# EVOLUCIÓN DE LAS TECNOLOGÍAS DE DESARROLLO WEB

(COMPLEMENTA AL APARTADO 1.2 - TECNOLOGÍAS ASOCIADAS A LAS APLICACIONES WEB)

## 1. PRINCIPALES PLATAFORMAS DE DESARROLLO

El desarrollo web ha evolucionado a través de distintas tecnologías. Si bien existen numerosas alternativas a la hora de elegir una tecnología para implementar una aplicación web, en la actualidad, la mayoría de aplicaciones web se implementan sobre una de las tres principales plataformas de desarrollo: PHP, Java y .NET.

### 1.1. COMMON GATEWAY INTERFACE (CGI)

Esta plataforma fue pionera en el desarrollo web, pero, a día de hoy, se considera obsoleta. No obstante, algunos hostings todavía proporcionan scripts de automatización bajo esta tecnología.

### 1.2. PHP: HYPERTEXT PREPROCESSOR (PHP)

PHP es una tecnología de desarrollo web basada en código embebido en lenguaje de marcas. Desde un punto de vista formal y metodológico, plantea ciertas desventajas respecto a otras plataformas:

- Es un lenguaje de scripting **embebido en lenguaje de marcas**, por lo que resulta difícil (o imposible) separar la aplicación en capas, delimitando claramente sus responsabilidades.

- Es un lenguaje es de **tipificación débil**, lo que resulta desfavorable desde un punto de vista metodológico, ya que dificulta la lectura del código y resulta más propenso a errores.

- **No es orientado a objetos** (aunque simula algunos comportamientos de la orientación a objetos, de manera muy peculiar).

- Es un lenguaje **interpretado**, lo que afecta, por un lado, al rendimiento y, por otro limita las posibilidades de depuración.

- No ofrece soporte nativo para muchos de los paradigmas actuales, como la arquitectura MVC, el desarrollo guiado por pruebas, el análisis estático o la implementación de determinados patrones. Para disponer de alguna de estas características es necesario recurrir a frameworks de terceros (Code igniter, Larabel, Django...)

No obstante, pese a todas estas desventajas, es posiblemente la plataforma de desarrollo más extendida, en parte por su arraigo histórico.

## 1.3. JAVA (API JSP/Servlets)

Java es una tecnología de desarrollo madura. Entre sus características destacan estos aspectos:

- Permite el desarrollo de todo tipo de aplicaciones: en consola, en ventana, para dispositivos móviles y también de aplicaciones web y servicios.

- En desarrollo web, el API JSP/Servlets, combinado con beans, permite, de manera nativa separar la aplicación en capas.

- El lenguaje Java es de tipado fuerte y orientado a objetos.

- No dispone de soporte nativo para algunos de los paradigmas actuales y, en estos casos, es necesario recurrir a frameworks específicos (JavaServerFaces, Spring, Hibernate, NUnit, etc.)

En general, Java ofrece desarrollos robustos y fiables, fáciles de diseñar, depurar y modificar. Sin embargo, el lenguaje impone pocas restricciones formales por lo que la responsabilidad de un buen diseño recae sobre el equipo de desarrollo. Por ejemplo, el lenguaje "se traga" antipatrones como la definición de una enumeración en una interfaz. O, por poner otro ejemplo, sería posible diseñar una aplicación web íntegramente con JSP, al estilo PHP, sin hacer uso de Servlets o Beans.

Una cuestión muy importante a tener en cuenta es que una aplicación escrita en Java **puede ejecutarse en el servidor o en el cliente**, dependiendo del tipo de aplicación (aunque en los apuntes induce a confusión ya que especifica que sólo se ejecuta en el cliente, sin contextualizar más). De este modo, dependiendo del tipo de aplicación desarrollada, podemos tener:

- Aplicaciones en ventana o en consola, que se ejecutan de manera convencional en un único equipo (arquitectura centralizada o arquitectura cliente, dependiendo del tipo de aplicación).

- Aplicaciones web desarrolladas con el API JSP/Servlets, que se despliegan y ejecutan en el servidor y NUNCA en el cliente. De igual modo, los servicios HTTP (implementados con SOAP+XML o cualquier otra tecnología similar) se ejecutan en el servidor.

- Aunque hoy en día su utilización es prácticamente nula debido a sus limitaciones, Java permite la creación de ***applets***, pequeñas aplicaciones embebidas en una página web, que se descargan y se ejecutan en el cliente, de manera similar a como lo haría un script de Javascript.

