Frameworks PHP

Un **framework** es una herramienta que proporciona una serie de módulos que ayudan a organizar y desarrollar un producto software. En el caso concreto de los frameworks PHP, la mayoría de ellos proporcionan una serie de comandos o herramientas para crear proyectos con una estructura determinada (normalmente, siguiendo el patrón MVC que veremos después), de forma que ya dan una base de trabajo hecha, y facilidades para poder crear el modelo de datos, la conexión a la base de datos, las rutas de las diferentes secciones de la aplicación, etc.

1. Ejemplos de frameworks PHP

Actualmente existe una gran variedad de frameworks PHP que elegir para desarrollar nuestras aplicaciones. Algunos de los más populares son:

- **Laravel**, un framework relativamente reciente (fue creado en 2011), y que ha ganado bastante popularidad en los últimos años. Su filosofía es el poder desarrollar proyectos de forma elegante y simple. Cuenta con una amplia comunidad de soporte detrás, y se le augura un futuro bastante consolidado.
- **Symfony**, creado en 2005, cuenta con más camino hecho que Laravel, y una estructura más consolidada. En sus primeras versiones se presentaba como un framework más monolítico (se instalaban demasiados módulos que luego no necesitábamos), pero recientemente ha adaptado su estructura para hacerla más modular. De hecho, podríamos considerar Symfony como un *metaframework*, es decir, un framework que, a su vez, sirve para desarrollar otros frameworks. Prueba de ello es que, por ejemplo, Laravel utiliza Symfony como base para ampliar esas funcionalidades.
- **CodeIgniter**, un framework más ligero que los anteriores, pero también con un amplio grupo de seguidores y desarrolladores. Fue creado en 2006 y, aunque ha sufrido una etapa de abandono, ha vuelto a coger fuerza en los últimos años, quizá debido a su simplicidad de uso.
- **CakePHP**, creado en 2005, es otro framework similar a CodeIgniter en cuanto a simplicidad y facilidad de uso, aunque con menor popularidad.
- **Zend**, creado en 2006, es otro framework bastante popular, aunque quizá con menor visibilidad que los anteriores hoy en día, a la altura de CakePHP.
- **Phalcon**, otro framework de reciente creación (2012), con una potente capacidad de procesamiento de páginas PHP, y la posibilidad de trabajar como microframework (más ligero, para ofrecer funcionalidades muy específicas) o como framework completo. De hecho, muchos frameworks más antiguos también han incorporado recientemente la posibilidad de ejecutarlos como microframeworks.
- ... etc.

Casi todos los frameworks PHP tienen una serie de características comunes, como son el uso del patrón MVC para desarrollar sus proyectos, la inyección de dependencias para gestionar recursos tales como conexiones a bases de datos, o elementos compartidos por toda la aplicación, la posibilidad de desarrollar tanto webs completas como servicios REST accesibles desde diversos clientes, etc.

2. ¿Cuál elegir?

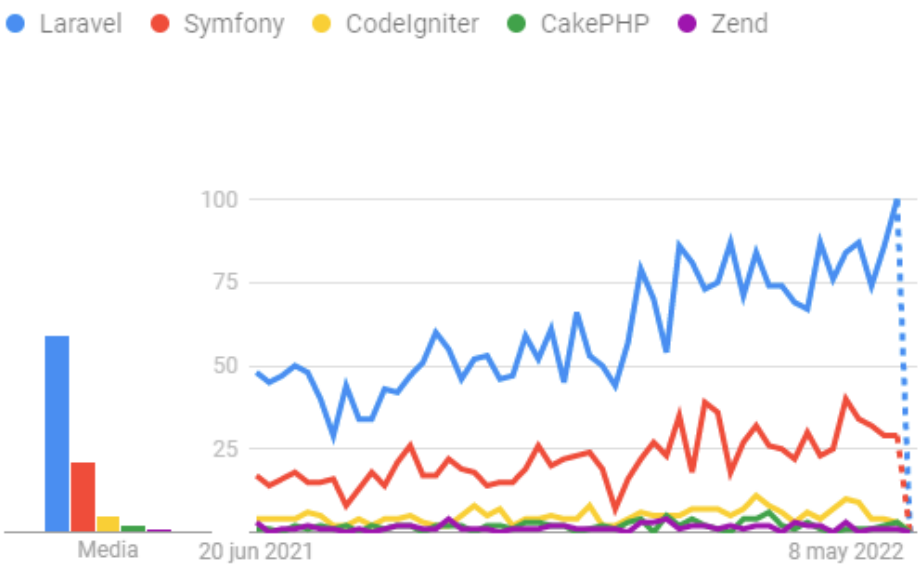
A la hora de decantarnos por uno u otro framework, no nos deberíamos dejar engañar por la popularidad del mismo, en términos de cuota de mercado. En ese terreno, Symfony y Laravel probablemente sean los más beneficiados, pero la curva de aprendizaje en ellos puede que sea más pronunciada que en otros a priori más sencillos, como CodeIgniter o CakePHP.

