

**Instituto Tecnológico de Culiacán**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**Servicio Social**

**Horario:** 2:00-6:00P.M.

**Maestro:** Dr. Luis Carlos Santillán Hernández

**Tema:** Fundamento teórico

**Alumnos:**

* Ortiz García Daniel Gerardo
* Plaza Reyes Priscila

**Fecha de entrega:** 5 de febrero de 2015

Índice

[1. Introducción y objetivos 3](#_Toc442388387)

[1.1. ¿Qué es un framework web? 3](#_Toc442388388)

[1.2. Objetivos 3](#_Toc442388389)

[2. Investigación y selección del lenguaje de desarrollo 4](#_Toc442388390)

[2.1. ¿Qué es PHP? 4](#_Toc442388391)

[2.2. Ventajas y desventajas 4](#_Toc442388392)

[2.3. ¿Por qué utilizar PHP y no otras opciones? 4](#_Toc442388393)

[3. Investigación y Comparativa de los Frameworks de Desarrollo 5](#_Toc442388394)

[3.1. CakePHP 5](#_Toc442388395)

[3.1.1. Ventajas CakePHP 5](#_Toc442388396)

[3.1.2. Desventajas CakePHP 6](#_Toc442388397)

[3.2. Symfony 6](#_Toc442388398)

[3.2.1. Ventajas de Symfony 6](#_Toc442388399)

[3.2.2. Desventajas de Symfony 6](#_Toc442388400)

[3.3. Yii 7](#_Toc442388401)

[3.3.1. Ventajas de Yii 7](#_Toc442388402)

[3.3.2. Desventajas de Yii 7](#_Toc442388403)

[3.4. Zend 7](#_Toc442388404)

[3.4.1. Ventajas de Zend 8](#_Toc442388405)

[3.4.2. Desventajas de Zend 8](#_Toc442388406)

[3.5. CodeIgniter 8](#_Toc442388407)

[3.5.1. Ventajas de CodeIgniter 9](#_Toc442388408)

[3.5.2. Desventajas CodeIgniter 9](#_Toc442388409)

[3.6. Prado 9](#_Toc442388410)

[3.6.1. Ventajas de Prado 10](#_Toc442388411)

[3.6.2. Desventajas de Prado 10](#_Toc442388412)

[3.7. PHPDevShell 10](#_Toc442388413)

[3.7.1. Ventajas de PHPDevShell 10](#_Toc442388414)

[3.7.2. Desventajas PHPDevShell 10](#_Toc442388415)

[3.8. Laravel 10](#_Toc442388416)

[3.8.1. Ventajas de Laravel 11](#_Toc442388417)

[3.8.2. Desventajas de Laravel 11](#_Toc442388418)

[3.8.3. Modular y extensible 11](#_Toc442388419)

[3.8.4 Micro-servicios y APIs 11](#_Toc442388420)

[3.8.5 HTTP Routing 11](#_Toc442388421)

[3.8.6 HTTP Middleware 11](#_Toc442388422)

[3.8.7 Caché 12](#_Toc442388423)

[3.8.8 Autenticación 12](#_Toc442388424)

[3.8.9 Integración con Stripe 12](#_Toc442388425)

[3.8.10 Tareas automatizadas 12](#_Toc442388426)

[3.8.11 Encriptación 13](#_Toc442388427)

[3.8.12 Eventos 13](#_Toc442388428)

[3.8.13 Paginación 13](#_Toc442388429)

[3.8.14 Object-Relational-Map (ORM) 14](#_Toc442388430)

[3.8.15 Unit Testing 14](#_Toc442388431)

[3.8.16 Cola de tareas (Queues) 14](#_Toc442388432)

[El Framework más popular de 2015 14](#_Toc442388433)

[Framework Ganador 14](#_Toc442388434)

[Marco ganador por país 15](#_Toc442388435)

[4. Propuesta del Framework 17](#_Toc442388436)

[5. Modelo Vista Controlador (MVC) 17](#_Toc442388437)

[6. Modelo 18](#_Toc442388438)

[7. Vista 19](#_Toc442388439)

[8. Controlador 19](#_Toc442388440)

[9. Referencias y Fuentes de Información 20](#_Toc442388441)

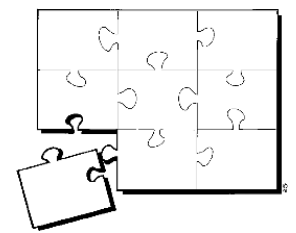
# Introducción y objetivos

En el desarrollo de software, un Framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un "Software Framework" es un diseño reusable de un sistema (o subsistema). Está expresado por un conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software. Todos los frameworks de software son diseños orientados a objetos".

## ¿Qué es un framework web?

Un framework web es una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web. En cierto sentido podemos afirmar que nos proveen una capa de abstracción sobre la arquitectura original ocultándola o adaptándola para no tener que utilizar el protocolo http de manera nativa y así acelerar los tiempos de desarrollo y mantenimiento.



## Objetivos

Los objetivos principales que persigue un framework son:

* Acelerar el proceso de desarrollo.
* Reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.
* Permitir la utilización de toda la infraestructura existente en cada plataforma (bibliotecas de clases, componentes, etc.).
* Extender el tiempo de vida de una implementación a décadas (20, 30, 50 años).
* Desarrollo de software multiplataforma.
* Portabilidad entre plataformas sin pérdida de rendimiento.
* Componentes modulares y abiertos (compilador extensible con "Plugins").
* Múltiples lenguajes de alto nivel como origen.
* Fácil implementación de lenguajes de alto nivel.

# Investigación y selección del lenguaje de desarrollo

El lenguaje de desarrollo escogido es PHP. A continuación explicaremos los motivos que nos han llevado a esta elección.

## ¿Qué es PHP?

PHP son las siglas "Hypertext Pre-processor".

Es un lenguaje de programación pensado en la web de forma que es ideal para la creación de páginas dinámicas. PHP es la versión libre del sistema equivalente de Microsoft ASP.

PHP es un lenguaje encapsulado dentro de los documentos html, de forma que se pueden introducir instrucciones php dentro de las páginas. Gracias a esto el diseñador gráfico del web puede trabajar de forma independiente al programador. PHP es interpretado por el servidor (apache) generando un HTML con el resultado de substituir las secuencias de instrucciones PHP por su salida.

## Ventajas y desventajas

Entre las ventajas podemos citar:

* Se basa en ser un lenguaje multiplataforma.
* Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
* Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
* Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
* Permite las técnicas de Programación Orientada a Objetos. (Clases y herencia.).
* Muy sencillo de aprender.
* Similar en sintaxis a C y a PERL.
* Se puede incrustar código PHP con etiquetas HTML.
* Excelente soporte de acceso a base de datos.
* Se puede hacer de todo lo que se pueda transmitir por vía HTTP.

Y las desventajas:

* Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
* La legibilidad del código puede verse afectada al mezclar sentencias HTML y php.
* La orientación a objetos es aún muy deficiente para aplicaciones grandes.

## ¿Por qué utilizar PHP y no otras opciones?

* PHP no soporta directamente punteros, como en C, de forma que no existen los problemas de depuración provocados por estos.
* El código PHP es mucho más legible que el de PERL.
* Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, xml, creación de PDF...).
* Al poderse encapsular dentro de código html se puede recoger el trabajo del diseñador gráfico e incrustar el código php posteriormente.
* Hay multitud de aplicaciones php para resolver problemas concretos (weblogs, tiendas virtuales, periódicos... ) listas para usar.
* Es multiplataforma, funciona en todas las plataformas que soporten apache.
* Es software libre. Se puede obtener en la web y su código está disponible bajo la licencia GPL.

# Investigación y Comparativa de los Frameworks de Desarrollo

Los frameworks que hemos elegido para realizar la investigación de los mismos son los siguientes: CakePHP, Symfony, Yii, Zend, CodeIgniter, Prado y PHPDevShell.

## CakePHP

CakePHP es un marco de desarrollo (Framework) rápido para PHP, libre, de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web. El principal objetivo es que se pueda trabajar de forma estructurada y rápida, sin pérdida de flexibilidad.

Con CakePHP el desarrollo web ya no es monótono porque ofrece las herramientas para que se pueda empezar a escribir el código que realmente se necesita: la lógica específica de la aplicación.

CakePHP tiene un equipo de desarrolladores y una comunidad activos, lo que añade valor al proyecto. Con CakePHP, el núcleo de la aplicación se mejora constantemente.

### Ventajas CakePHP

* El patrón de desarrallo MVC: se basa en la arquitectura de desarrollo del Modelo-Vista-Controlador.
* Reutilización de código.
* Objetos Mapeo Relacional. CakePHP soporta técnicas de ORM (Object Relational Mapping).
* Ninguna configuración complicada: CakePHP no requiere configuraciones para empezar a utilizarlo.
* Plataforma de código abierto.
* Validación incorporada.
* Comunidad activa y amistosa.
* Licencia flexible.
* Compatible con PHP4 y PHP5.
* CRUD (Crear, Obtener, Actualizar y Borrar) integrado para la interacción con la base de datos.
* Soporte de aplicación (scaffolding).
* Generación de código.
* Arquitectura Modelo Vista Controlador (MVC).
* Despachador de peticiones (dispatcher), con URLs y rutas personalizadas y limpias.
* Validación integrada.
* Plantillas rápidas y flexibles (sintaxis de PHP, con ayudantes).
* Ayudantes para AJAX, Javascript, formularios HTML y más.
* Componentes de email, sookie, seguridad, sesión y manejo de solicitudes.
* Listas de control de acceso flexibles.
* Limpieza de datos.
* Caché flexible.
* Localización.
* Funciona en cualquier subdirectorio del sitio web, con poca o ninguna configuración de Apache.

### Desventajas CakePHP

* Malos hábitos: algunos puntos de su documentación te motivan a realizar algunos malos hábitos que deberían evitarse al trabajar con cualquier tipo de framework MVC, como por ejemplo, con el método find de los modelos.
* Exceso de consultas: en proyectos en los que el número de tablas es muy elevado y hay muchas relaciones, el ORM integrado puede llegar a realizar excesivas consultas.
* Tamaño de ficheros: Algunos de los ficheros del propio framework son demasiados pesados.

## Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.

Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas \*nix (Unix, Linux, etc.) como en plataformas Windows.

### Ventajas de Symfony

* Basado en PHP 5 y, por tanto, lenguaje orientado a objetos.
* Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y \*nix estándares)
* Independiente del sistema gestor de bases de datos
* Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
* Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional
* Sigue la mayoría de mejores prácticas y patrones de diseño para la web
* Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo
* Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo
* Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros

### Desventajas de Symfony

* Primero que nada necesitas por lo menos un VPS para poder publicar tus aplicaciones en la web ya que necesitas tener la habilidad de poder descargar e instalar cosas en tu servidor para que symfony funcione apropiadamente.
* El otro problema de Symfony es el caché. Gran parte de la velocidad de Symfony se debe a un uso extensivo del caché por lo que cuando estás desarrollando tiende a ser algo tedioso tener que estar limpiando el caché de vez en cuando.
* A veces tener tanta flexibilidad son las causas de mal diseño de aplicaciones, pero el que es mal programador va a ser mal programador con cualquier framework.
* Además, los procesos utilizan demasiada memoria.

## Yii

Yii es un framework PHP basado en componentes de alta performance para desarrollar aplicaciones Web de gran escala. El mismo permite la máxima reutilización en la programación web y puede acelerar el proceso de desarrollo. El nombre Yii (pronunciado /i:/) es por fácil (en inglés: easy), eficiente (en inglés: efficient) y extensible (en inglés: extensible).

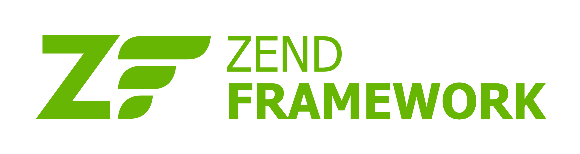
### Ventajas de Yii

* Patrón de diseño Modelo Vista Controlador (MVC).
* Database Access Objects (DAO.
* Integración con jQuery.
* Entradas de Formulario y validación.
* Widgets de Ajax, como autocompletado de campos de texto y demás.
* Personalización de aspectos y temas.
* Generación compleja automática de WSDL, especificaciones y administración de peticiones Web service.
* Esquema de caching por capas.
* El manejo de errores y logging. Los errores son manejados y personalizados, y los log de mensajes pueden ser categorizados, filtrados y movidos a diferentes destinos.
* Las medidas de seguridad incluyen la prevención cross-site scripting (XSS), prevención cross-site request forgery (CSRF), prevención de la manipulación de cookies, etc.
* Generación automática de código para el esqueleto de la aplicación, aplicaciones CRUD, etc.
* Generación de código por componentes de Yii y la herramienta por línea de comandos cumple con los estándares de XHTML.
* Cuidadosamente diseñado para trabajar bien con código de terceros. Por ejemplo, es posible usar el código de PHP o Zend Framework en una aplicación Yii.

### Desventajas de Yii

* Tiempo de aprendizaje. Aprender a utilizar un framework no necesariamente es algo que podamos hacer en un par de días. En mi experiencia si bien ahorramos mucho tiempo, también tenemos que dedicar muchísimas horas en aprender los aspectos básicos y no tanto. A mayor conocimiento del framework, más jugo le podemos sacar.
* Dependencia. Una vez que escribimos un programa para determinado framework es casi imposible migrar a otro sin tener que reescribir todo el código. En ese sentido cuando comenzamos un proyecto con un framework prácticamente nos estamos casando con él.
* Falsa sensación de conocer el lenguaje. Esto es importantísimo porque veo que mucha gente esta confundida en este aspecto. Programar para determinado un framework no nos hace conocedores del lenguaje para el que esta creado. El mejor ejemplo: Saber programar con jQuery no significa saber programar en JavaScript. Mi recomendación, aprendan el lenguaje y luego aprendan el framework.

## Zend

Zend Framework destaca el hecho de que no sólo busca facilitar la programación a través del patrón MVC, sino también automatizar tareas más específicas, como el acceso a base de datos, el filtrado de datos ingresados a la aplicación o la búsqueda en un sitio web ordenando resultados por relevancia.

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5.

Es una implementación que usa código 100% orientado a objetos. La estructura de los componentes es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.

Ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos.

### Ventajas de Zend

* Reduce el "time to market" de las aplicaciones, permitiendo ofrecer presupuestos más ajustados.
* Estandariza los procesos más frecuentes, dotándolos de gran robustez.
* Facilita el mantenimiento de las aplicaciones.
* Ofrece muchas facilidades para el acceso a recursos avanzados, que de otro modo resultan bastante más costosos de desarrollar
* A diferencia de otros frameworks, es posible utilizarlo en modo "desacoplado", es decir, aquellas clases o componentes que sean necesarios en cada proyecto, sin arrastrar todo el framework detrás para cualquier pequeña necesidad.
* Tiene el respaldo de la propia ZEND, creadora de PHP, lo que asegura su continuidad futura tanto como la del propio lenguaje PHP.

