MARCO DE TRABAJO MODULAR PARA LA FACILITACIÓN DEL DESARROLLO DE APLICACIONES WEB EN EL LENGUAJE PHP

MATEO SANDOVAL LUNA

UNIVERSIDAD DE CUNDINAMARCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
FUSAGASUGÁ
2019

MARCO DE TRABAJO MODULAR PARA LA FACILITACIÓN DEL DESARROLLO DE APLICACIONES WEB EN EL LENGUAJE PHP

MATEO SANDOVAL LUNA (161214225)

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

DIRECTOR:
Ing. MsC. WILSON DANIEL GORDILLO OCHOA

UNIVERSIDAD DE CUNDINAMARCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
FUSAGASUGÁ
2019

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

1. Planteamiento Del Problema

1.1. Descripción del problema

Debido a que el desarrollo web está en auge, es una tarea demandada no sólo por el sector netamente tecnológico, sino que ha abarcado otras disciplinas y mercados, esto ha llevado a que haya un aumento en la demanda de desarrolladores, y a pesar de ser uno de los trabajos con mejores prestaciones a nivel internacional, sigue existiendo una gran población aun en el sector de la informática que le tiene temor a programar debido a su complejidad y constante renovación tecnológica.

1.2. Formulación del problema

El lenguaje de programación web php lleva veintitrés años en el mercado luego de su lanzamiento, y sigue siendo uno de los lenguajes más utilizados para desarrollar aplicaciones web; el tiempo le ha dado la madurez necesaria para seguir siendo demandado por las empresas, y ser el favorito para aprender a programar ambientes web, es por esto que nace la necesidad de crear un marco de trabajo que se apalanque con el programador no necesariamente experto, le ayude en su proceso de aprendizaje y en la creación de aplicaciones web en dicho lenguaje, y tenga una comunidad que lo respalde y lo retroalimente, para así lograr una mejora continua.

2. Objetivos de la Investigación

2.1. Objetivo General

Desarrollar y validar mediante la implementación en entornos académicos el nivel de aceptación del marco de trabajo Molecular.

2.2. Objetivos Específicos

- Desarrollar un marco de trabajo bajo Php que aproveche al máximo los recursos tecnológicos del lenguaje de programación.
- Crear la documentación necesaria sobre el marco de trabajo, para su fácil aprendizaje e implementación.
- Generar espacios a nivel académico donde se lleven a cabo talleres prácticos para que la comunidad pueda aprender el uso básico de la herramienta con el propósito de validar su funcionalidad.

3. Justificación

Debido a la alta demanda de desarrollo web en los últimos años, lenguajes de programación convencionales como php, siguen teniendo la misma fuerza a medida que pasa el tiempo, por su sencillez, flexibilidad y facilidad de aprendizaje e implementación. Es por esto que aun con la oleada de nuevas tecnologías para el desarrollo web, php sigue siendo de los lenguajes más demandados por las grandes empresas, no solo a nivel local, sino internacional ya que su tiempo en el mercado brinda la confianza y robustez que se necesita para emprender un nuevo desarrollo. Para hacerle frente a dicha demanda, se requiere de un marco de trabajo (Framework) que ataque las necesidades no solo del mercado, sino a la comunidad de desarrolladores, brindándoles un esquema de trabajo enfocado a la organizacionalidad modular al momento de programar, para así facilitar el desarrollo, ahorrar tiempo y sin la necesidad de tener conocimientos avanzados, poder realizar cualquier aplicación web de alto alcance utilizando dicho Framework. Por ende, una etapa inicial de la presente propuesta de desarrollo es evaluar su utilización y despliegue en entornos académicos que serán nuestros grupos objetivo. Con todo esto se espera que la herramienta sea de gran impacto a nivel social ya que estimula el aprendizaje y las buenas prácticas guiadas por la filosofía de organización por módulos que se trabajará en el framework.

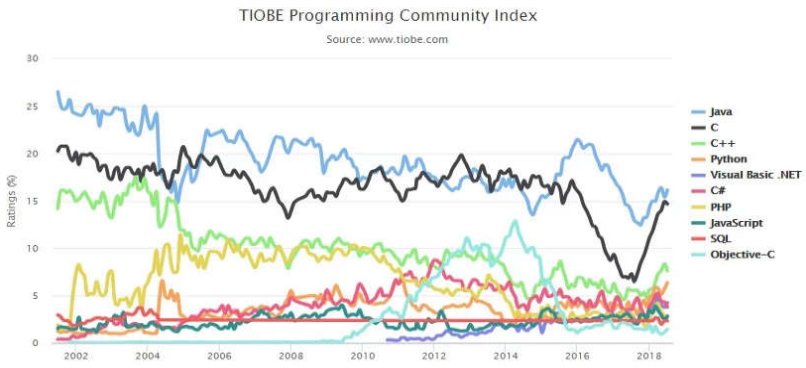


Ilustración 1 Comunidad de programadores Fuente: TIOBE Index July 2018 Trends

Jul 2018	Jul 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.139%	+2.37%
2	2		C	14.662%	+7.34%
3	3		C++	7.615%	+2.04%
4	4		Python	6.361%	+2.82%
5	7	^	Visual Basic .NET	4.247%	+1.20%
6	5	v	C#	3.795%	+0.28%
7	6	v	PHP	2.832%	-0.26%
8	8		JavaScript	2.831%	+0.22%
9	-	^^	SQL	2.334%	+2.33%
10	18	^^	Objective-C	1.453%	-0.44%
11	12	^	Swift	1.412%	-0.84%
12	13	^	Ruby	1.203%	-1.05%
13	14	^	Assembly language	1.154%	-1.09%
14	15	^	R	1.150%	-0.95%
15	17	^	MATLAB	1.130%	-0.88%
16	9	vv	Delphi/Object Pascal	1.109%	-1.38%
17	11	vv	Perl	1.101%	-1.23%
18	10	vv	Go	0.969%	-1.39%
19	16	v	Visual Basic	0.885%	-1.21%
20	20		PL/SQL	0.704%	-0.84%

Ilustración 2 Lenguajes de Programación más utilizados Fuente: TIOBE Julio 2018 Top 20

4. Alcance

El framework molecular está diseñado teniendo en cuenta los principales obstáculos identificados en los aprendices de ingeniería ya que se enfrentan a la hora de programar con problemas de lógica y complejos escenarios para desarrollar sus primeros proyectos de software, por este motivo los estudiantes no toman en cuenta esta carrera como una posible en su proyecto de vida.

Esta herramienta está enfocada a las personas que empiezan a programar debido a su fácil instalación, manejo y amigable diseño, para que aprendan a desarrollar proyectos de software.

Está proyectado para ser el nuevo framework utilizado en ámbitos académicos y profesionales.

5. Marco Referencial

5.1. Estado del arte

A lo largo de los años, luego de ser lanzado el lenguaje php oficialmente en el año 1995 por Rasmus Lerdorf, se ha venido creando diversos marcos de trabajo o también llamados Frameworks, los cuales han facilitado la vida del programador enormemente, en los más conocidos podemos encontrar:

- Symfony: Lanzado en el 2005, es uno de los frameworks más flexibles y escalables que existen para el desarrollo de aplicación bajo la primicia MVC (modelo vista, controlador) buscando que el desarrollo sea más sencillo y menos repetitivo.

- Laravel: creado en 2011 por Taylor Otwell bajo la filosofía de desarrollar código en PHP de manera elegante y simple, este fue desarrollado bajo dependencias de Symfony y paquetes de Composer, es de los más populares y es por esto que es uno de los Frameworks más utilizados para trabajar en php.

- CodeIgniter: Framework gratuito mantenido por EllisLab, empresa que también creo ExpressionEngine, otro framework para php. CodeIgniter es destacado por su gran comunidad de usuarios, su fácil instalación y lo ligero que es, ideal para principiantes ya que cuenta con librerías que gestionan errores, sistemas de seguridad y sistema de encriptado, estas librerías evitan la configuración inicial por parte del usuario así mismo evitando errores comunes que si persisten en otros frameworks.

- KumbiaPHP: es un esfuerzo por producir un framework que ayude a reducir el tiempo de desarrollo de una aplicación web, es gratuito basado en las prácticas de desarrollo web como DRY y el Principio KISS, nos permite construir robustas aplicaciones para entornos comerciales.

Como podemos ver ya hay frameworks destacados en la comunidad de desarrolladores de php, algunos gratuitos y otros de pago, pero ¿Por qué existe la necesidad de utilizar un Framework para desarrollar no solo a nivel individual sino empresarial? y ¿Por qué no incentivar su conocimiento y uso desde los centros educativos? la respuesta a esta pregunta son las ventajas que trae utilizar frameworks en la hora de programar:

- Velocidad de desarrollo
- Código documentado y bien organizado
- Desarrollo escalable, los frameworks se destacan por su reutilización de funciones y cómo estas se pueden escalar.
- Casi todos los frameworks manejan la primicia MVC, esto permite una separación entre el diseño o presentación y la lógica de la aplicación.
- Mayor seguridad en las aplicaciones, los frameworks en su mayoría cuentan con paquetes o librerías que se encargan de brindar seguridad a nuestras aplicaciones, evitando ataques informáticos (Inyecciones SQL, ataques XSS y etc)

6. Marco Conceptual

6.1. Lenguajes de Programación.

En esta sección se explica de manera concreta el ámbito web; los lenguajes de programación, su evolución, de la misma manera como estos han ido contribuyendo con el desarrollo y el avance tecnológico de la web con el pasar de los años. También se hará hincapié en aquellos lenguajes que son partícipes en el Framework.

6.1.1. Historia del desarrollo web.

Entre los años 1992 y 1994 épocas donde Netscape era el navegador web definitivo, se puede recalcar que se tenían sitios webs estáticos, con tecnología limitada, incluso el uso de imágenes era escaso, estas primeras páginas en internet tenían como propósito ser informativas o educativas.

