servers and Network Infrastructure

Analyze and improve network infrastructure performance. Implement a network Firewall using Endian. Implement a network load balancer and Captive Portal using PFSense. Create a Server Infrastructure for Information Systems using Linux, Bash, VestaCP, Apache, TomCat, MySQL, Oracle DBA, Nginx, bind9, MySQL. Create a redundant and economical back up system using Perl, Shell Scripting, and LINUX.

Aug 09 - May 11

Web Developer @ Amphitheater High School

Develop and maintain a web page for Gallery 125 with interactive photo galleries capable of managing and interacting with different kinds of media using PHP, HTML, CSS, and JavaScript.

Education

May 2015 Computer Systems Engineer, **Universidad de Navojoa**, Sonora, MX
Major: Programing, Multimedia, and Networking
Gpa: 93.65/100

Languages

Spanish: Fluent
English: Fluent

Computer Skills

Basic Knowledge: C++, Ruby, LESS, Oracle DBA, CakePHP, Docker, VMWare
Advance Knowledge: JAVA, PHP, JavaScript, JSP, DOJO, HTML, CSS, Shell Scripting
Linux, JUnit, MySQL, Eclipse, AJAX, JQuery, SQL, VPS, GIT

ANEXO P

Código Fuente

https://github.com/medanny/raquisFW

REFERENCIAS

Adobe. (2014). *Aspectos básicos de las aplicaciones Web*. Obtenido de Aspectos básicos de las aplicaciones Web: http://helpx.adobe.com/es/dreamweaver/using/web-applications.html

Alegsa, L. (s.f.). *diccionario de informática y tecnología*. Obtenido de Definición de UML: http://www.alegsa.com.ar/Dic/uml.php

Alegsa. (13 de Septiembre de 2009). *¿Qué es una Utilería (informática)?* Obtenido de http://www.alegsa.com.ar/Diccionario/C/4667.php

Álvarez, M. A. (01 de Enero de 2001). *Qué es HTML*. Obtenido de Desarrolloweb.com: http://www.desarrolloweb.com/articulos/que-es-html.html

Alvarez, M. A. (2008). *desarrolloweb*. Obtenido de Que es un CMS: http://www.desarrolloweb.com/articulos/que-es-un-cms.html

Caro, P. S. (s.f.). *Tutorial de UML*. Obtenido de Tutorial de UML: http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html

Carrero, F. B. (2011). *glosarium.com*. Obtenido de Diccionario Informático: http://glosarium.com/

Cunningham & Cunningham Inc. (2014, Julio 8). *Cunningham & Cunningham Inc.* Retrieved from Model View Controller History: http://c2.com/cgi/wiki?ModelViewControllerHistory

David Carrero Fernandez-Baillo. (2011). *glosarium.com*. Obtenido de Diccionario Informático: http://glosarium.com/

Definicion.de. (2015). *Definicion.de*. Obtenido de Estructura: http://definicion.de/estructura/

DocForge. (20 de Junio de 2014). *DocForge*. Obtenido de Software Development Resources: http://docforge.com/wiki/Web_application_framework

Doug Rosenberg, M. C.-C. (2005). *Agile Development with ICONIX Process: People, Process, and Pragmatism.* Apress.

Ecured. (2015). *Herramientas informáticas*. Obtenido de http://www.ecured.cu/index.php/Herramientas_inform%C3%A1ticas

EllisLab. (2015). *ellislab*. Obtenido de A Brief History of CodeIgniter: https://ellislab.com/codeigniter

eNubes. (2014). *eNubes*. Obtenido de Desarrollo de aplicaciones web.

GitHub, Inc. (2014). *GitHub*. Obtenido de GitHub Features: https://github.com/features

Gravelle, R. (12 de Enero de 2011). *Top 10 MySQL Best Practices.* Obtenido de Database Journal: http://www.databasejournal.com/features/mysql/article.php/3918631/Top-10-MySQL-Best-Practices.htm

Guzel, B. (25 de Nov de 2009). *Top 20+ MySQL Best Practices.* Obtenido de Code.Tutsplus.com: http://code.tutsplus.com/tutorials/top-20-mysql-best-practices--net-7855

Ide, A. (31 de Enero de 2013). *PHP just grows & grows*. Obtenido de http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html

Jojooa. (s.f.). *tecnología, marketing y crm*. Obtenido de Definicion de Actor - ¿qué es un Actor?: https://sites.google.com/site/jojooa/analisis-de-sistemas/definicion-de-actor-que-es-un-actor

Kioskea. (10 de 2014). *Ciclo de vida del software*. Obtenido de Kioskea High-tech: http://es.kioskea.net/contents/223-ciclo-de-vida-del-software