Cada framework puede estar mejor orientado que otro para determinados tipos de proyectos o necesidades. Si queremos aprender algo rápido para lanzar la aplicación cuanto antes mejor, quizá Symfony no sea la mejor opción. Si, por el contrario, preferimos empaparnos de un framework con una comunidad importante detrás que nos pueda dar soporte y nos garantice un tiempo de vida largo, entonces Symfony o Laravel pueden ser mejores candidatos.

Llegados a este punto... ¿qué características debemos mirar a la hora de decantarnos por uno u otro framework? Quizá algunas de las más importantes (y no necesariamente excluyentes entre sí) son:

Popularidad

La popularidad la podemos medir en base a diferentes webs estadísticas. Por ejemplo, si comparamos las búsquedas en *Google Trends* de los principales frameworks PHP, podemos determinar cuáles son los más buscados a nivel mundial:



Demanda laboral

Otro factor determinante para elegir un framework de desarrollo es la demanda laboral que tiene, las puertas que se nos pueden abrir al aprenderlo. Por ejemplo, si hacemos una búsqueda en el portal español de

búsqueda de empleo *InfoJobs* de algunos frameworks PHP, a fecha de *Junio de 2022*, obtenemos estos datos aproximados:

Framework	Ofertas encontradas
Laravel	107
Symfony	76
CodeIgniter	15
Zend	4
CakePHP	3

Facilidad de aprendizaje

Es otro factor importante a tener en cuenta, especialmente cuando el tiempo de que disponemos para realizar el proyecto es escaso. En este sentido, frameworks como Symfony o Laravel suelen ser más "pesados de digerir", precisamente por su envergadura y la cantidad de opciones que ofrecen, mientras que otros más livianos como CodeIgniter o Zend son más fácilmente asimilables.

Soporte y documentación

Es importante analizar la documentación y la comunidad de desarrollo y soporte que hay detrás de cada framework, para saber si vamos a poder resolver dudas sobre su uso con facilidad. Por ejemplo, Laravel y Symfony, que son dos de los frameworks más difundidos, cuentan con una gran comunidad detrás (especialmente Laravel), y una documentación muy completa y actualizada.

- Documentación de Laravel
- Documentación de Symfony

Extensiones o *plug-ins*

También puede resultar un dato relevante el conocer la capacidad de ampliación de un framework, qué otras funcionalidades adicionales se le pueden incluir en caso necesario. Muchos de los frameworks PHP pueden hacer uso de la herramienta *composer* que veremos en otras secciones para instalar dependencias o módulos externos, y enriquecer así la funcionalidad de la aplicación.

En el caso de algunos frameworks, ya vienen con ciertas funcionalidades o extensiones incorporadas que suponen un valor añadido. Por ejemplo, Laravel incorpora un motor de plantillas llamado Blade para desarrollar fácilmente vistas HTML, así como un ORM llamado Eloquent para trabajar con bases de datos relacionales como si fueran objetos. Symfony también hace lo propio con el motor de plantillas Twig y el ORM Doctrine, respectivamente.

En realidad, una vez se conoce uno de estos frameworks, es más sencillo asimilar el resto, llegado el momento. Así que cualquiera de ellos, con unos motivos que se ajusten a nuestras necesidades, puede ser un buen punto de partida.

3. El patrón MVC

Como hemos comentado anteriormente, la gran mayoría de frameworks PHP se apoyan en el patrón MVC. MVC son las siglas de *Modelo-Vista-Controlador* (o en inglés, *Model-View-Controller*), y es el patrón por excelencia ahora mismo en el mundo de las aplicaciones web, e incluso muchas aplicaciones de escritorio.

Como su nombre indica, este patrón se basa en dividir el diseño de una aplicación web en tres componentes fundamentales:

- El **modelo**, que podríamos resumir como el conjunto de todos los datos o información que maneja la aplicación. Típicamente serán variables u objetos extraídos de una base de datos o cualquier otro sistema de almacenamiento, por lo que el código del modelo normalmente estará formado por clases o elementos donde almacenar los datos que extraigamos o vayamos a almacenar en esa base de datos. Generalmente, el modelo no tendrá conocimiento del resto de componentes del sistema.
- La **vista**, que es el intermediario entre la aplicación y el usuario, es decir, lo que el usuario ve en pantalla de la aplicación. Por lo tanto, la vista la compondrán las diferentes páginas, formularios, etc, que la aplicación mostrará al usuario para interactuar con él.
- El **controlador** (o controladores), que son los fragmentos de código encargados de coordinar el funcionamiento general de la aplicación. Ante peticiones de los usuarios, las recogen, las identifican, y utilizan el modelo para actualizar o recuperar datos, y a su vez, deciden qué vista mostrarle al usuario a continuación de la acción que acaba de realizar.

Es un patrón de diseño muy conciso y bien estructurado, lo que le ha valido la fama que tiene hoy en día. Entre sus muchas ventajas, permite aislar los tres elementos involucrados (vista, modelo y controlador), de forma que el trabajo es mucho más modular y divisible, pudiendo encargarse de las vistas, por ejemplo, un diseñador web que no tenga mucha idea de programación en el servidor, y del controlador un programador PHP que no tenga muchas nociones de HTML.

En forma de esquema, podríamos representarlo así: