Sobre la enseñanza de PHP en el ámbito universitario.

José Antonio Álvarez Ocete

Durante los últimos años, una de las cuestiones más repetidas en el ámbito de la programación web, tanto en el mundo empresarial como en la academia, ha sido si debería enseñarse PHP a futuras generaciones de ingenieros informáticos. Esta pregunta nace por un lado gracias a los nuevos lenguajes de programación en el ámbito del servidor que han surgido en las últimas décadas, y por otro lado por los problemas de diseño que PHP acarrea desde sus inicios. En este breve ensayo mostraremos distintos puntos de vista sobre esta cuestión, e intentaremos responder a la pregunta inicial.

En primer lugar, estudiemos el descontento general por parte del público con PHP. Dicho lenguaje fue creado en 1995 con el *boom* de Internet. Sin embargo, no es de los lenguajes de programación más antiguos. Por ejemplo, C++ fue creado en 1980, Python en 1990 y Java en 1995 [1]. Si no es comparativamente más antiguo, ¿por qué se dice que PHP está desfasado o tiene problemas en su diseño cuando sus semejantes temporales carecen de los mismos en elementos similares? Si el lector no está familiarizado con estos problemas de diseño puede encontrar algunos de los más usuales explicados técnicamente en [2].

Algunos de estos problemas han sido corregidos con el tiempo en las nuevas versiones de PHP. Esto se debe a la gran comunidad de software libre que mantiene hoy en día este lenguaje [3]. Pero el rechazo de la comunidad informática hacia el mismo no se debe únicamente a sus problemas de diseño. Al nacer PHP pocos años después que el World Wide Web, es uno de los primeros lenguajes que se utilizó en el desarrollo de sitios y posteriormente aplicaciones web. Debido a esto, ciertas prácticas de programación se fueron desarrollando sobre la marcha en este ámbito. Con el nacimiento de la Web 2.0 en el cambio de siglo, los desarrolladores se fueron dando cuenta de que la programación web podía funcionar con los mismos principios y técnicas de programación que ellos bien conocían. Se produjo entonces una migración paulatina de la forma de desarrollo de este tipo de aplicaciones a prácticas más conocidas, apareciendo nuevos frameworks dedicados a la web a partir de conocidos lenguajes como Ruby on Rails en 2004 y Django en 2005 a partir de Python [4].

A pesar de ello, ya existían muchos sitios web creados con malas prácticas y grandes problemas que han tenido que mantener o seguir desarrollándose a partir de las mismas. Estos problemas de legado hacen que aquellos desarrolladores que trabajen o hayan trabajado con versiones antiguas de PHP sufran particularmente este lenguaje.

Finalmente, PHP tiene grandes problemas de seguridad [5]. Si bien las más relevantes pueden ser paliadas por un desarrollador con detallado conocimiento del lenguaje [6], esto requiere de dicho

conocimiento y experiencia, y claramente no todo el mundo dispone de dicho trasfondo con este lenguaje.

Y sin embargo, a pesar de todos sus problemas, PHP es a día de hoy el lenguaje más utilizado con diferencia en el ámbito del servidor en desarrollo web. Por mencionar algunos ejemplos: Facebook, Wikipedia, Yahoo y Wordpress utilizan PHP hoy en día [7]. Explorando las herramientas de tendencias de Google y Stackoverflow podemos observar la popularidad en búsquedas de este lenguaje en los últimos años [8] [9].

Algunas de las causas de la popularidad de PHP ya han sido comentadas. Por ejemplo, su gran comunidad de software libre ha generado buena documentación y una gran cantidad de especialistas que controlan los detalles de este lenguaje. Además, es gratis utilizarlo. Esto significa que, combinado con las tecnologías adecuadas se puede desplegar una aplicación web full stack con costes mínimos. Por ejemplo, utilizando una pila LAMP: Apache HTTP para el servidor web sobre un sistema Linux con MySQL y PHP [10].

Dentro de este último ejemplo yace subyacente otra de las razones por las que PHP es tan ampliamente popular hoy en día: la facilidad de integración con otras tecnologías y herramientas. Esto incluye desde un amplísimo abanico de bases de datos (relacionales como mySQL, SQLite o PostgreSQL; o no relacionales como Redis o MongoDB), cómoda integración con HTML y otras tecnologías de desarrollo en el ámbito del cliente como Javascript, Typescript o Angular.

