Tecnologías y herramientas para el desarrollo web (1)

1. Introducción

La ingeniería de *software* es una rama de la ingeniería relativamente nueva, y la ingeniería *web* de *software* más nueva aún. De esta última se ha discutido desde finales de los años noventa y principios del 2000 respecto a los retos que se presentan con el desarrollo de este tipo de software, asícomo también sobre la necesidad de adaptar procesos convencionales del desarrollo de software a este ámbito.

Desde estas primeras disertaciones e inquietudes a la actualidad han surgido propuestas de metodologías para el desarrollo de *software* basadas en el modelo incremental en espiral y como también alternativas o adaptaciones del estándar UML (Unified Modeling Language) para el diseño de *software web*, tales como UWE (UML-based Web Engineering), I para el desarrollo de aplicaciones *web,* porque las metodologías tradicionales no separan la metodología de modelamiento del *front-end* a la del *back-end.*

Además, con las tecnologías actuales usadas para el desarrollo de aplicaciones *web*, se ha desacoplado los lenguajes implementados en el *back-end*, como PHP, Java, Ruby, Python, Lua, entre otros; y del *front-end,* tales como HTML5, CSS3, Javascript y AJAX, siendo estas tecnologías con las que interactúa el usuario (O'Reilly, 2007), diferenciándose así de las aplicaciones de escritorio que usan las mismas tecnologías tanto en el cliente como en el servidor.

Por último, tenemos estándares de calidad dentro de la industria del *software* (ISO/ IEC, 2006), los cuales brindan una referencia como marco de procesos, aunque no explican cómo implementar estos procesos en un ambiente de desarrollo de aplicaciones *web*.

Software engineering	Web engineering
El sistema de software tiene un pequeñorango de usuarios.	WebApps tiene un amplio rango de usuarios.
Los requerimientos del usuario son específicos.	Los requerimientos del usuario cambian con el tiempo.
El crecimiento y el cambio son reducidos.	Cambia rápidamente.
Los presupuestos de desarrollo varían enun amplio rango según el tamaño de la empresa.	Los presupuestos de desarrollo son reducidos.
El tiempo de desarrollo es más largo.	El tiempo de desarrollo es más corto.
Las restricciones de los entornos de hardware y software son específicas.	Las restricciones de los entornos de hardware y software no son específicas.
La experiencia en diseño y desarrollo esescasa.	La experiencia en diseño y desarrollo está disponible en una amplia gama.
La seguridad y las cuestiones legales noson muy importantes.	La seguridad y las cuestiones legales noson muy importantes.

2. Marco teórico

2.1. Frameworks

Los *frameworks* proveen una implementación del andamiaje para el desarrollo completo de una aplicación, facilitando la reutilización de componentes presentes en la estructura. Proporcionan una serie de puntos donde se pueden acoplar funcionalidades adicionales. No son un patrón arquitectural pero sí una colección de patrones de diseño y clases trabajando en conjunto, que tienen como fin resolver un problema específico. Por esta razón, se cuenta con una gran variedad de *frameworks* en distintos lenguajes de programación ya sea en PHP, Java, Ruby, Pyton, Javascript, entre otros.

2.2. Librería

El concepto de "librería" es más antiguo que el de *framework*. Las librerías consisten en una colección de clases o métodos que proveen comportamiento a otra aplicación. También se diferencian de los *frameworks* en que no se especializan en un control de flujo de datos interno, uso de herencia y patrones de diseño. Dentro de estas librerías tenemos por ejemplo Jquery y Mootools como librerías de Javacript y Twitter Bootstrap como librería de CSS.

2.3. Editor de código fuente

Permite ver y editar archivos propios de una aplicación. Muchos de ellos ofrecen al programador, según el lenguaje de programación que se esté editando, ayuda para la edición del código, ayuda visual en el uso de palabras reservadas del lenguaje, asistencia automática para la identificación, navegación dentro de los archivos y carpetas, entre otros. Esta funcionalidad está integrada dentro de los IDEs (como Netbeans, Eclipse, Visual Studio, etc.), pero, a diferencia de un IDE, los editores de código fuente como Sublime Text, Brackets, Pluma no tienen fácil instalación de *plugins*, opciones de compilación, *debugers*, detección de errores de sintaxis, entre otros.