Con la llegada de la tecnología GCI en el año 1995 se permite la creación de sitios webs dinámicos, en estos ya era más frecuente el uso de imágenes, iconos, banners, y es aquí donde empiezan a aparecer las primeras revistas online, libros y algunos pequeños formularios de contacto, permitiendo al usuario dar sus primeros pasos a la interacción con los sitios en internet. A pesar de lo anterior mencionado, siguen siendo páginas que tardan en cargar, aunque con la llegada de mejores tarjetas gráficas, se puede evidenciar la mejora en el diseño del sitio web, en este punto había dos grandes navegadores en el mercado, Internet Explorer y Netscape, es por esto que nace la W3C que busca recomendar y estandarizar conceptos del desarrollo web para así asegurar su crecimiento a mediano y largo plazo.

En 1996 el gigante tecnológico Microsoft crea nuevas tecnologías como Internet Database Connector, Active Server Pages, posteriormente aparece coldfusion, Php y las java Server Pages,

basadas en Java. La llegada de toda esta horda de tecnologías hace que el año en cuestión sea recordado como el año en que la web dinámica se consolidó, Flash llega y es uno de las más importantes implementaciones ya que da un salto importante en la experiencia de usuario, se crean las primeras tiendas virtuales, y con la evolución de la tecnología, los elementos multimedia se adueñan cada vez más de todo internet, transmitir audio o ver un video en tiempo real, ya es un hecho.

Las bases de datos empiezan a tener un papel importante en la construcción de los sitios web, ya que su creación parte de la información que se puede obtener con la base de datos.

La evolución de la experiencia de usuario empieza a tener una curva de crecimiento notoria con la evolución de HTML y la aparición de AJAX, lo que cambia el paradigma de la web dinámica, y desde entonces en los siguientes avances de la tecnología web, la experiencia de usuario es una constante, todo se basa en el mejoramiento permanente de cómo se siente el usuario desde que entra hasta que abandona una web, independientemente del enfoque de la página.

Al pasar los años las webs dejan de ser simples sitios donde la gente va a consumir información, y con la evolución de HTML5 y CSS3, el término “aplicación web” está en boca de todos, como se mencionó antes ya no es el sitio web plano, con un único carácter informativo, sino pasamos de solo consumir contenido multimedia a interactuar con una aplicación. Cada vez más las aplicaciones web prestan un servicio similar a como lo hace un programa instalado de manera local en un computador.

A medida que la tecnología web avanza los navegadores convencionales han tenido que de igual medida evolucionar, ya no es la web adaptándose a los navegadores que tienen controlado

el mercado, sino por el contrario los navegadores adaptándose a la tecnología web. Hoy por hoy un navegador que se niegue a implementar el uso de un lenguaje para el desarrollo web en crecimiento, es un navegador que quedará obsoleto en menos de lo que se imagina, y por lo tanto nadie lo usará.

Llegando a lo que sería la web actual, los términos como; escalabilidad, multiusuario, multilenguaje, los servicios de Cloud computing, son el pan de cada día de los desarrolladores, las aplicaciones web ya saltaron a otro nivel de uso, ya suplen las necesidades no solo de programas instalados en un computador sino incluso de herramientas propias de un sistema operativo; llega el Responsive por supuesto, que alude a la capacidad de una página web de adaptarse a un tamaño de pantalla especifico, ya que no solo tenemos computadores conectados a sitios en internet, sino también tablets, celulares, televisores, etc.

6.1.2. Evolución de los lenguajes de programación.

Cuando hablamos de lenguajes de programación, decimos que un lenguaje de programación es el conjunto de órdenes e instrucciones programadas que facilitan la creación de un software; aunque el concepto es general, la verdad es que lenguajes hay muchos y no todos tienen el mismo fin, es por esto que podemos clasificarlos o diferenciarlos uno del otro por el propósito de su creación o de su uso más frecuente entre desarrolladores.

Existen los lenguajes de bajo nivel, que son aquellos que sus instrucciones programadas están directamente relacionadas con el hardware en el que se ejecuta, es decir, se crea a partir de las características del hardware para sacar su mayor provecho, es condicionado y dependiente de la máquina.

Dentro de los lenguajes de bajo nivel encontramos el Lenguaje Máquina, lentos en construcción, propenso a errores y escritos en código binario, a este lenguaje se le conoce como ASSEMBLY, siendo el pionero en el mundo de la programación. En conjunto al lenguaje máquina nace el primer compilador o traductor de código llamado ASSEMBLER, la función de este era traducir el lenguaje de "alto nivel" escrito por las personas a lenguaje de "bajo nivel" o lenguaje máquina.

BASIC y FORTRAN fueron los primeros lenguajes de alto nivel, estos utilizaban una programación no estructurada y de único bloque. Luego apareció PASCAL permitiéndo hacer uso de subrutinas. Ante una latente necesidad de los desarrolladores, se empiezan a crear las primeras reglas y estructuras en torno a la codificación de programas, surgiendo así la programación estructurada, teniendo como característica una estructura jerárquica, diseño modular, también constructores de programación como: secuencia, selección, iteración y llamados a procedimientos.

Como se mencionó con anterioridad la programación modular surge con la programación estructurada, lo que permite separar por módulos cientos o miles de líneas de código a la hora de la creación de un programa, esto quiere decir, separación de tareas entre los diferentes programadores que trabajan en un mismo proyecto; la desventaja de la programación modular, está en que no se permitía la exportación o instancia de datos, el programa podría estar separado en módulos, pero aún no estaba intrínsecamente unido.

La solución a la problemática que presentaba la programación por módulos, fueron los TADs (Tipo Abstracto de Datos) permitiendo la instancia y exportación de datos. Como consecuencia

de este tipo de programación, se abre una ventana que aún perdura en el tiempo y es la programación orientada a objetos POO.

La POO da luz a un nuevo paradigma de programación, en el cual contemplamos conceptos nuevos como: clases, objetos y herencia, permitiendo así diferentes tipos de jerarquías, subclases y superclases, y con esto nace el término polimorfismo.

El primer lenguaje orientado a objetos fue Smalltalk, el cual dio los pilares para lenguajes posteriores como C++, Eiffel, Perl entre otros. Cualquier lenguaje orientado a objetos en la actualidad tiene como base los ya mencionados, ya sea heredando características o la estructura de sintaxis de dichos lenguajes; en estos nuevos lenguajes los más conocidos son: PHP, Ruby, Python, Java, C#, Go, cada uno con sus características propias, que los hacen mejores para un asunto en especial, pero todos basados en los pioneros de la programación orientada a objetos.

6.1.3. PHP

PHP (Hypertext Preprocessor) es un lenguaje creado por una comunidad de personas, El sistema fue desarrollado originalmente en el 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados de 1997 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a

nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Luego en su versión 4 vemos que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez -gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba el código-, su mayor independencia del servidor web -creando versiones de PHP nativas para más plataformas- y un API más elaborado y con más funciones.

6.1.4. HTML5, CSS3.

HTML5 es en la actualidad la versión más reciente del lenguaje de marcado para páginas web, HTML y cuyo estándar es mantenido por la wc3. Creado en 1991 y estandarizado posteriormente en 1995 ha ido evolucionando hasta la versión actual que incluye grandes funcionalidades para el desarrollo de web modernas como son: más semántico, más simple, adaptado para incluir más elementos gráficos y multimedia, geolocalización, etc.

HTML5 no estaría completo sin el uso de las hojas de estilos CSS que se encargan de proporcionar un diseño moderno y agradable.

6.2. Programación Orientada a Objetos. POO.

6.2.1. Elementos de la POO.

Los principales elementos de la POO son: clases, objetos, métodos, propiedades y mensajes y eventos.

Clase: Es un modelo que se utiliza para crear objetos que comparten propiedades y comportamientos en común.

```php
class Personas
{
    //Atributos y métodos

}
```

Objeto: Es una entidad provista de métodos o mensajes a los cuales responde (comportamiento); atributos con valores concretos (estado); y propiedades (identidad).

```php
$persona = new Personas();
```

Método: Es una función asociada a un objeto que indica la capacidad de que este puede hacer.

```php
function caminar(){
    //Algoritmo que hace caminar a una persona
}
```

Propiedades o atributos: Son variables que contienen datos asociados a un objeto y que determinan su identidad

```php
public $nombre = 'John';
public $apellido = 'Doe';
private $edad = '25';
```

Evento y Mensaje: Mientras el evento es un suceso del sistema, el mensaje es la comunicación del suceso dirigido a un objeto.

6.2.2. Características de la POO

La POO posee ciertas características que la identifican y diferencian de otros paradigmas. Estas características son: Abstracción, Encapsulamiento, Modularidad, Ocultación, Polimorfismo, Herencia y Recolector de basura.

Abstracción: Aislación de un elemento de su contexto. Define las características esenciales de un objeto y su comportamiento. La abstracción permite ocultar el cómo se implementa el comportamiento de un objeto en relación con otros en el mismo sistema. Se hace más énfasis en el ¿Qué hace? que en ¿Cómo se hace?

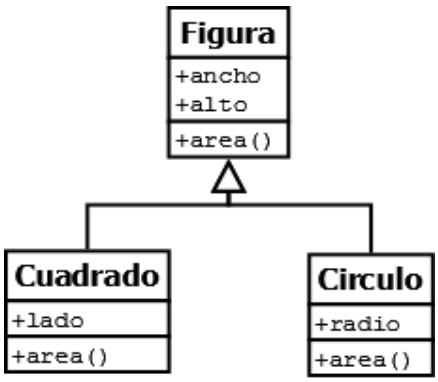


Ilustración 3 Abstracción Fuente:

En la figura observamos 3 clases diferentes, Figura, Cuadrado y Circulo, todas implementan un método para calcular el área. La Abstracción impide conocer la lógica del método.

Encapsulamiento: Reúne al mismo nivel de abstracción, a todos los elementos que puedan considerarse pertenecientes a una misma entidad. Básicamente protege las propiedades y métodos de los objetos ante posibles cambios desde fuera de la clase a la que pertenecen. Se suele confundir con la Ocultación, debido a que ambas características van de la mano.