### Desventajas de Zend

* Es necesario comprender algunos patrones de diseño y programación orientada a objetos para utilizar todo el potencial de Zend Framework. Y eso no sólo requiere tiempo dedicado al aprendizaje, sino también la experiencia de trabajar con él.
* Es grande, es pesado, gasta mucha memoria y tiene una gran cantidad de inclusiones.

## CodeIgniter

CodeIgniter es un entorno de desarrollo abierto (licencia Open Source Apache/BSD-style) que permite crear webs dinámicas con PHP.

Su principal objetivo es ayudar a que los desarrolladores puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero. Esto se debe a que dispone de un conjunto bastante amplio de librerías útiles para realizar tareas comúnmente necesarias, así como una interfaz simple y una estructura lógica sencilla para acceder a esas librerías.

Entre otras características podemos destacar que es un entorno muy simple. El núcleo del sistema sólo requiere unas pocas librerías para funcionar adecuadamente. Esto supone una gran ventaja frente a otros frameworks de desarrollo que quieren muchos más recursos para realizar las mismas tareas. Las librerías adicionales que se necesiten se cargan de forma dinámica, con lo cual el sistema en sí es muy simple y bastante rápido (está considerado como el framework más rápido, sobretodo en desarrollo bajo PHP).

### Ventajas de CodeIgniter

* Sistema basado en Modelo-Vista-Controlador.
* Compatible con PHP4.
* Muy liviano.
* Clases de base de datos llenas de características con soporte para varias plataformas.
* Formulario y Validación de datos.
* Seguridad y filtro XSS.
* Manejo de sesión.
* Librería de manipulación de imágenes (cortar, copiar, redimensionar...)
* Clase de upload de fichero.
* Clase de FTP.
* Encriptación de datos.
* Scaffolding.
* Clase de codificación Zip.
* Clase de motor de plantillas.
* Ruteo de URI Flexible.
* Larga librería de funciones auxiliares.

### Desventajas CodeIgniter

* No tiene sistema de plantillas.
* No tiene un layout general.
* No hay módulos.
* No hay un ORM.
* Los controladores no cargan por defecto las listas.
* Las vistas no tienen un orden por defecto ( ejemplo, controlador/método).
* Hay algunas cosas que no se pueden configurar, por lo tanto te obligan a modificr el núcleo.

## Prado

PRADO es una capa de programación (framework) basada en componentes y programación dirigida por eventos, para desarrollos de aplicaciones Web en PHP 5. Las siglas PRADO significan en español Desarrollo Rápido de Aplicaciones con Programación Orientada a Objetos en PHP (PHP Rapid Application Development Object-oriented).

El principal objetivo de PRADO es utilizar al máximo la reutilización en la programación Web. Por reusabilidad, queremos decir no solamente reutilizar el código propio, si no el de otros programadores de una manera fácil. Lo último dicho es más importante, ya que evita el esfuerzo de reinventar nuevamente la rueda y además posibilita disminuir notablemente los tiempos de desarrollos.

Para facilitar la interacción con componentes, PRADO implementa el paradigma de la programación dirigida por eventos (event-driven) que permite la delegación de comportamientos extensibles a los componentes.

Métodos o funciones deben ser enlazadas a dichos eventos de tal manera que cuando los eventos sucedan, estos son invocados automáticamente para responder a dicho evento.

### Ventajas de Prado

* Reutilización.
* Programación dirigida por eventos.
* Integración de equipo.
* Controles webs Potentes.
* Fuerte soporte de bases de datos.
* Soporte de AJAX sin fisuras.
* Soporte de I18N y L10N - PRADO incluye soporte completo para crear aplicaciones con múltiples idiomas y la localización.
* Compatibilidad XHTML.

### Desventajas de Prado

* No tiene sistema de plantillas.
* No tiene un layout general.
* No hay un ORM.

## PHPDevShell

PHPDevShell es un framework liberado bajo GNU / GPL el cual se diseñó para que los desarrolladores de PHP que desean empezar a desarrollar con PHP inmediatamente puedan hacer de manera sencilla. Proporciona después de la instalación, autenticación del usuario y el registro, seguridad a nivel de acceso múltiples, múltiples a nivel del sistema de plantillas, y un menú de sistema de nivel múltiple.

PHPDevShell ofrece un avanzado sistema de control de administrador que permite a los desarrolladores a comenzar a concentrarse en su proyecto de inmediato. Con su sistema de plugins, la creación de paquetes de la distribución es una tarea sencilla.

### Ventajas de PHPDevShell

* Soporta PHP5.
* Cuenta con un excelente sistema de plantillas.
* AJAX navegador habilitado cliente.

### Desventajas PHPDevShell

* No tiene un Sistema basado en Modelo-Vista-Controlador.
* No soporta técnicas de ORM (Object Relational Mapping).
* No cuenta con Database Access Objects (DAO).

## Laravel

Laravel es otro framework de PHP brillante que está equipado con un montón de características interesantes, incluyendo enrutamiento RESTful, PHP nativo o atractivo motor ligero y muchos más. Construido con varios componentes de Symfony, laravel ofrece a las aplicaciones web una increíble base de código confiable y bien probado. Algunas otras características interesantes de laravel incluyen: un poderoso conjunto de librerías, un ORM increíble, fácil enrutamiento  y una autenticación simple.