2.4. Microservicios

Los microservicios son pequeños, autónomos e independientes servicios que trabajan juntos para reemplazar el uso de una aplicación desarrollada bajo una arquitectura monolítica. La forma de despliegue de estos microservicios es a manera de plataforma como servicio (PAAS), pues cada uno puede estar en un sistema operativo diferente, como también cada uno puede ser implementado en lenguajes de programación distintos.

2.5. Frontend

El frontend son aquellas tecnologías de desarrollo web del lado del cliente, es decir, las que corren en el navegador del usuario y que son básicamente tres: HTML, CSS y JavaScript. El frontend se enfoca en el usuario, en todo con lo que puede interactuar y lo que ve mientras navega. Una buena experiencia de usuario, inmersión y usabilidad son algunos de los objetivos que busca un buen desarrollador frontend, y hoy en día hay una gran variedad de frameworks, preprocesadores y librerías que ayudan en esta tarea.

Backend es aquello que se encuentra del lado del servidor y se encarga de interactuar con bases de datos, verificar maniobras de sesiones de usuarios, montar la página en un servidor y servir todas las vistas creadas por el desarrollador frontend. En este caso el número de tecnologías es mucho menos limitado, puesto que la

programación backend puede alcanzar lenguajes como PHP, Python, .NET, Java, etc., y las bases de datos sobre las que se trabaja pueden ser SQL, MongoDB, MySQL, entre otras.

La idea de esta abstracción es mantener separadas las diferentes partes de un sistema web o software para tener un mejor control. En pocas palabras, el objetivo es que el frontend recoja los datos y el backend los procese.

Estas dos capas que forman una aplicación web son independientes entre sí (no comparten código), pero intercambian información. Esta división permite que el acceso a las bases de datos solo se haga desde el backend y el usuario no tenga acceso al código de la aplicación, mientras que la programación del lado del cliente permite que el navegador pueda, por ejemplo, controlar dónde el usuario hace clic o acceder a sus ficheros.

Con esta separación de entornos el usuario de una aplicación web lo que hace es, por ejemplo, iniciar sesión escribiendo su usuario y contraseña en un formulario; a continuación los datos se envían y el backend toma esta información que viene desde el HTML y busca las coincidencias de usuario en la base de datos con una serie de procesos invisibles para el usuario. En este punto, el servidor mandaría un mensaje al frontend dándole acceso (o no) a la aplicación.

A pesar de que hay varios lenguajes que se usan en el frontend, nosotros de momento nos fijaremos en tres: HTML, CSS y JavaScript (aunque HTML y CSS no son len- guajes de programación).

HTML es un lenguaje de marcado de los contenidos de un sitio web, es decir, para designar la función de cada elemento dentro de la página: titulares, pá- rrafos, listas, tablas, etc. Es el esqueleto de la web y la base de todo.

CSS es un lenguaje de hojas de estilo creado para controlar la presentación de la página definiendo colores, tamaños, tipos de letras, posiciones, espaciados, etc.

JavaScript es un lenguaje de programación interpretado que se encarga del comportamiento de una página web y de la interactividad con el usuario.

Aparte, junto al cliente también tenemos los frameworks, las librerías, los pre-procesadores, los plugins... pero todo gira alrededor de HTML, CSS y JavaScript.

2.6. Backend

Aquí encontramos unos cuantos, por ejemplo, PHP, Python, Rails, Go, C#, Java, NodeJS (JavaScript)... entre otros. Como vemos, mientras que para el frontend se acostumbra a trabajar solo con tres lenguajes, en el backend hay unos cuantos más. Por suerte, un desarrollador backend no necesita saberlos todos.

Quizás habréis notado que tenemos JavaScript tanto por el lado del cliente como por el lado del servidor. JavaScript se creó originalmente como lenguaje para el frontend, pero los últimos años se ha creado su lugar dentro del backend con NodeJS, un motor que interpreta JavaScript en el servidor sin necesidad de un navegador.