## 1.4. ASP *CLÁSICO*

ASP (clásico) es, al igual que CGI, una tecnología obsoleta. Microsoft abandonó hace años el desarrollo de esta plataforma en favor de la plataforma ASP.NET. Conviene en este punto diferenciar claramente ASP (clásico) del actual ASP.NET ya que ambas tecnologías no están relacionadas y entre ambas median muchos años de diferencia.

ASP clásico era un lenguaje de scripting embebido en lenguaje de marcas, lo que lo hacía muy parecido a PHP. Además, estaba orientado a la utilización de componentes ActiveX. El lenguaje de scripting utilizado era VBscript y no se disponía de un soporte orientado a objetos. Tampoco se daba soporte a los actuales paradigmas de desarrollo web, como MVC. En la actualidad puede considerarse una plataforma de desarrollo residual.

## 1.5. ASP .NET *WEBFORMS*

En el año 2002, con la introducción de la plataforma .NET, (cuya sección de desarrollo web se denomina ASP.NET) se introduce un nuevo paradigma de desarrollo web, los *Webforms*. Con esta tecnología se hace posible separar las capas de presentación, aplicación y acceso a datos, utilizando cualquiera de los lenguajes de la plataforma (a la cabeza de los cuales se sitúa C#). Webforms, además, introduce un ciclo de vida de la aplicación basado en eventos y controles de servidor.

## 1.6. ASP .NET *MVC* y ASP .NET *Core*

ASP.NET evoluciona hacia el paradigma MVC (arquitectura ***Modelo-Vista-Controlador***) ofreciendo un nuevo modelo de programación basado en estándares y buenas prácticas. Además, ASP.NET se convierte en un **stack de código abierto (no propietario)** mantenido por una extensa comunidad de empresas, investigadores y entusiastas. Con ASP.NET MVC y su adhesión al código abierto llega, además de la arquitectura MVC, el desarrollo guiado por pruebas, el análisis estático, los framework de pruebas automatizadas, ejecución paralela en el servidor y ejecución paralela no-concurrente, un modelo unificado para desarrollar servicios HTTP y aplicaciones web, inversión de control mediante inyección de dependencias, aplicaciones web en tiempo real, middleware, etc.

En la actualidad, ASP.NET mantiene dos branches en desarrollo activo y de código abierto: **ASP.NET MVC**, en versión 4.7, ligado todavía a sistemas Windows y al servidor IIS, y el nuevo **ASP.NET Core**, que es un stack totalmente reescrito desde cero y orientado a su ejecución en **todo tipo de servidores** (IIS, Apache o nginx) y **sistemas operativos** (Linux, MacOs, Windows). Ambas tecnologías constituyen una excelente opción de desarrollo y, desde un punto de vista formal y metodológico son pioneras y muy superiores a otras tecnologías más extendidas. En ambos casos se dispone de las siguientes características:

- .NET permite el desarrollo de todo tipo de aplicaciones, en consola, en ventana, para dispositivos móviles y también de aplicaciones web y servicios.

- Se dispone de un conjunto de lenguajes (como C# o F#) y se puede programar en cualquiera de ellos (incluso combinándolos), con idéntico rendimiento. La utilización de uno u otro lenguaje sólo difiere en la sintaxis específica, ya que las APIs son compartidas.

- Permiten, de manera nativa, dar soporte a los actuales paradigmas de programación, sin necesidad de recurrir a frameworks de terceros.

- Todos los lenguajes de la plataforma son orientados a objetos y de tipado fuerte, aunque soportan otros modos, como la tipificación dinámica y la tipificación implícita.

- La plataforma y los lenguajes son restrictivos: sólo es posible hacer las cosas de una manera, la correcta.

## 2. LENGUAJES DE SCRIPTING EN EL CLIENTE

Respecto a los lenguajes de script, ni VBscript ni JScript existen en la actualidad en desarrollos web. Javascript es, a día de hoy, la realidad que se ejecuta en el lado del cliente.

En este sentido, es conveniente no confundir Java con Javascript. Java (JSP/Servlets) se ejecuta en el servidor, es un lenguaje orientado a objetos, de tipado fuerte y compilado a máquina virtual. Por el contrario, Javascript es un lenguaje estructurado, sin orientación a objetos pura (la simula, también de forma muy peculiar), interpretado (no compilado), de tipificación débil y cuyos scripts se ejecutan en el cliente.

Existen algunas ramificaciones curiosas, como Typescript, que es un intento por dotar de cierta tipificación a Javascript.

Dada la disparidad en las formas de implementar Javascript por parte de los distintos navegadores y de la relativa complejidad para realizar determinadas implementaciones, se han popularizado numerosas APIs que ofrecen la realización de tareas habituales de manera simplificada (JQery es un ejemplo).