### Ventajas de Laravel

* Adaptable y modificable: Es muy adaptable ya que puedes crear tus vistas tanto en html,  php como en su sistema de plantillas blade. Además, todas las funcionalidades pueden ser accedidas desde cualquier sitio, desde controladores, vistas, modelos, rutas.
* Rápido
* Seguro: Posee un método de encriptación muy potente para la autenticación de usuarios. Posee filtros de seguridad e incluso para ataques por post (csrf) al sitio.
* No tienes que programar en los modelos. Muy importante, pues todo está echo en laravel, solo crear los modelos y listo.

### Desventajas de Laravel

* Capacidad: Debido a los vendors se hace un poco grande como tal el framework.

### Modular y extensible

Laravel es modular y extensible. Esto quiere decir que te permite agregar todo lo que necesitas a través de su directorio [Packalyst](http://packalyst.com/) que cuenta con más de 5,500 paquetes. Esto con el objetivo de que siempre encuentres lo que necesitas.

### Micro-servicios y APIs

Lumen es un micro-framework derivado de Laravel con un enfoque en lean development . Es decir, te permite desarrollar fácil y rápidamente micro-servicios y APIs de gran rendimiento para proyectos. Lumen integra todas las características de Laravel con una mínima configuración y te permite migrar al framework completo con sólo copiar el código en un proyecto de Laravel.

1. <?php $app->get('/', **function**() {
2. **return** view('lumen');
3. });
4. $app->post('framework/{id}', **function**($framework) {
5. $this->dispatch(**new** Energy($framework));
6. });

### HTTP Routing

Laravel cuenta con un sistema de enrutamiento rápido y eficiente, similar al que se usa en Ruby on Rails. Este nos permite relacionar las partes de nuestra aplicación con las rutas que ingresa el usuario en el navegador.

1. Route::get('/', **function** () {
2. **return** 'Hello World';
3. });

### HTTP Middleware

Tus aplicaciones estarán blindadas usando Middleware; ya que se encarga de analizar y filtrar las llamadas HTTP en tu servidor. Puedes instalarlo para que se encargue de verificar que se trate de un usuario registrado, de evitar problemas de tipo Cross-Site-Scripting (XSS) y otras medidas de seguridad.

1. <?php **namespace** App\Http\Middleware; **use** Closure; **class** OldMiddleware { **public** **function** handle($request, Closure $next) { **if** ($request->input('age') <= 200) {
2. **return** redirect('home');
3. }
4. **return** $next($request);
5. }
6. }

### Caché

Tus aplicaciones contarán con un robusto sistema de caché el cual puedes ajustar para que tu aplicación cargue más rápido y, así, ofrecer la mejor experiencia posible a tus usuarios.

1. Cache::extend('mongo', function($app) {
2. **return** Cache::repository(**new** MongoStore);
3. });

### Autenticación

La seguridad es muy importante. Laravel viene listo para implementar autenticación de usuarios de forma nativa e incluye la opción de “recordar” al usuario. Además, te permite incluir parámetros adicionales, lo que nos asegurará, por ejemplo, si se trata de un usuario activo.

1. **if** (Auth::attempt(['email' => $email, 'password' => $password, 'active' => 1 ], $remember)) {
2. // The user is being remembered...
3. }

### Integración con Stripe

Laravel Cashier incluye todo lo necesario para integrar tu desarrollo con este servicio de cobro. Además, este se puede sincronizar e integrar con el sistema de autenticación de usuarios. Así que ya no te tienes que preocupar por cómo integrar un sistema de cobrosa tu desarrollo.

1. $user = User::find(1);
2. $user->subscription('monthly')->create($creditCardToken);

### Tareas automatizadas

Elixir es una API de Laravel que nos permite definir tareas de Gulp con las que podemos definir el uso pre-procesadores para comprimir nuestro CSS y JavaScript .

1. elixir(**function**(mix) {
2. mix.browserify('main.js');
3. });

### Encriptación

Una aplicación segura necesita ser capaz de encriptar sus datos. Con Laravel tienes todo lo necesario para empezar a usar seguridad OpenSSL y cifrado AES-256-CBC. Adicionalmente, todos los valores encriptados están firmados por un código de autenticación de mensaje que detecta si el mensaje encriptado fue alterado.

1. **use** Illuminate\Contracts\Encryption\DecryptException;
2. **try** {
3. $decrypted = Crypt::decrypt($encryptedValue);
4. } **catch** (DecryptException $e) {
5. //
6. }

### Eventos

Definir, registrar y escuchar eventos en tu aplicación es muy sencillo. La propiedad listende EventServiceProvider contiene una lista de todos los eventos registrados en tu aplicación. Además, usa como llave el nombre del evento y como valor los listeners registrados a él.

1. **protected** $listen = [
2. 'App\Events\PodcastWasPurchased' => [
3. 'App\Listeners\EmailPurchaseConfirmation',
4. ],
5. ];

### Paginación

La paginación es muy fácil con Laravel pues genera un rango de enlaces según la página actual en el navegador del usuario.