Esto no quiere decir que el JavaScript que tenemos en el cliente tenga alguna conexión con el que se encuentra en el servidor: cada uno corre por su parte de manera independiente. El JavaScript del cliente corre en el navegador y no tiene ningún enlace ni ninguna conexión con el que hay en el servidor y no le interesa saber cómo está montada la arquitectura del servidor ni cómo se conecta a la base de datos.

Ahora se puede utilizar el mismo lenguaje en todos los contextos del desarrollo: JavaScript en el cliente de escritorio (DOM), en el cliente móvil (Cordova, React Native), en el servidor (Node.js) o en la BBDD (MongoDB). La posibilidad de trabajar frontend y backend con un mismo lenguaje desde el punto de vista del desarrollador es muy cómoda, especialmente para aquellos que trabajan ambos mundos.

En cuanto a la tecnología, las herramientas que se utilizan en el backend son: editores de código, compiladores, algunos depuradores para revisar errores y seguridad, gestores de bases de datos y algunas otras cosas.

Uno de los stacks4 más utilizados por los desarrolladores es el que se conoce por LAMP: Linux, Apache, MySQL y PHP. Cualquier web hecha con Wordpress, Drupal o Prestashop, por ejemplo, están hechas sobre estos cuatro pilares.

Pero se pueden hacer las variaciones que se crean convenientes, puesto que muchas de estas tecnologías son intercambiables por otras similares. Por ejemplo NginX en lugar de Apache, PostgreSQL en lugar de MySQL o Ruby on Rails en lugar de PHP.

Otro stack muy utilizado es el llamado MEAN, que se compone de MongoDB, Express, Angular y NodeJS. A diferencia del conjunto anterior, esta pila de trabajo busca entregar la mayor cantidad de carga junto al cliente pero requiere una forma muy diferente de pensar las cosas.

2.7. Ventajas del frontend y el backend

Mayor escalabilidad

Se trata de un entorno mucho más escalable, al estar separado, es posible que una de las dos partes necesite más recursos en un algún momento, por lo que se hace más sencillo y eficaz dividir los recursos.

Distintos equipos de desarrollo

Al tener estas dos partes que se diferencian, puedes tener en tu equipos diversos perfiles dedicados únicamente a su parte del trabajo.

Ampliación de plataformas disponibles

Imagina que tienes una web con backend y frontend separados; si requieres desarrollar una app nativa para dispositivos móviles, el desarrollo de la misma será mucho más sencillo, ya que solamente te enfocarías en desarrollar la app.

La API de comunicación ya la tendrías desarrollada y quizás solamente requeriría de pequeños ajustes, además, en el proceso de desarrollo, la parte front de tu web existente, no se vería afectada en ningún momento.

Las migraciones y actualizaciones son más sencillas

Normalmente, cuando ya está desarrollado el backend, lo que más se suele cambiar es la parte frontend, por esa razón, si te piden actualizaciones de diseño en la web o cambio de alojamiento, es más fácil, ya que la parte de servidor siempre va a estar funcionando por muchas actualizaciones que se tengan en la parte visual.

2.8. ¿Qué diferencias tienen frontend y backend?

Es verdad que se determinan por las funciones que realizan en un sitio web, pero hay similitudes y diferencias importantes que te presentaremos a continuación.

Conceptos diferentes

Las diferencias entre frontend y backend resultan evidentes, como se mencionó anteriormente, una es la parte del sitio web con la que puedes ver e interactuar, mientras que en la parte trasera, se engloba el funcionamiento estructural y no es visible por el cliente.