Un ejemplo claro suele ser lo que sucede con las librerías externas que se implementan en un sistema. Una vez instanciado un objeto, este tiene acceso solo a los métodos que la librería permite y no deja interactuar al programador con el funcionamiento interno de esta.

Modularidad: Característica que permite dividir una aplicación en varias partes más pequeñas (denominadas módulos), independientes unas de otras. Esta característica es la que hace de la POO y en general de cualquier sistema construido bajo esta filosofía, en términos de calidad de software, un sistema escalable y mantenible.

Se recomienda, debido a esta característica programar usando la premisa “Programe pensando en quien mantendrá el código”.

Ocultación: Los objetos están aislados del exterior, protegiendo a sus propiedades para no ser modificadas por aquellos que no tengan derecho a acceder a las mismas. A las propiedades del objeto solo se puede acceder desde métodos internos y si es necesario la modificación de alguna propiedad, esta sea a través de un mensaje generado por un evento del usuario. Esta característica también es conocida como “principio de visibilidad”

PHP define tres niveles de visibilidad para las propiedades y los métodos de una clase, public (accesible desde cualquier parte del sistema), private (accesible solo desde la clase que lo contiene) y protected (accesibles desde la propia clase y desde la clase que la herede)

Polimorfismo: Es la capacidad que da a diferentes objetos, la posibilidad de contar con métodos, propiedades y atributos de igual nombre, sin que los de un objeto interfieran con el de otro. Si revisamos el ejemplo mencionado en la Abstracción podemos ver como objetos diferentes acceden al método propio area() y pueden implementar las modificaciones pertinentes sin interferir el uno con el otro. Además cada clase hereda de Figura, clase abstracta padre, quien obliga a sus hijas a implementar el método area().

Herencia: Esta característica hace de la POO la diferencia con otros paradigmas de programación. Es un mecanismo mediante el cual objetos específicos incorporan la estructura y comportamientos de objetos más generales.

Una clase hija heredará las propiedades y métodos de una clase padre dependiendo el nivel de visibilidad de estas en la clase padre. Así las clases hijas solo heredarán las propiedades y métodos que sean protegidos en la clase padre y claramente por defecto tienen acceso a aquellas que tienen una visibilidad pública.

En PHP no existe la herencia múltiple, sin embargo, para saltarse esta prohibición este lenguaje permite crear interfaces (clases especiales que implementan sólo la definición de métodos). Una clase en PHP puede implementar múltiples interfaces.

Recolector de basura: El GC es una función de la POO que se encarga de liberar la memoria que utilizan los objetos o variables dentro del lenguaje. Cuando un objeto o variable es creada y se le asigna un valor, se crea un puntero a la memoria que contiene esa información y cuando este puntero ya no existe, no es referenciado, el GC se preocupa de liberar ese espacio en la memoria.

En PHP no se crean punteros, sino una referencia a dicho objeto en una tabla de símbolos. Un objeto o variable podrá ser eliminada de la memoria cuando el GC no tenga ninguna referencia a dicho objeto en la tabla de símbolos. Es decir, PHP mantiene un contador de referencias a un objeto desde el momento en que éste es creado, de forma que durante la ejecución del script PHP va incrementando y decrementando dicho contador de referencias en función de las variables que

le van “apuntando”. Una vez ese contador de referencias llega a 0 (es decir, nadie está relacionado con ese objeto y, por lo tanto, no se está utilizando dicho objeto), PHP marca ese objeto como basura o eliminable, de forma que en la siguiente pasada del GC, éste eliminará dicho objeto de memoria y esa posición de memoria será utilizable por PHP nuevamente.

Esta función en PHP viene activada por defecto, pero también es posible ejecutarla o desactivarla en tiempo de ejecución.

6.3. Patrones de Diseño

Desde la aparición de la Reusabilidad (o quizás antes) la programación se ha desarrollado bajo este concepto, donde librerías o bibliotecas (independientes del lenguaje en el que se han escrito) ya probadas y documentadas se pueden usar en cualquier proyecto. Sin ir muy lejos, PHP y JAVA están compuestos principalmente por librerías (de acceso a datos, de matemáticas, para manejar arreglos, etc.) desde las que podemos llamar a sus clases y funciones sin más que importar esa librería con una simple línea de código.

Lo anterior estaba bien, pero la Reusabilidad no solucionaba de igual manera problemas comunes dentro de aplicaciones similares (El menú de una aplicación no tiene que ver con el menú de otra aplicación). Aparecen entonces los patrones de diseño como soluciones a problemas recurrentes en el desarrollo de un software. Estos patrones de diseño, facilitan la reutilización de diseños y arquitecturas que han tenido éxito en otras aplicaciones.

Patrones de Diseño hay muchos y siguen apareciendo continuamente. El desarrollo de aplicaciones, como disciplina, está en constante cambio y así también los problemas de diseño. Así que las herramientas que se usan para los desarrollos también se van actualizando y

mejorando en favor de estos cambios. Se pueden clasificar en tres tipos, patrones de creación, patrones de comportamiento y patrones de estructura.

Patrones de Creación: Hace referencia a la creación de objetos dinámicamente y algunos de estos patrones son, *Singleton, Factory, Factory Method, Abstract Factory, Builder* y otros.

Patrones de Comportamiento: Se preocupa de resolver problemas sobre la estructura de clases. Tenemos *a Interpreter, Iterator, Mediator, Memento, Observer, Strategy* y otros.

Patrones de Estructura: Relaciona el comportamiento entre los objetos de la aplicación. Acá encontramos a *Adapter, Bridge, Composite, Decorator, Flyweight, Proxy*.

Es importante tener presente los siguientes elementos para entender un patrón y sobre todo para la elección de uno: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Con relación a las consecuencias, el uso de un patrón viene dado por su eficiencia, flexibilidad, uso de memoria, tiempo de ejecución y seguridad.

En el framework a desarrollar se implementa el Patrón Singleton que pertenece a los patrones de creación. Este patrón restringe la creación de objetos de clase a uno solo. Garantiza que hay solo una instancia y proporciona un único punto de acceso global.

6.4. MVC

En la extensa literatura sobre el tema, podemos decir que no hay un acuerdo sobre qué patrón de diseño cumple MVC. Algunos dicen, que no es un patrón de diseño, sino un patrón arquitectónico, otros lo dan como patrón de creación.

Lo que se puede asegurar es que MVC cumple con todos los elementos ya señalados para denominarse un patrón, con la salvedad de que abarca no un problema concreto en una aplicación, sino más bien se enfoca en cómo estructurar la solución al desarrollo (estilo arquitectónico).

MVC, son las siglas de Modelo-Vista-Controlador y como buen patrón brinda de un marco de desarrollo de aplicaciones a través de la construcción de tres componentes distintos, el Modelo, la Vista y el Controlador

Este patrón separa la lógica de negocio (Modelo) de la interfaz de usuario (Vista) y de la lógica de la aplicación (Controlador). Esta separación implica una mejor funcionalidad, mantenibilidad, escalabilidad y reusabilidad (Conceptos de Calidad del Software) de un sistema, de forma simple, sencilla, en código legible, entendible, y flexible a todo programador que esté en condiciones de desarrollar el sistema.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad. Modelos, Vistas y Controladores.

Modelos: Representan la lógica del negocio. Es quien hace de intermediario con la Base de Datos, accediendo a los datos de esta. También posee reglas de negocio, que hace referencia al manejo de datos.

Vistas: Representa la Interfaz de Usuario. Es quien se encarga de presentar los datos al usuario mediante una interfaz gráfica. Gestiona la información de entrada (como pedirla) y la de salida (como mostrarla).

Controladores: Representa la lógica de la aplicación. Es quien hace de intermediario entre el modelo y la vista, a través de eventos generados por el usuario para así acceder a los datos necesarios y presentarlos en la vista.

El funcionamiento básico del patrón MVC, puede resumirse en:

- El usuario realiza una petición
- El controlador captura el evento (puede hacerlo mediante un manejador de eventos o peticiones, por ejemplo)
- Hace la llamada al modelo/modelos correspondientes (por ejemplo, mediante una llamada de retorno o callback) efectuando las modificaciones pertinentes sobre el modelo
- El modelo será el encargado de interactuar con la base de datos, ya sea en forma directa, con una capa de abstracción para ello, un Web Service, etc. Y retornará esta información al controlador.
- El controlador recibe la información, la procesa y la envía a la vista
- La vista, recibe esta información y genera el diseño de la interfaz gráfica o GUI. La lógica de la vista, una vez procesados los datos, los acomodará con base en el diseño de la plantilla, y los entregará al usuario de forma humanamente legible.

7. Marco Metodológico

Para la realización del proyecto, se tuvo en cuenta diferentes metodologías de trabajo que se utilizan en el desarrollo de software, sin embargo, se seleccionó la metodología tradicional Desarrollo en Cascada, debido a que se hace importante desarrollar las fases de manera secuencial. Esta metodología se desarrolla en fases claramente definidas, donde cada fase no puede comenzar hasta tanto la anterior no haya concluido. El modelo de Desarrollo en Cascada fue el primero en originarse, y es la base para el ciclo de vida de un programa.

Las fases que plantea el Desarrollo en Cascada son:

1. Análisis de requerimientos.
2. Diseño del sistema.
3. Desarrollo o codificación.
4. Pruebas
5. Implementación o validación.
6. Mantenimiento.

La Implementación (Fase 5) hará referencia a la validación del Framework por parte de diferentes grupos académicos, y Mantenimiento (Fase 6) estará dirigida a la publicación del framework para su uso.

7.1. Análisis De Requerimientos

En el momento del planteamiento del proyecto y definir a PHP como lenguaje de programación para desarrollar el proyecto, se encontraron puntos importantes a tener en cuenta para desarrollar el framework:

- Estructura flexible.
- Reutilización de código.
- Manejo básico de seguridad.