1. <?php **namespace** App\Http\Controllers; **use** DB; **use** App\Http\Controllers\Controller; **class** UserController **extends** Controller { **public** **function** index() { $users = DB::table('users')->paginate(15);
2. **return** view('user.index', ['users' => $users]);
3. }
4. }
5. <div **class**="container">
6. @foreach ($users **as** $user)
7. {{ $user->name }}
8. @**endforeach**
9. </div>
10. {!! $users->render() !!}

### Object-Relational-Map (ORM)

Laravel incluye una capa para manejo de bases de datos que cuenta con un ORM llamado[Eloquent](http://laravel.com/docs/5.1/eloquent). Este, además, funciona perfectamente con PostgreSQL.

1. $users = User::where('votes', '>', 100)->take(10)->get();
2. **foreach** ($users **as** $user) {
3. var\_dump($user->name);
4. }

### Unit Testing

El desarrollo de Unit Testing es una tarea que consume una cantidad considerable de tiempo. Pero nos da la certeza de que nuestra aplicación funcionará sin problemas. Laravel incluye métodos para realizar Unit Testing usando PHPUnit.

1. <php **use** Illuminate\Foundation\Testing\WithoutMiddleware; **use** Illuminate\Foundation\Testing\DatabaseTransactions; **class** ExampleTest **extends** TestCase { **public** **function** testBasicExample() { $this->visit('/')
2. ->see('Laravel 5')
3. ->dontSee('Rails');
4. }
5. }

### Cola de tareas (Queues)

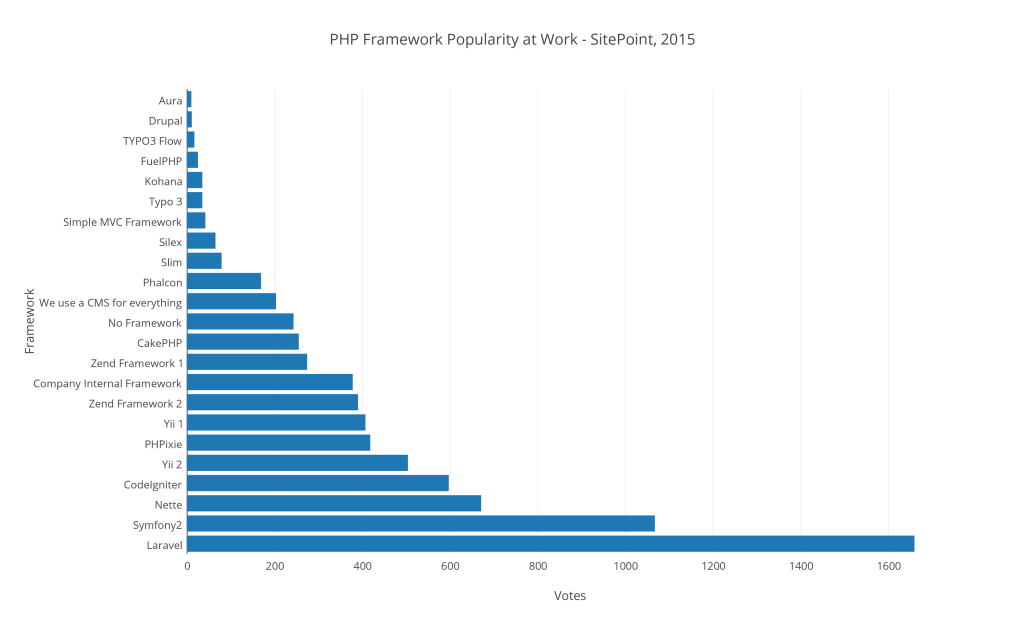
Laravel nos ofrece la opción de ejecutar procesos largos y complejos en segundo plano usando listas de tareas. Esto nos permite trabajar de forma asíncrona con ciertos procesos que no son necesarios para continuar con la navegación del usuario.

1. Queue::push(**new** SendEmail($message));

## **El Framework más popular de 2015**

### **Framework Ganador**

Como era de esperar, laravel ganó por un amplio margen, una vez más.