Caben destacar dos tecnologías particulares con las que PHP tiene sencilla integración y que dirigen el futuro del desarrollo web y la sociedad en su totalidad. En primer lugar, este lenguaje se integra a la perfección con los proveedores de servicios Cloud Computing más famosos del mercado: Google Cloud, Amazon Web Services y Azure App Services [11]. En segundo lugar, es sencillo crear y utilizar imágenes de contenedores que utilizan PHP. Las imágenes oficiales de *Docker* con PHP se pueden encontrar en [12].

Si bien estos son argumentos de peso sobre por qué utilizar o no hoy en día PHP en el ámbito de la programación web, la tesis original de este texto tiene matices adicionales. No se está decidiendo si se debería utilizar PHP en la próxima aplicación que desarrollemos como empresa, sino si debería enseñarse PHP en primera instancia, como aproximación inicial al desarrollo en el ámbito del servidor para el alumnado.

Por un lado, este lenguaje se puede aproximar de forma directa por parte de los alumnos: es sencillo de utilizar e integrar con otras tecnologías de primera aproximación obligatoria, como son HTML y Javascript, y tiene una sintaxis familiar para el alumnado. No hace falta comentar la utilidad directa que tiene PHP en el mundo empresarial hoy en día, pues ya ha sido ampliamente discutida con anterioridad.

Por otro lado, ¿podría merecer la pena aprender a utilizar otras tecnologías más recientes y novedosas directamente? Estudiando y conociendo PHP en primer lugar, apreciamos las carencias del mismo, y ello nos hace mejores desarrolladores en última instancia.

Hay una analogía que me parece especialmente relevante: Estudiar C/C++ o Python como primera aproximación a la programación en su totalidad. Python es un lenguaje de más alto nivel: es más sencillo de entender y utilizar, sobre todo para un estudiante que nunca ha conocido la programación. Sin embargo, Python utiliza punteros bajo la superficie. Son transparentes para el programador y puede generar grandes errores difíciles de entender si no se maneja este concepto adecuadamente.

Esta decisión depende de la educación completa que vaya a recibir el estudiante. Para un Ingeniero Informático, entender adecuadamente ciertos conceptos como los punteros y el paso por referencia es clave en su formación a largo plazo. Y dicha comprensión se alcanza de manera más directa y sencilla aprendiendo C++ a pesar de que pueda parecer más complicado.

De la misma forma, un ingeniero informático que aprende PHP puede entender y extrapolar detalles de bajo nivel o de seguridad que pueden ser menos directos en otros lenguajes. Si bien no se tiene suficiente tiempo para explorar otras alternativas y poder compararlas adecuadamente dentro del ámbito de la carrera universitaria, si queda plantada la semilla para el futuro del desarrollador.

En este texto hemos expuesto y discutido las razones históricas del rechazo del lenguaje PHP, y cómo a pesar de ello sigue siendo uno de los lenguajes más utilizados en su ámbito. En vista de la evidencia recaudada, es claro que la respuesta a la cuestión original no es ni mucho menos sencilla y directa. Hay amplias razones por las PHP debe ser enseñado hoy en día en el ámbito universitario, y amplias razones por las que podrían enseñarse otras tecnologías alternativas. Lo que sin duda es indiscutible es la presencia y popularidad de dicho lenguaje en la sociedad actual, y la utilidad del mismo en el futuro.

Bibliografía

[1]: Wikipedia: Historia de los lenguajes de programación.

[2]: Blog: Why does PHP suck.

[3]: Repositorio en GitHub de PHP y actividad sobre el código fuente de PHP en los últimos meses.

[4]: Discusión en Reddit sobre el rechazo a PHP.

[5]: Vulnerabilidades de seguridad en PHP.

[6]: Blog: Las vulnerabilidades más famosas de PHP y cómo prevenirlas

[7]: Wikipedia: Los lenguajes de programación utilizados en los sitios webs más famosos.

[8]: Stackoverflow Trends: los lenguajes más famosos en los últimos años.

[9]: Google Trends: comparativa con los lenguajes más famosos y otros frameworks de desarollo web.

[10]: Blog: Por qué usar PHP - principales ventajas y desventajas.

[11]: Documentación oficial sobre PHP y Cloud Computing en Google Cloud, Amazon Web Services y Azure App Services.

[12]: Documentación oficial: Imágenes oficiales de PHP en Docker.