- Velocidad de implementación.
- Trabajo colaborativo.

Es por esto que la definición de requerimientos se hizo teniendo en cuenta cada uno de ellos.

7.1.1. Requerimientos funcionales

Identificación del requerimiento	RF1
Nombre del requerimiento	Gestión de la URL
Características	URL de navegación limpia y libre de caracteres especiales que muestren variables o el archivo que se está ejecutando.
Descripción del requerimiento	El sistema deberá ser capaz de entender una URL dividida con barras diagonales y procesar la petición.
Requerimiento no funcional	RNF:

Identificación del requerimiento	RF2
Nombre del requerimiento	Archivo de configuración
Características	Archivo de parametrización del sistema, donde se parametrizan aspectos generales como los datos de acceso a una base de datos, nombre de la aplicación, url base.
Descripción del requerimiento	El sistema deberá tener un archivo de configuración, en el que se definan las Constantes globales y que se reconozcan por todo el sistema.
Requerimiento no funcional	

Identificación del requerimiento	RF3
Nombre del requerimiento	Patrón Singleton

Características	Uso del patrón de diseño Singleton.
Descripción del requerimiento	El sistema deberá restringir la instancia de objetos a uno solo y proporcionar un único punto de acceso global.
Requerimiento no funcional	

Identificación del requerimiento	RF4
Nombre del requerimiento	Conexión a base de datos
Características	En el sistema se definirán Constantes en las que se almacenará la información de conexión a una base de datos (host, usuario, contraseña, nombre de base de datos).
Descripción del requerimiento	El sistema deberá proveer la conexión a un motor de base de datos, haciendo uso de los valores definidos en las Constantes de configuración (host, usuario, contraseña, nombre de base de datos).
Requerimiento no funcional	

Identificación del requerimiento	RF5
Nombre del requerimiento	Patrón MVC
Características	Uso del patrón de arquitectura MVC
Descripción del requerimiento	El sistema deberá contar con arquitectura Modelo-Vista-Controlador, estructurado de manera que cada parte lógica se guarde en archivos separados.
Requerimiento no funcional	

Identificación del requerimiento	RF6

Nombre del requerimiento	Módulos
Características	El sistema contará con una carpeta en la que se guarden paquetes de funcionalidades individuales denominados módulos.
Descripción del requerimiento	El sistema deberá reconocer de manera automática los paquetes instalados en una carpeta determinada y poder redireccionar a cada uno de ellos. Estos deberán seguir el patrón MVC.
Requerimiento no funcional	

Identificación del requerimiento	RF7
Nombre del requerimiento	Excepciones
Características	Mensajes de alerta en el que se indican posibles errores del sistema.
Descripción del requerimiento	El sistema deberá contar con un manejo de excepciones en el que se muestre a través de la interfaz el mensaje.
Requerimiento no funcional	

Identificación del requerimiento	RF8
Nombre del requerimiento	Manejo de sesión
Características	Escritura, lectura y administración de variables de sesión para mantener datos temporales durante la ejecución del sistema.
Descripción del requerimiento	El sistema deberá suministrar la estructura para el manejo de una sesión por parte del desarrollador, de manera que solo tenga que declarar y leer las variables.
Requerimiento no funcional	

Identificación del requerimiento	RF9
Nombre del requerimiento	Uso de widgets
Características	Fragmentos de código que contienen su propia lógica y diseño, pueden usarse en todas las Vistas,
Descripción del requerimiento	En el sistema se podrán diseñar widgets o fragmentos de código únicos, que puedan ser utilizados por todas las Vistas
Requerimiento no funcional	

Identificación del requerimiento	RF10
Nombre del requerimiento	Autocarga de clases
Características	Reconocimiento automático de clases nuevas para cargar librerías.
Descripción del requerimiento	El sistema deberá reconocer de manera automática nuevas Clases que se instalen sin necesidad de incluirlas.
Requerimiento no funcional	

Identificación del requerimiento	RF11
Nombre del requerimiento	Función de encriptación
Características	Clase preestablecida con un método de encriptación y que utiliza una llave establecida por el desarrollador.
Descripción del requerimiento	El sistema establecerá un método de encriptación para que el desarrollador pueda hacer uso de él solo enviando la cadena de texto que desea encriptar y el tipo de encriptado.

Requerimiento no funcional	

Identificación del requerimiento	RF12
Nombre del requerimiento	Plantilla para la interfaz
Características	Diseño del interfaz dividido en encabezado y pie de página.
Descripción del requerimiento	El sistema definirá el diseño de la interfaz, dividiéndolo en encabezado, contenido y pie de página, para ser usado como plantilla.
Requerimiento no funcional	

Identificación del requerimiento	RF13
Nombre del requerimiento	Estilos de plantilla
Características	El desarrollador podrá diseñar diferentes estilos para su aplicación y elegir uno como predeterminado.
Descripción del requerimiento	El sistema deberá almacenar diferentes estilos de interfaz, separándolos por carpetas y permitiendo seleccionar uno por defecto en la configuración general.
Requerimiento no funcional	

Identificación del requerimiento	RF14
Nombre del requerimiento	Cambio de estilo de una
Características	

Descripción del requerimiento	
Requerimiento no funcional	

7.1.2. Requerimientos no funcionales

Identificación del requerimiento	RNF1
Nombre del requerimiento	Condiciones
Características	La versión mínima de PHP será de 7.0 .
Descripción del requerimiento	Para garantizar un óptimo funcionamiento el servidor en el que se instale deberá tener PHP versión 7.0 o superior.
Prioridad del requerimiento	Alta

Identificación del requerimiento	RNF2
Nombre del requerimiento	Navegador
Características	El sistema será funcional en cualquier navegador.
Descripción del requerimiento	El sistema será funcional en cualquier navegador web (Mozilla Firefox, Google Chrome, Opera)
Prioridad del requerimiento	Alta

Identificación del requerimiento	RNF3
Nombre del requerimiento	Validación de archivos
Características	El sistema validará los tipos de archivos.

Descripción del requerimiento	El sistema validará los tipos de archivos y formatos admitidos a la hora de crear nuevos.
Prioridad del requerimiento	Media

Identificación del requerimiento	RNF4
Nombre del requerimiento	Contenido
Características	El sistema sigue las reglas básicas de gramática y ortografía.
Descripción del requerimiento	El sistema sigue las reglas básicas de gramática, ortografía y composición literaria.
Prioridad del requerimiento	Alta

Identificación del requerimiento	RNF5
Nombre del requerimiento	Estructura
Características	El sistema tendrá una estructura organizada de manera lógica y coherente.
Descripción del requerimiento	El sistema tendrá una estructura organizada de manera lógica y coherente según las funciones de cada módulo.
Prioridad del requerimiento	Alta

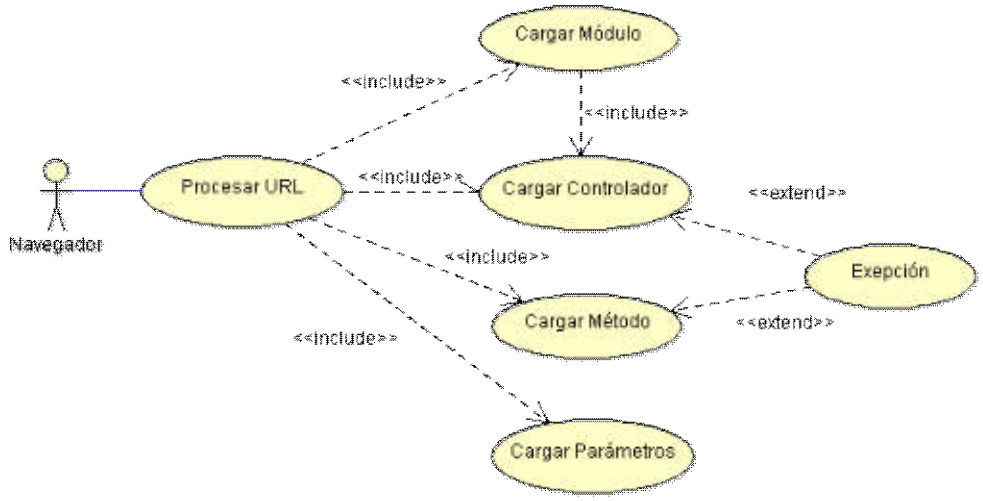
Identificación del requerimiento	RNF6
Nombre del requerimiento	Calidad
Características	El sistema tendrá calidad.

Descripción del requerimiento	El sistema tendrá buena calidad técnica y estética.
Prioridad del requerimiento	Alta

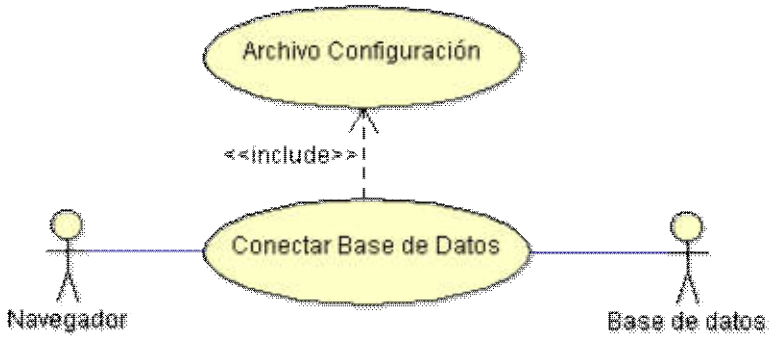
Identificación del requerimiento	RNF7
Nombre del requerimiento	Confiabilidad
Características	El sistema deberá ser confiable.
Descripción del requerimiento	El sistema deberá ser confiable tolerante a errores además que sus operaciones deben ser transaccionales.
Prioridad del requerimiento	Alta

Identificación del requerimiento	RNF8
Nombre del requerimiento	Escalabilidad
Características	El sistema será escalable.
Descripción del requerimiento	El sistema contemplará el crecimiento de usuarios y recursos.
Prioridad del requerimiento	Alta