Marco ganador por país

Si nos fijamos en todos los países con más de 50 votos, estos son sus favoritos:Es una tendencia interesante de observar. La mayoría de los países de habla inglesa favorecen laravel, mientras que Francia es leal a Symfony - es propio producto. Curiosamente, un increíblemente gran porcentaje de los checos (En el segundo país más activo en la encuesta) favorecer Nette - un marco en gran parte desconocido en el mundo occidental, mientras que Ucrania tiene su propio favorito local - PHPixie. Se hace aún más interesante cuando nos fijamos en los cinco primeros de cada país - no sólo el ganador - pero voy a dejar que depende de usted para explorar!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **País** | **Total de votos** | **Trabajo Favorite** | **Votos** | **Favorito Personal** | **Votos** |
| Estados Unidos | 819 | Laravel | 219 | Laravel | 293 |
| República Checa | 770 | Nette | 611 | Nette | 639 |
| Reino Unido | 496 | Laravel | 138 | Laravel | 166 |
| Alemania | 428 | Symfony2 | 76 | Laravel | 100 |
| Francia | 343 | Symfony2 | 149 | Symfony2 | 136 |
| Brasil | 305 | Laravel | 100 | Laravel | 111 |
| India | 287 | Laravel | 62 | Laravel | 77 |
| Ucrania | 263 | PHPixie | 66 | PHPixie | 67 |
| Federación Rusa | 235 | Yii 2 | 53 | Yii 2 | 72 |
| Polonia | 216 | Symfony2 | 52 | Symfony2 | 46 |
| Países Bajos | 209 | Laravel | 64 | Laravel | 84 |
| Rumania | 183 | Symfony2 | 49 | Symfony2 | 48 |
| Canadá | 138 | Laravel | 40 | Laravel | 52 |
| España | 131 | Symfony2 | 47 | Symfony2 | 43 |
| Vietnam | 112 | Laravel | 34 | Laravel | 43 |
| Corrí | 101 | Laravel | 34 | Laravel | 35 |
| Italia | 100 | Laravel | 20 | Laravel | 25 |
| Australia | 99 | Laravel | 30 | Laravel | 39 |
| Eslovaquia | 94 | Nette | 48 | Nette | 47 |
| Bélgica | 79 | Laravel | 26 | Laravel | 31 |

# Propuesta del Framework

Podemos afirmar que los Frameworks representan una arquitectura de software que modela las relaciones generales de las entidades. Proveen una estructura de trabajo la cual extienden o utilizan las aplicaciones.

Todos los frameworks antes mencionados son perfectamente valiosos para el desarrollo de aplicaciones de diversos tipos, compatibles además con PHP5 y capaces de soportar múltiples bases de datos y objetos DB, excepto Prado, que no soporta objetos DB ni MVC. CakePHP, y CodeIgniter pueden de manera adicional ejecutarse utilizando PHP4.

En el caso particular de Symfony, añade una nueva capa por encima de PHP y proporciona herramientas que simplifican el desarrollo de las aplicaciones web. Su comprensión se centra en el conocimiento de conceptos básicos de la programación moderna, sobre todo acerca de la programación orientada a objetos (OOP), el mapeo de objetos a bases de datos (ORM) y el desarrollo rápido de aplicaciones (RAD), teniendo como único requisito técnico para su manipulación el conocimiento previo de PHP 5.

Como elección para nuestro trabajo hemos decidido escoger Laravel Framework. La elección la hemos hecho atendiendo a las características que tiene junto con su adaptabilidad para crear vistas en HTML o PHP y su velocidad para la generación automática de código para el esqueleto de la aplicación además que incluye un potente medio de encriptación que nos da la seguridad al momento de autenticar con los usuarios. Sin contar que también incluye el modelo vista controlador que nos permite gestionar las peticiones de los usuarios para poder procesarlas.

# Modelo Vista Controlador (MVC)

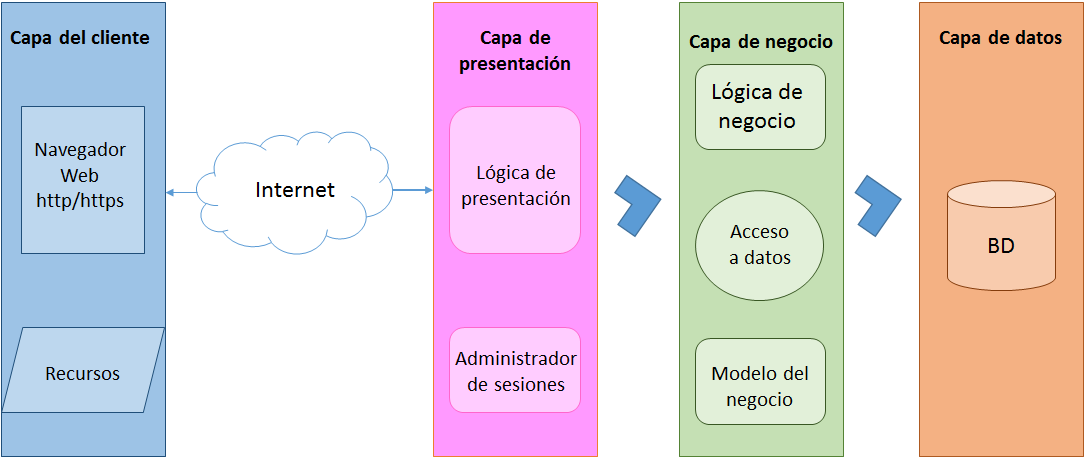
EL patrón **MVC** (Model, View, Controller) o **Modelo**, **Vista** **Controlador**, es un tipo de diseño que separa en capas bien definidas el desarrollo de una aplicación, esas partes son tres, el **Modelo** encargado de la lógica del negocio y la persistencia de los datos, las **Vistas** son las responsables de mostrar al usuario el resultado que obtienen del modelo a través del controlador, el **Controlador** encargado es el encargado de gestionar las peticiones del usuario, procesarlas invocando al modelo y mostrarlas al usuario a través de las vistas.