Kosher, A. W. (2012, Junio 26). *forbes.com.* Retrieved from Bold Move For Brightcove, Makes Tools For App Dev Open And Free, Even For Apple TV: http://www.forbes.com/sites/anthonykosner/2012/06/26/bold-move-for-brightcove-makes-tools-for-app-dev-open-and-free-even-for-apple-tv/

Martí, R. (07 de Octubre de 2012). *stackexchange.com*. Obtenido de programmers: http://programmers.stackexchange.com/questions/167859/what-is-actually-a-module-in-software-engineering

Mastermagazine. (2015). *Definición de Casos de uso.* Obtenido de Master Magazine: http://www.mastermagazine.info/termino/4184.php

Microsoft. (2007). *Conceptos básicos sobre bases de datos*. Obtenido de support.office.com: https://support.office.com/es-ar/article/Conceptos-b%C3%A1sicos-sobre-bases-de-datos-a849ac16-07c7-4a31-9948-3c8c94a7c204?ui=es-ES&rs=es-AR&ad=AR

Mikluk, K. (28 de Octubre de 2013). *Udemy.com.* Obtenido de A Brief History of Model-View-Controller: https://www.udemy.com/blog/php-mvc-tutorial/

MySQL. (2014). *Documentacion Oficial de MySQL*.

OMG.org. (10 de Septiembre de 2014). *Introduction to OMG's Unified Modeling Language™ (UML®)*. Obtenido de Introductixzon to OMG's Unified Modeling Language™ (UML®): http://www.omg.org/gettingstarted/what_is_uml.htm

Opensource.org. (02 de Febrero de 2014). *http://opensource.org/*. Obtenido de The Open Source Initiative: http://opensource.org/

Oracle Corporation. (2014). *1.3.1 What is MySQL?* Obtenido de 1.3.1 What is MySQL?: http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html

Patricia, C. R. *Carla Rebeca Patricia de San Martin Oliva.*

Péraire, C., Edwards, M., Fernandes, A., Mancin, E., & Carroll, K. (2007). *The IBM Rational Unified Proccess for System Z.* IMB REDBOOKS.

PHP&Stuff. (6 de Noviembre de 2009). *Top 10 Reasons Why You Should Use a PHP Framework*. Obtenido de PHP&Stuff: http://www.phpandstuff.com/articles/top-10-reasons-why-you-should-use-a-php-framework

Potencier, F. (2007). *The Definitive Guide to symfony.*

Potencier, F. *Symfony 1.4, la guía definitiva.*

Prakash, S. (10 de May de 2008). *satya-weblog.com*. Obtenido de Types of PHP Framework: Glue and Full-Stack: http://www.satya-weblog.com/2008/05/types-of-php-framework-glue-and-full.html

ri5. (2014). *ri5.com.ar*. Obtenido de ri5.com.ar: http://www.ri5.com.ar/ayuda03.php

Riehle, D. (2000). *Framework Design: A Role Modeling Approach.* Zürich, Switzerland: Ph.D. Thesis, No. 13509. Zürich.

SANDOVAL, C. H. (2014). *CREACIÓN DE FRAMEWORKS CON PATRONES DE DISEÑO PARA EL DESARROLLO DE APLICACCIONES EMPRESARIALES.* Ciudad Universitaria Mexico.

Sanchez. (06 de Mayo de 2011). *Buenas prácticas para mantener odenado nuestro código PHP*. Obtenido de Interfgraphic Designs: http://www.intergraphicdesigns.com/blog/2011/06/10/buenas-practicas-para-mantener-odenado-nuestro-codigo-php/

symfony es. (11 de Octubre de 2011). *Drupal 8 integra los primeros componentes de Symfony2*. Obtenido de Drupal 8 integra los primeros componentes de Symfony2: http://symfony.es/noticias/2011/10/26/drupal-8-integra-los-primeros-componentes-de-symfony2/

The PHP Group. (2001-2014). *Manual de PHP: Prefacio*. Obtenido de http://php.net/manual/es/preface.php

The PHP Group. (2001-2014). Obtenido de Escaping from HTML: http://php.net/manual/en/language.basic-syntax.phpmode.php

The PHP Group. (2014). *php.net*. Obtenido de Historia de PHP: http://php.net/manual/es/history.php.php

Valdés, D. P. (26 de Octubre de 2007). *Maestro del WEB*. Obtenido de ¿Qué son las bases de datos: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

w3Schools. (1999-2014). *PHP 5 Variables*. Obtenido de W3Schools: http://www.w3schools.com/php/php_variables.asp

W3Schools. (2014). *PHP MySQL Database*. Obtenido de PHP MySQL Database: http://www.w3schools.com/php/php_mysql_intro.asp