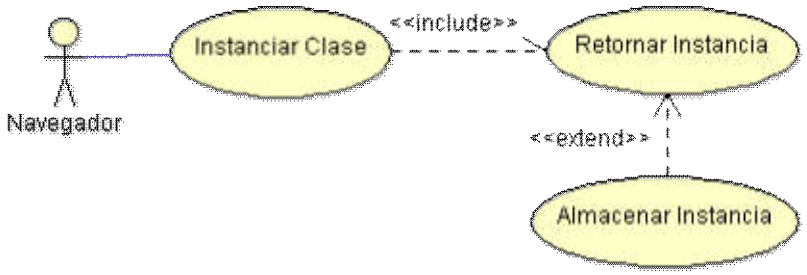
7.1.3. Casos de uso



RF<1>	**Url Amigable**	
Descripción	El sistema procesa la url del navegador	
Precondición		
Secuencia Normal	**Paso**	**Acción**
	1	El navegador recibe la url.
	2	La Clase Request recibe la petición y extrae las variables que están separadas por barras diagonales.
	3	Cargar el controlador.
	3a	Si la variable 1 se encuentra como nombre de los módulos registrados, busca el controlador con la segunda variable.
	3b	Busca el controlador con el nombre de la variable 1 en la carpeta de los controladores.
	4	Con la siguiente variable carga el Método.
	5	Las siguientes variables que encuentre las carga en un arreglo para usarlas como parámetros.
Postcondición	Llama al archivo que tiene el controlador.	
Excepciones	**Paso**	**Acción**
	1	En caso de no encontrar el controlador o no poder cargar el archivo, debe lanzar una excepción informando.
	2	Si no encuentra el método dentro del controlador, debe lanzar una excepción informando.
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.	
Prioridad	Vital	
Urgencia	Inmediatamente	
Comentarios		

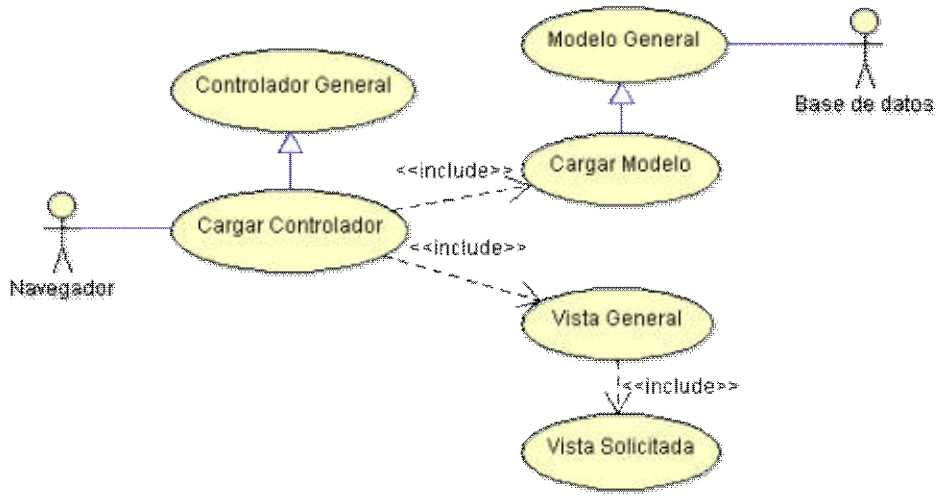


RF<4>	Conexión a base de datos	
Descripción	El sistema instancia y se conecta a una base de datos.	
Precondición	Los datos de conexión a la base de datos deben estar definidos en el archivo de configuración.	
Secuencia Normal	**Paso**	**Acción**
	1	Crear el objeto de tipo Database con los datos de conexión que se encuentran en el archivo de configuración.
	2	Realizar la conexión con el motor de base de datos.
	3	Instanciar el Modelo con el objeto de la conexión.
Postcondición	Llama al archivo que tiene el controlador.	
Excepciones	**Paso**	**Acción**
	1	Si la conexión no se puede realizar, debe lanzar una excepción informando.
	2	Si ya existe una instancia del objeto Database, se reemplaza con la nueva.
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.	
Prioridad	Importante	
Urgencia	Inmediatamente	
Comentarios		



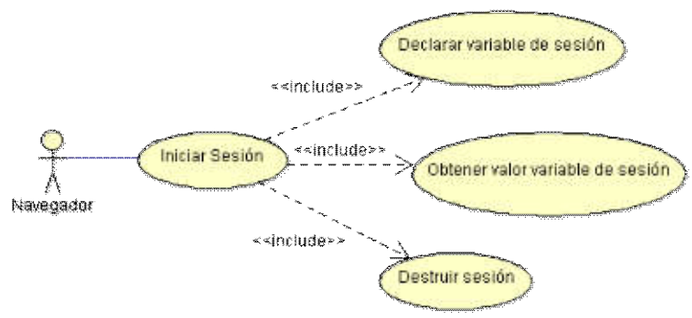
RF<3>	Uso del patrón de diseño Singleton.
Descripción	El sistema procesa la url del navegador
Precondición	

Secuencia Normal	**Paso**	**Acción**	
	1	Instanciar el objeto de tipo Registry.	
	2	Cuando una Clase requiera instanciar un objeto, debe hacerlo a través del objeto Registry.	
	3	Instanciar un objeto	
		3a	Si la instancia ya se encuentra almacenada, devolverá el objeto.
		3b	Si la instancia no se encuentra registrada, se almacenará y se devolverá el objeto.
Postcondición	El tiempo de vida de las instancias dependerá del objeto Registry.		
Excepciones	**Paso**	**Acción**	
	1		
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.		
Prioridad	Importante		
Urgencia	Inmediatamente		
Comentarios			

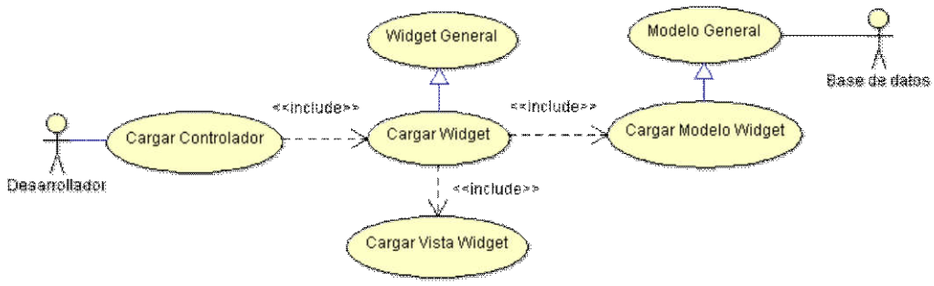


RF<5>	**Patrón MVC.**	
Descripción	El sistema usa arquitectura Modelo Vista Controlador	
Precondición	Procesar URL para determinar el controlador a cargar.	
Secuencia Normal	**Paso**	**Acción**
	1	Incluir el archivo del controlador solicitado. Este controlador deberá heredar del Controlador General.
	2	El controlador hace una llamada al modelo que requiera. Los modelos deben heredar del Modelo General que es el que contiene la conexión a la base de datos.
	3	El controlador incluirá la Vista General que es donde se define la plantilla, y a través de ella incluirá la Vista solicitada.
Postcondición		
Excepciones	**Paso**	**Acción**

	1	En caso de no encontrar el controlador o no poder cargar el archivo, debe lanzar una excepción informando.
	2	Si no se encuentra el Modelo requerido, el sistema debe lanzar una excepción informando.
	3	Si no se encuentra la Vista requerida, el sistema debe lanzar una excepción informando.
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.	
Prioridad	Importante	
Urgencia	Inmediatamente	
Comentarios		



RF<8>	**Manejo de Sesión.**	
Descripción	El sistema administra la sesión.	
Precondición		
Secuencia Normal	**Paso**	**Acción**
	1	Instanciar el objeto de tipo Session.
	2	Declarar una variable de sesión enviando el nombre y el valor. Desde cualquier lugar del sistema se podrá declarar una variable.
	3	Consultar el valor de la variable.
Postcondición	El tiempo de vida de la sesión por inactividad será definido en el archivo de configuración.	
Excepciones	**Paso**	**Acción**
	1	Si una variable de sesión no se encuentra declarada, no se retornará nada.
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.	
Prioridad	Media	
Urgencia	Inmediatamente	
Comentarios		



RF<9>	**Uso de Widgets.**		
Descripción	Los widgets se podrán usar desde cualquier controlador y por ende visualizarse en cualquier vista.		
Precondición	El widget por utilizar debe estar en la carpeta de los Widgets, heredando de la clase Widget.		
Secuencia Normal	**Paso**	**Acción**	
	1	Carga de modelo.	
		1a	Si el widget requiere conexión a la base de datos, se instancia el modelo que requiere.
		1b	Si los datos son estáticos recibirá un arreglo desde el controlador.
	2	Instanciar la vista del widget.	
	3	Declarar el widget en el controlador.	
	4	Imprimir el widget en la vista.	
Postcondición	Visualización del widget en la vista		
Excepciones	**Paso**	**Acción**	
	1	Si la vista o el modelo del widget se encuentran, deberá generar una excepción.	
	2	Si se imprime el widget en la vista sin estar declarado en el controlador, no se mostrará.	
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos.		
Prioridad	Media		
Urgencia	Inmediatamente		
Comentarios			

