Introducción al frontend y backend

PID_00250214

Anna Ferry Mestres

Universitat Oberta de Catalunya

Índice

1. Introducción

El frontend son aquellas tecnologías de desarrollo web del lado del cliente[1], 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.

(1)En programación se conoce como «cliente» a la persona humana que está haciendo algo, y «servidor», los procesos que suceden de manera transparente al usuario.

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.

2. Lenguajes web frontend

A pesar de que hay varios lenguajes que se usan en el frontend, nosotros nos fijaremos en tres[2]: HTML, CSS y JavaScript (aunque HTML y CSS **no** son lenguajes de programación[3]).

(2)Hay otros lenguajes frontend, como por ejemplo ActionScript, Java, Silverlight, VBScript u otros lenguajes XML, pero por varios motivos se usan muy poco en comparación con los tres con los que trabajaremos.

(3)No se debe confundir lenguajes de programación (como por ejemplo JavaScript, ActionScript o Java) con lenguajes de marcado (como HTML) o el lenguaje de hojas de estilo, que es el CSS.

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, párrafos, 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 preprocesadores, los plugins... pero todo gira alrededor de HTML, CSS y JavaScript.

3. Lenguajes web 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 stacks[4] 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.

(4)Un stack es un anglicismo que significa 'pila de cosas'. Cuando hablamos de un stack de desarrollo nos referimos a la pila de tecnologías que se utilizan en un entorno de desarrollo determinado (en este caso sería la suma del sistema operativo del servidor, el lenguaje de programación del backend y el software de base de datos).

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.

También existe un equivalente en Microsoft que sería Windows + Microsoft IIS + .NET + SQL Server.

4. 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».