En pocas palabras, el patrón de diseño MVC organiza el código en base a su función. De hecho, este patrón separa el código en tres capas:

* La capa del **modelo** define la lógica de negocio (la base de datos pertenece a esta capa). Como ya sabes, Symfony guarda todas las clases y archivos relacionados con el modelo en el directoriolib/model/.
* La **vista** es lo que utilizan los usuarios para interactuar con la aplicación (los gestores de plantillas pertenecen a esta capa). En Symfony la capa de la vista está formada principalmente por plantillas en PHP. Estas plantillas se guardan en varios directorios llamados templates/ repartidos por todo el proyecto, tal y como veremos hoy mismo.
* El **controlador** es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Cuando instalamos Symfony el primer día, explicamos que todas las peticiones se canalizan a través de los controladores frontales (index.phpy frontend\_dev.php). Estos controladores frontales realmente delegan todo el trabajo en las **acciones**. Como vimos ayer, las agrupaciones lógicas de acciones se denominan **módulos**.



**Figura 4.1** Arquitectura MVC



**Figura 4.2** Arquitectura MVC

### Modelo

Laravel incluye un sistema de mapeo de datos relacional llamado Eloquent [ORM](https://es.wikipedia.org/wiki/ORM) que facilita la creación de modelos. Este ORM se funda en patrón active record y su funcionamiento es muy sencillo. Es opcional el uso de Eloquent, pues también dispone de otros recursos que nos facilitan interactuar con los datos, o específicamente la creación de [modelos](https://es.wikipedia.org/wiki/Modelos).[7](https://es.wikipedia.org/wiki/Laravel#cite_note-eloquent-7)

La forma de crear Modelos en Laravel usando Eloquent ORM, es tan simple como:

*class Libro extends Eloquent { //defiendo el nombre de la tabla con la info de los libros*

*public static $table = 'tb\_libros';*

*}*

Incluso el código del modelo puede ser más simple aún, en el caso que el nombre de la tabla coincida con el nombre de la clase. Pues Laravel al igual que [Ruby on Rails](https://es.wikipedia.org/wiki/Ruby_on_Rails), usa el paradigma de programación donde se favorece "la convención sobre la configuración".Y si ahora necesitamos disponer un listado, en la ruta: http://mi-aplicacion.com/libro/listar Entonces, sólo bastaría crear la Ruta e interactuar con el modelo 'Libro' anteriormente creado, del modo siguiente:

*Route::get('libro/listar', function() {*

*$libros = Libro::all();*

*return View::make('mi\_vista', $libros); //Muestra los datos*

*});*

### Vista

Laravel incluye de paquete un sistema de procesamiento de plantillas llamado Blade. Este sistema de [plantillas](https://es.wikipedia.org/wiki/Plantillas) favorece un código mucho más limpio en las Vistas, además de incluir un sistema de [Caché](https://es.wikipedia.org/wiki/Memoria_cache) que lo hace mucho más rápido. El sistema Blade de Laravel, permite una sintaxis mucho más reducida en su escritura.[4](https://es.wikipedia.org/wiki/Laravel#cite_note-plantillas-4) Por ejemplo, en vez pintar la vista usando el código PHP:

*<?php echo $mi\_nombre; ?>*

En Blade se escribiría:

{{$mi\_nombre}}

### Controlador

Los controladores contienen la lógica de la aplicación y permiten organizar el código en clases sin tener que escribirlo todo en las rutas. Todos los controladores deben extenderse de la clase BaseController.

Un ejemplo de un controlador en Laravel:

*class UserController extends BaseController {*

*public function mostrarPerfil($id)*

*{*

*$user = User::find($id);;*

*return View::make('user.profile', array('user' => $user));*

*}*

*}*

Estos pueden ser llamados en las rutas de diferentes maneras, pero la más común usándo rutas es:

*Route::get('user/{id}', 'UserController@mostrarPerfil');*

# Referencias y Fuentes de Información

* <http://book.cakephp.org/1.3/es/contents.html>
* <http://librosweb.es/symfony_1_2/capitulo_1/symfony_en_pocas_palabras.html>
* <http://www.pradosoft.com/demos/quickstart/index.php?page=GettingStarted.AboutPrado&notheme=true&lang=es>
* <http://www.phpframeworks.com/top-10-php-frameworks/>
* <http://pwneds.blogspot.com.es/2007/11/comparacion-de-10-frameworks-de-php.html>
* <http://db.grinnell.edu/sigcse/sigcse2013/Program/viewAcceptedProposal.pdf?sessionType=paper&sessionNumber=116>
* <http://www.youtube.com/watch?v=w-3x7bIj6pc>
* <http://www.youtube.com/watch?v=QD-NuVImBDA>
* <http://www.tucamon.es/contenido/frameworks-en-php-por-que-usarlos>
* <http://es.wikipedia.org/wiki/Categor%C3%ADa:Frameworks_para_PHP>
* <http://tuxpuc.pucp.edu.pe/articulo/comparativa-de-frameworks-en-php-cakephp-symfony-y-zend-framework>
* <http://www.kabytes.com/programacion/usar-o-no-un-framework/>
* <http://pavithrakathirvel.wordpress.com/2010/01/20/top-10-php-frameworks/>
* <http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/>