7.1.4. Diagrama de Secuencia

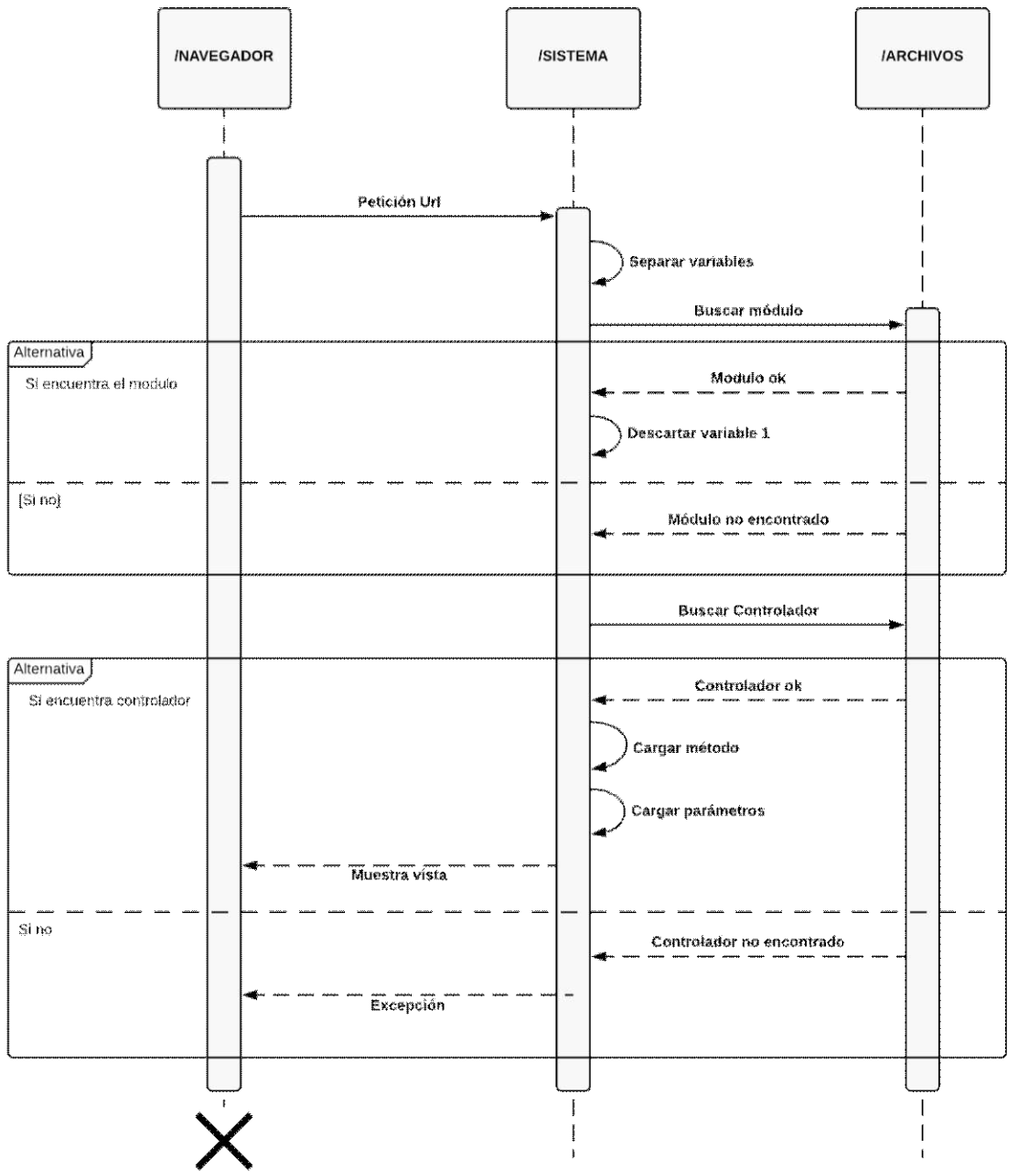


Ilustración 4 Diagrama de secuencia Fuente: Autor

7.2. Diseño del sistema

7.2.1. Diagrama de clases

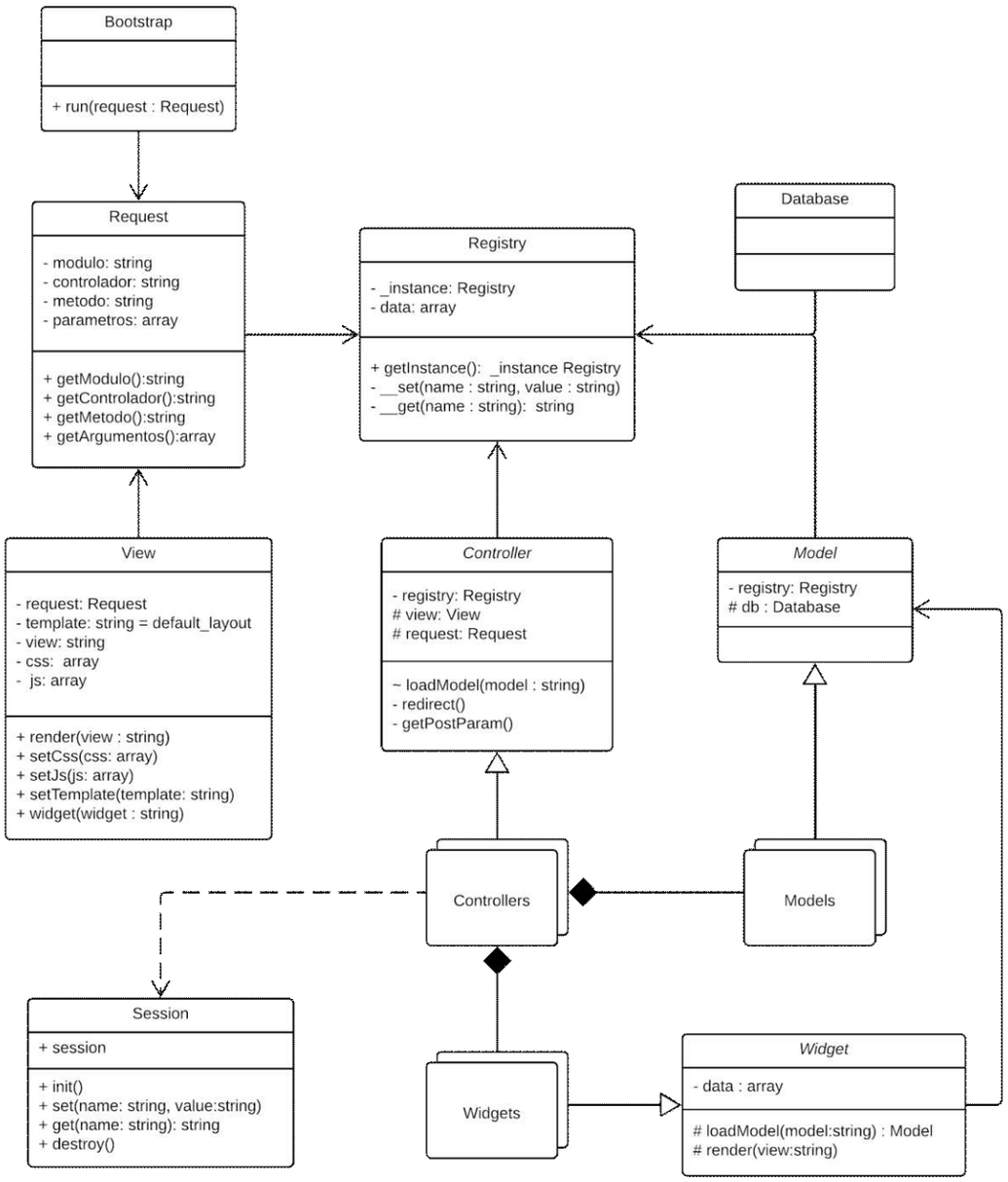


Ilustración 5 Diagrama de clases Fuente: Autor

7.2.2. Diagrama de paquetes

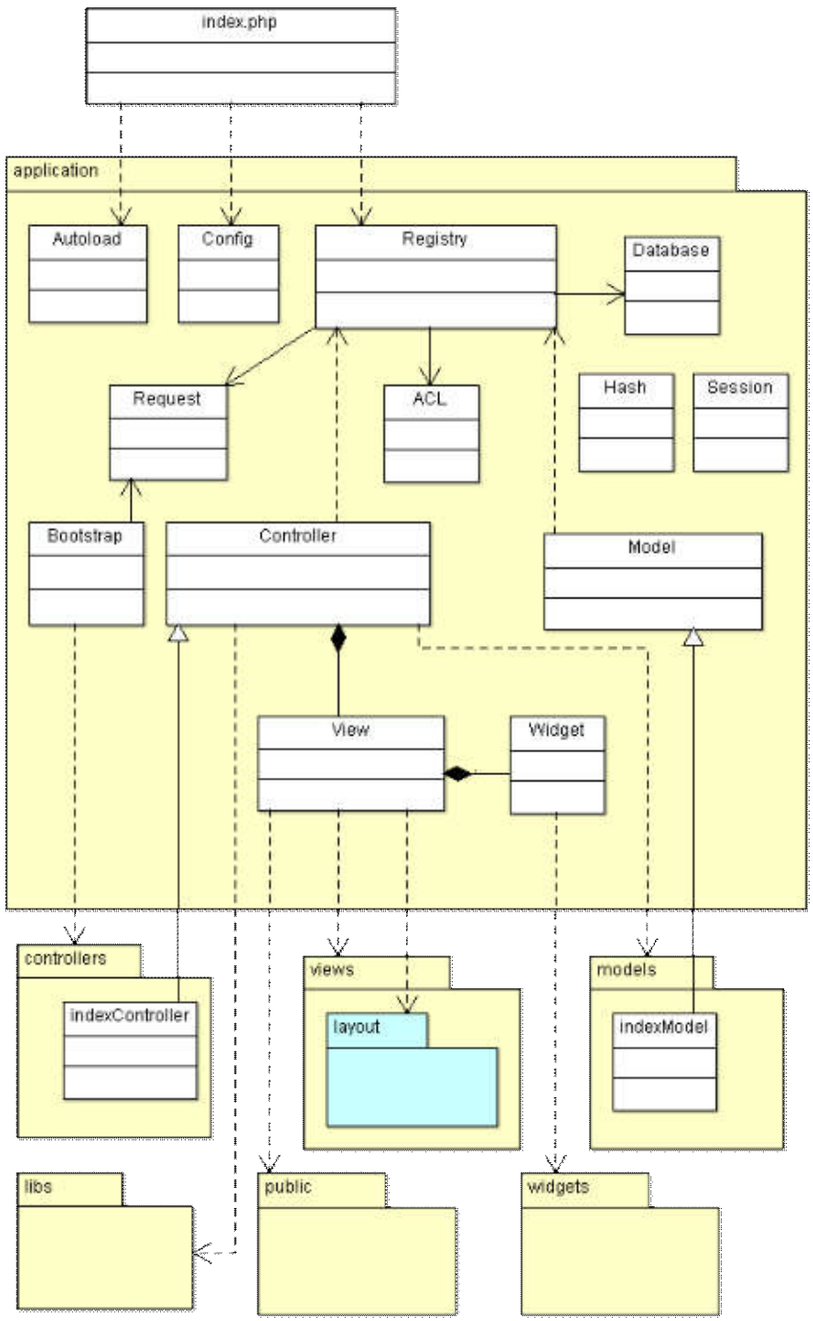


Ilustración 6 Diagrama de paquetes Fuente: Autor

7.3. Desarrollo

7.3.1. Estructura básica

A continuación, se muestra la estructura de carpetas y archivos del framework

Ilustración 7 Estructura Framework Molecular Fuente: Autor

En la siguiente tabla se relaciona cada componente de la estructura básica, así como su función o contenido dentro del framework.