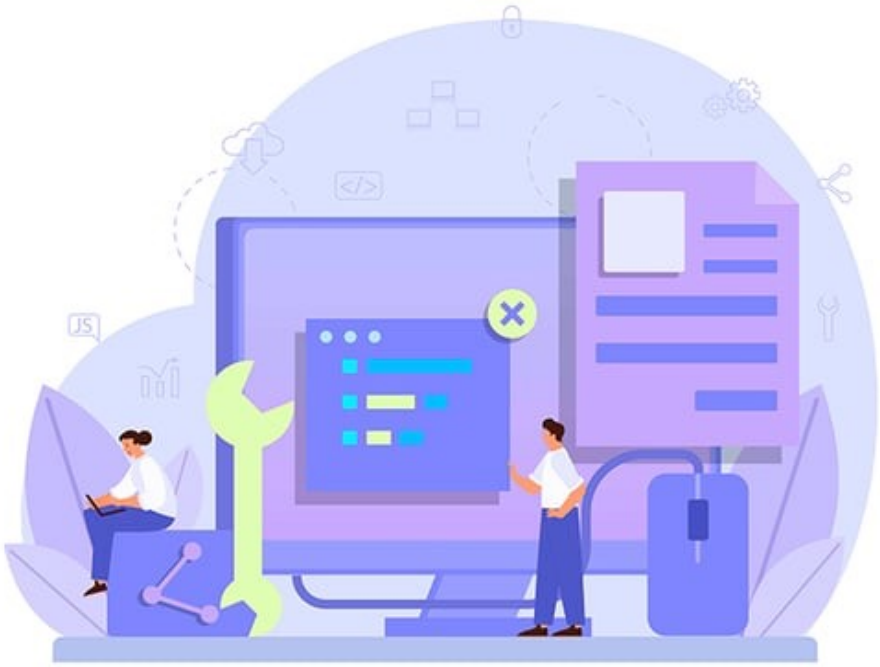
Roles distintos

Frontend tiene que ver con aspectos visuales con los que experimenta cualquier persona, en el backend se atribuye todo lo que sucede en segundo plano porque facilita la interacción web.

Funciones de los desarrolladores de Frontend y Backend

Un diseñador web se encarga de construir sitios web teniendo en cuenta los aspectos visuales.

Los desarrolladores web de backend se aseguran de que los datos y sistemas solicitados por el software funcionen de manera eficiente.



2.9 Frontes vs Backend

¿Qué es mejor Backend o Frontend?

Ninguna es más importante que la otra, se complementan y necesitan mutuamente para funcionar, ya que por una parte, los sitios web deben ser bonitos y amigables y por la otra parte ese mismo sitio debe ser funcional y debe poder realizar operaciones.

En el frontend se necesitan personas con capacidad creativa y en el backend se necesitan personas con muy buena lógica para resolver problemas.

Considera que una web se conforma por una gran variedad de documentos que se relacionan entre ellos por medio de enlaces, lo que significa que si quieres entrar a una web y escribes la dirección URL en el navegador, se

traduce como que estás solicitando que se muestre dicha página web.

Lo siguiente que hará el dispositivo es verificar qué servidor de software tiene el sitio. El servidor recibe esta información, verifica la petición que hizo y te permite ejecutar la acción.

En ocasiones, se puede presentar el caso de que no se requiere una conexión a la base de datos, por ejemplo, accedes a una página y cuando inicias sesión, se hace de manera automática una petición que conecta a la base de datos para verificar los accesos y la suscripción que se tiene, es allí donde el backend devuelve la respuesta al servidor.

Después aparece el frontend, que es quien va a recibir la información que transmitió el backend y la va a acomodar en la interfaz del sitio web o perfil del usuario.

2.10 Desarrolladores Full Stack

Lo más habitual es que los desarrolladores se especialicen bien con frontend o bien con backend, pero hay una tercera especie que son los llamados «Full- Stack», unos programadores con un perfil técnico muy completo que conocen bien tanto lo referente a backend como a frontend, así como la administración de servidores. Tienen un buen conocimiento de todas las áreas del desarrollo y esto les permite construir proyectos complejos por ellos mismos, sin la ayuda de terceras personas. Se trata de un perfil cada vez más demandado y bien remunerado.

Es normal que las empresas quieran conseguir los mejores trabajadores con el mínimo coste posible, pero este tipo de conocimientos no es trivial, sino que se consigue después de muchos años de práctica. Por lo tanto, no existe (o no tendría que existir teóricamente) un perfil de desarrollador «Full Stack Junior».