Tabla 1 Estructura básica Molecular Fuente: Autor

Estructura básica de Molecular	
Carpeta / Archivo	Función / Contenido
Application	Contiene los archivos del núcleo del Framework.
Controllers	Archivos de las clases que actúan como controladores.
Libs	Archivos o carpetas de librerías externas.
Models	Archivos de las clases que actúan como modelos.
Modules	Carpetas de módulos que contienen sus propios controladores, modelos y vistas.
Public	Archivos y carpetas públicas que no hacen parte del framework.
Views	Archivos y carpetas con las vistas necesarias para cada controlador. También contiene la carpeta que almacena las plantillas generales.
Widgets	Archivos y carpetas que contienen la lógica de los widgets.
.htaccess	Archivo de configuración del servidor Apache.
index.html	Archivo de control de acceso.
index.php	Archivo que recibe todas las peticiones al framework.

A partir de esta estructura de carpetas y archivos es que funcionan las clases diseñadas en el núcleo del framework. Parte de esta estructura radica en la necesidad de tener separados los bloques de código y la lógica de los sistemas. De esta forma, se puede saber dónde buscar cuando se presentan los errores.

7.3.2. Núcleo del framework

En la carpeta *Application* se encuentra ubicado el núcleo del framework. Es en esta carpeta donde se alojan todas las clases maestras que aportan las diferentes funcionalidades al framework.

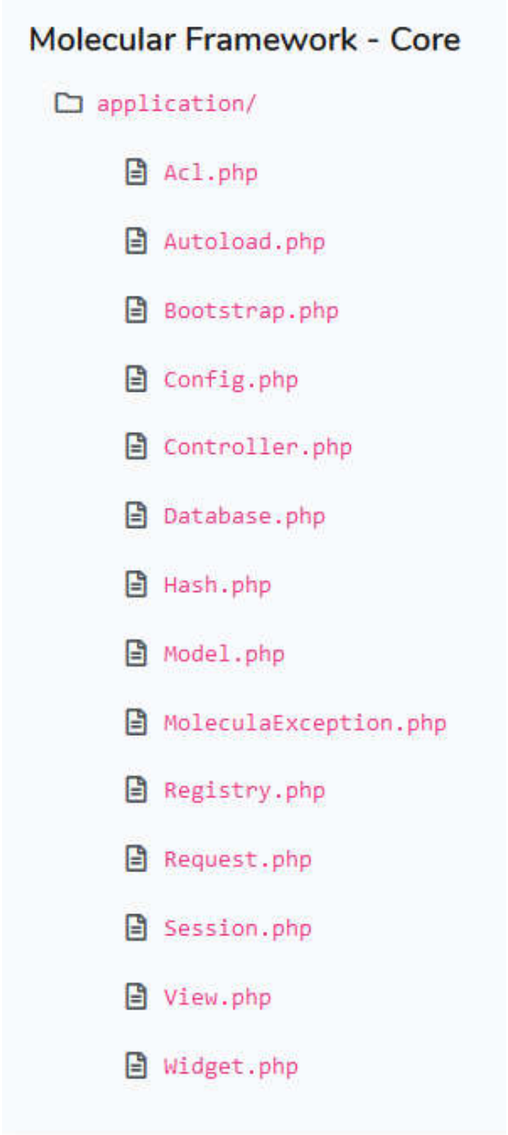


Ilustración 8 Núcleo Framework Molecular Fuente: Autor

En la siguiente tabla se explica la función de cada una de las clases que intervienen en el núcleo del framework, así como el tipo de archivo que es:

Tabla 2 Núcleo del Framework Fuente: Autor

Núcleo del Framework		
Archivo	Tipo	Función / Contenido
Autoload.php	Archivo PHP	Funciones para cargar automáticamente las clases que se añaden.
Bootstrap.php	Clase PHP	Clase que se encarga de llamar el controlador que se envía en la petición.
Config.php	Archivo PHP	Contiene la definición de las constantes de todo el sistema, y sirve para parametrizar el sistema.
Controller.php	Clase abstracta PHP	Es el Controlador padre que contiene todas las funciones globales.
Database.php	Clase PHP	Clase que hereda de la clase PDO de PHP para hacer la conexión con la base de datos.
Hash.php	Clase PHP	Clase que permite hacer encriptación de los datos.
Model.php	Clase PHP	Es la Modelo padre del framework.
MolecularException.php	Clase PHP	clase que maneja las excepciones y errores del framework.
Registry.php	Clase PHP	Clase con la que se mantiene una sola instancia de las clases. Implementación del patrón Singleton.
Request.php	Clase PHP	Clase que procesa las peticiones de la URL, y define el controlador que se debe cargar.
Session.php	Clase PHP	Clase que administra la sesión y sus variables en el framework.
View.php	Clase PHP	Clase que administra las vistas en el framework.

Widget.php	Clase PHP	Es el Widget padre del framework que contiene todas las funciones globales

7.3.3. Archivo de configuración

Las constantes que se utilizan a lo amplio de todo el framework se encuentran declaradas en el archivo *application/Config.php*, y esto se hace como medida para facilitar la administración del sistema que se va a desarrollar. Se utiliza principalmente para definir:

- Dominio del sistema.
- Controlador por defecto.
- Plantilla por defecto.
- Información de conexión a la base de datos.
- Tiempo de la sesión.
- Llave para la encriptación.
- Habilitar el modo depuración.
- Publicar el sitio.
- Nombre del sistema.
- Nombre y url de la compañía.

7.3.4. Controladores

Los controladores son los encargados de administrar la lógica de negocio, usando la información que recuperan de los modelos y transfiriendola a la vista para que sea visualizada por el usuario. Los controladores hijos deben heredar del Controlador general, así pues, quedan dotados de todas las funcionalidades que vienen por defecto.

El Controlador general, se encuentra en *application/Controller.php*, y se encarga de escuchar las peticiones e instanciar el objeto de la Vista. Contiene los métodos de instancia a Modelos y

Librerías, así como varios métodos que permiten un manejo seguro de la información que se captura en los formularios de las vistas.

Los controladores hijos pueden estar ubicados en la carpeta ***controllers/*** o dentro de la carpeta ***modules/.../controllers/*** si es que pertenecen a algún módulo.

La existencia de un controlador hijo no implica que deba llevar un modelo o una vista asociada, puesto que pueden ser utilizados para manejar objetos de tipo XML o JSON a través de AJAX.

7.3.5. Modelos

Los modelos, al igual que con los controladores, son clases que heredan de un Modelo padre que se encuentra en application/Model.php. En este Modelo padre es donde se usa la instancia de la clase Database, que no es más que una clase que contiene la conexión a la base de datos, debido a que está heredando de la librería PDO de PHP.

Así pues, es el Modelo padre quien reparte la conexión en los Modelos hijos.
Los Modelos hijos se ubican en la carpeta ***models/*** de manera general, o en la carpeta ***modules/.../models/*** si es que pertenecen a algún módulo.

7.3.6. Vistas

Cuando el Controlador padre instancia el objeto de la clase View, está cargando por defecto la plantilla que se define en el Archivo de configuración.

Las plantillas complementan a las vistas y les aportan el estilo gráfico. Están compuestas por dos archivos: header.php, footer.php; y por tres carpetas de recursos: css, js, media. Se ubican en una carpeta que lleva su nombre y se aloja dentro de la carpeta views/layout/. De esta forma la plantilla da la estructura visual que compone toda la vista.

Las vistas son archivos con extensión phtml, que son requeridos desde los controladores hijos y están ubicados de manera general en la carpeta ***views/***, o en la carpeta ***modules/.../views/*** si es que pertenecen a algún módulo.

7.3.7. Módulos y librerías

Un Módulo, es un conjunto de clases y archivos que se disponen en una estructura MVC, y que tienen como objetivo aportar un conjunto de funcionalidades específicas al sistema.

Los Módulos son integrados al framework de manera automática, solo alojando la carpeta de este dentro de la carpeta modules/. De esta forma pueden ser también eliminados sin necesidad de alterar el funcionamiento del framework. Un ejemplo de uno de los módulos más utilizados es la gestión de los usuarios en el sistema, donde se incluye: registro, inicio de sesión, roles, permisos, etc.

Una librería, es un archivo o conjunto de varios elementos que cumplen tareas específicas dentro del framework y sirven para cualquier tipo de proyecto. No siguen es necesario que cumplan con la estructura MVC. Las librerías son integradas al framework también de manera automática. En su mayoría son desarrolladas por terceros. Algunas de las más utilizadas son PHPMailer y FPDF.

7.3.8. Funcionamiento

Un framework que usa el patrón MVC, cuenta con una arquitectura de tres capas: una lógica, una de datos, y otra visual. El Controlador, o capa lógica es la que se encarga de relacionar a las otras dos, siendo esta la que articula la comunicación entre los datos y la parte gráfica.

El ciclo del funcionamiento se compone de 6 pasos:

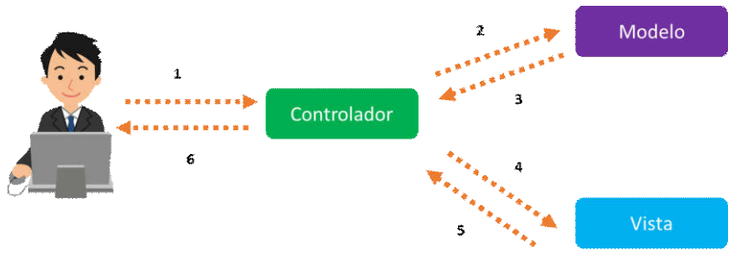


Ilustración 9 Patrón MVC Fuente: Autor

1. El usuario envía una petición a través de la URL. El encargado de procesar la URL es la clase Request. Los tipos admitidos de URL son:
   - Dominio/módulo/controlador/método/param1/param2
   - Dominio/controlador/método/param1/param2

   Primero se realiza la separación de la url a partir de cada barra diagonal que se encuentre y se almacena en un arreglo. De esta forma cada valor del arreglo representa uno de los valores a buscar. Se realiza una comprobación de existencia del módulo en el framework con el primer valor, y en caso de encontrarlo, procede a cargar un controlador de este módulo, ejecuta el método y por último envía los parámetros recibidos en un arreglo a ese método. En caso de que la comprobación del módulo falle, se procede a buscar en los controladores usando el primer valor del arreglo y sigue los pasos anteriormente descritos.
   Si la URL que se ejecuta no contiene ningún valor aparte del dominio, la clase Request asigna el controlador que está asignado por defecto en el Archivo de configuración.
   El método por defecto a ejecutar en los controladores es el Index.
2. El controlador solicita los datos al modelo. Es paso depende de si la información que se va a mostrar depende de una base de datos o es simplemente estática.
   En caso de requerir la base de datos, es necesario que el controlador cargue el modelo que requiere y ejecute el método que necesite.

3. El modelo retorna los datos al controlador. Este paso está ligado al anterior, por lo que depende de si se hizo alguna petición al modelo.
4. El controlador llama a una vista. Si bien la vista que se llame debería ir ligada al método que se está ejecutando, es posible reutilizar otra vista del mismo paquete de vistas del controlador.
5. La vista retorna al controlador. La vista se carga en el controlador y queda lista para mostrarse.
6. El controlador carga la vista con los datos. La vista se renderiza en el navegador, haciendo uso de las variables que se cargaron desde el controlador.

7.4. Implementación

7.4.1. Talleres prácticos

Dentro de la Universidad de Cundinamarca, se buscaron grupos que tuvieran conocimientos de programación y bases de datos. Para ello se seleccionaron dos grupos de Ingeniería de Software I y uno de Ingeniería de Software II, a quienes contando con el aval de los docentes se les realizaría la socialización de la herramienta desarrollada.

Cada uno de los talleres se realizó de manera individual entre los grupos, buscando que con ello se apalancaran en la herramienta y desarrollaran el proyecto semestral que exige cada una de las materias.

Los talleres se desarrollaron en cuatro momentos:

1. Socialización del proyecto e introducción a la herramienta.
2. Aspectos básicos y manejo de la documentación.

3. Actividad práctica de desarrollo.

4. Validación de la herramienta.

Ilustración 10 Taller Práctico en Universidad de Cundinamarca Fuente: Autor

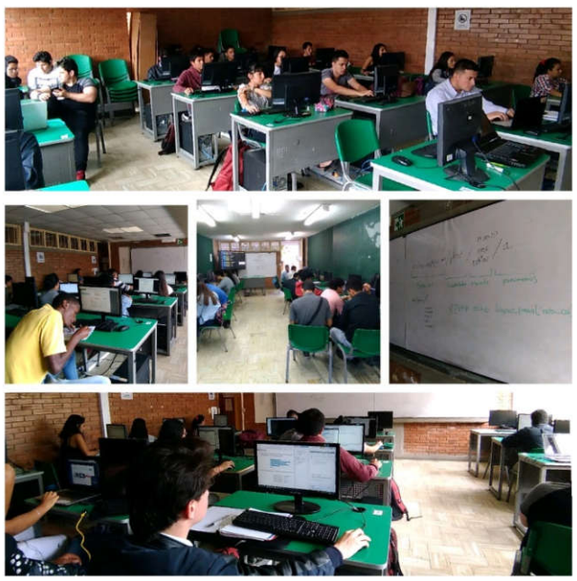


Ilustración 11 Taller Práctico en Universidad de Cundinamarca Fuente: Autor

7.4.2. Validación de la herramienta

Como herramienta para validar el framework, se diseñó un cuestionario en el que se plasmara la experiencia y la percepción de cada uno de los participantes ante la herramienta. El cuestionario fue resuelto por 50 personas.

1. La instalación del framework fue:

Muy sencilla	95%
De dificultad media	5%
Complicada	0%

2. ¿Considera completa la documentación que acompaña el proyecto?

SI	76%
NO	24%

3. De 1 a 10, ¿Cómo evalúa el rendimiento del framework?

   El promedio obtenido fue de **7,8**

4. ¿Qué características le gustaron más del framework?

Velocidad de desarrollo	81%
Facilidad de uso	77%
Estética	48%
Soporte	40%

5. Le parece que el framework Molecular cumple los estándares de seguridad básicos?

SI	70%
NO	30%

6. Cómo evalúa la curva de aprendizaje del framework?

Muy sencilla	95%
De dificultad media	5%
Complicada	0%

7. Le parece innovador?

SI	84%
NO	16%

8. Usaría molecular framework para un proyecto de gran escala?

SI	96%
NO	4%

9. Considera que la estructura del framework basado en moléculas realmente facilita el trabajo en equipo?

SI	92%
NO	8%

10. Recomendaría el framework a un amigo o compañero de trabajo?

SI	100%
NO	0%

7.5. Mantenimiento

Pasada la etapa de implementación y retroalimentación, se encontró que era pertinente hacer algunos ajustes del framework antes de ponerlo en un repositorio público que fuera abierto al público.

Antes de saltar al público, era necesario bautizarlo. Es entonces cuando se decidió por el nombre de **Molecular**, derivado de una de las características más fuertes y llamativas que se identificaron, y es el trabajo modular. Es así como estos módulos que pueden ser provistos por terceros e integrados a cada proyecto, serán conocidos como **moléculas**.

Se elaboró la página oficial donde se enlazaron cada uno de los elementos desarrollados en este proyecto: Framework y Documentación.

Ilustración 12 Pagina web Molecular. Fuente Autor

8. Resultados

- Molecular, se desarrolló como una herramienta para ser implantada en entornos académicos e incentivar a los estudiantes a seguir el camino del desarrollo web, sin embargo se logró ir un poco más allá, y se contó con la experiencia de ser probado en un ambiente laboral. La empresa CNA.NET, quien cuenta con sus oficinas en Fusagasugá, adoptó el framework para desarrollar uno de sus proyectos. La aplicación que desarrollaron fue para el Hospital de San Rafael de Fusagasugá, denominado APS (Atención Prioritaria en Salud).
- El grupo de investigación del SENA - Centro agroecológico y empresarial cuenta con integrantes en sus semilleros del Tecnólogo en Análisis y Desarrollo de Sistemas de Información, para ejecutar el desarrollo del software de sus proyectos, es así como aprendices de esta titulada utilizaron el framework para dos de sus proyectos. Uno se denomina Control Urbano en Línea, y hace parte del semillero SITOPO. El otro se llama Agrocosto y hace parte del semillero ASISNOVA.
- La documentación del framework ha sido parte fundamental a la hora de expandir el uso de la herramienta en más escenarios lejos de la academia.
- La retroalimentación por parte de los usuarios ha sido fundamental para pensar en la continuación del proyecto.

9. Conclusiones

- Molecular es un marco de trabajo que establece una metodología de trabajo favorable para proyectos académicos de desarrollo de software, donde el tiempo de aprendizaje de la herramienta debe ser corto y el desarrollo rápido y colaborativo.
- Molecular constituye una herramienta útil en el cierre de la brecha entre la academia y el mundo laboral al ser un framework que permite abordar proyectos de corto y gran alcance al ser trabajados con la misma dinámica y facilidad.
- Molecular más que una herramienta, convierte en un método o una filosofía de trabajo que el desarrollador adopta, enseñándole una forma eficiente de abordar el problema al cual le quiere construir una solución.
- La validación del framework que se desarrolló en el proyecto, Molecular, dejó ver que es un trabajo interesante y diferente hablando de temas locales, con lo que permite que a través de estos espacios académicos, surja y se construya una comunidad alrededor del mismo.

10. Recomendaciones

- Pese a que la validación del framework se realizó con estudiantes de semestres intermedios de la Universidad de Cundinamarca, es necesario realizar más pilotos de implantación en diferentes instituciones educativas, de forma que se logre un mayor impacto y uso por parte de los aprendices en el desarrollo de software, al igual que su difusión.

- Es necesario establecer un mecanismo de control para la maduración y evolución del framework, de manera que los aportes que realice la comunidad mantengan una misma línea y puedan ser integrados.

11. Bibliografía

Ángel, T. R. (2016). Desarrollo de aplicaciones web con PHP. Marcombo.

Fontela, C. (2003). Programacion orientada a objetos: Técnicas avanzadas de programación. Nueva Libreria.

González, A. G., & Luis, L. G. (2018). Desarrollo y programación en entornos web. Marcombo.

Javier, C. S. (2018). Programación orientada a objetos con C. RA-MA Editorial.

Mateu, C. (2004). Desarrollo de aplicaciones web. UOC.

Moseley, R. (2008). Manual avanzado de desarrollo de aplicaciones Web. Anaya Multimedia.