

¿Cómo convertirte en Desarrollador Web?

Todo lo que necesitas saber para iniciar o continuar tu aprendizaje



Introducción

El desarrollo Web es uno de los campos más emocionantes de la actualidad. Servicios como [Netflix](#) y [Spotify](#) han reemplazado a gigantes como Blockbuster y Tower Records. [Amazon](#) ha cambiado la forma en que compramos. [Uber](#) y [AirBnB](#) están cambiando la forma en que nos transportamos y nos alojamos. Las empresas están migrando a sistemas en "la nube" con modelos basados en suscripción. Las agencias de viajes físicas han prácticamente desaparecido.

No existe una sola industria que no haya sido o no esté siendo transformada por Internet. Es una revolución sin precedente, pero este es solo el comienzo. Sectores como la educación, la política y la salud, entre muchos otros hasta ahora están empezando a ser transformados. En los próximos años seremos testigos de enormes cambios, cambios de los que tu puedes ser parte también.

Cualquier persona que pueda usar un computador se puede convertir en Desarrollador Web. A diferencia de lo que muchos creen no necesitas ser bueno en matemáticas ni tener un cerebro especial. Tampoco necesitas gastar una fortuna y asistir a una universidad por 4 años. Necesitas es ser perseverante, optimista y determinado(a).

Esta guía te va a mostrar cómo convertirte en Desarrollador Web: la mentalidad que necesitas cultivar, las tecnologías que debes aprender, por dónde empezar y cómo encontrar tu primer empleo.

Si tienes alguna pregunta o comentario no dudes en escribirme a german.escobar@makeitreal.camp. Intentaré responderte lo antes posible.

Nota: La ilustración de la carátula es una adaptación de un diseño creado por [Freepik](#), un sitio donde puedes encontrar ilustraciones e imágenes gratis para tus proyectos.

¿Qué hace un Desarrollador Web?

Un **desarrollador Web** se encarga de los aspectos técnicos de un sitio o de una aplicación Web. A diferencia del **diseñador Web**, que se encarga de la apariencia visual y la distribución de los elementos en el diseño, el **desarrollador Web** convierte un diseño estático (una imagen) en una aplicación completamente funcional disponible en Internet.

Existen varios roles en el desarrollo Web. Un **desarrollador Web** puede suplir uno o varios de estos roles al mismo tiempo.

Frontend

El frontend es la persona que se encarga de convertir un diseño (una imagen) en código que entiendan los navegadores usando HTML, CSS, y JavaScript.

Este es quizá el rol con más demanda en la actualidad y el que más rápido está evolucionando. La razón es que las empresas buscan que sus sitios y aplicaciones se adapten a diferentes pantallas, tengan animaciones, se actualicen sin necesidad de refrescar la página, reaccionen rápidamente al usuario, etc. Muchas funcionalidades que antes se hacían en el servidor (por el backend) ahora se hacen en el navegador y son responsabilidad del frontend.

Backend

El backend se encarga de conectar la aplicación con la base de datos y otros servicios externos. Son los responsables de la autenticación y autorización de usuarios, lectura y escritura de información de la base de datos, envío de correos automatizados, tareas recurrentes y API's, entre otros.

Operaciones

Las personas que ejecutan este rol se encarga de desplegar la aplicación en Internet y monitorearla. Tambbién se encargan de manejar los servidores, hacer copias de seguridad de la base de datos y en general estar atentos al correcto funcionamiento de la aplicación en producción.

A los desarrolladores que hacen todos los roles se les conoce como Full Stack Developers. Aunque todavía hay mucha demanda por Full Stack Developers, la tendencia es a la especialización.

Independientemente si deseas especializarte en alguno de estos roles, nuestra recomendación es que igual aprendas un poco de cada uno.

¿Cuál es la mentalidad que necesita un Desarrollador Web?

Al ser un campo relativamente nuevo, el desarrollo Web está evolucionando constantemente. La característica más importante de un Desarrollador Web es ser capaz de adaptarse y aprender nuevas tecnologías rápidamente.

Por otro lado, es importante entender que la programación es una herramienta para resolver problemas. Cada problema es diferente de los demás, cada problema requiere su propio aprendizaje y tiene su propia dificultad. En general, si quieres convertirte en Desarrollador Web te debe gustar -o debes desarrollar el gusto de- enfrentarte a nuevos retos y desafíos.

¿Qué necesito aprender para convertirme en Desarrollador Web?

Antes de empezar a hablar sobre tecnologías particulares es importante, y quizá obvio, decir que te debe gustar la tecnología. Debes manejar bien un computador y conocer sus partes básicas (procesador, memoria RAM, disco duro, etc). Debes entender a grandes rasgos qué es un dominio, una IP, un hosting, un sistema operativo y una base de datos, entre otros conceptos relacionados.

Necesitas entender algo de Inglés, lo básico. Sin embargo, ten en cuenta que entre mejor tu Inglés, mejores oportunidades puedes encontrar.

Ahora si, hagamos un recorrido muy rápido por las tecnologías que debes aprender. No te preocupes si no entiendes de qué estamos hablando, eventualmente lo harás.

Lo mínimo que debes aprender es HTML y CSS, que son los lenguajes con los que se crean las páginas Web. También necesitas un editor de texto. Te recomendamos descargar uno diseñado para programadores como [VSCode](#) o [Atom](#), en vez del que viene por defecto con tu sistema operativo.

El siguiente paso es aprender un lenguaje de programación. Existen varias opciones para empezar pero yo las reduciría a las siguientes: JavaScript, Ruby o Python. JavaScript no es el más fácil de aprender pero es el más popular y es el único lenguaje de programación que entienden los navegadores. Ruby y Python son más amigables pero eventualmente tendrás que aprender también JavaScript.

Para convertirte en un desarrollador Web necesitas aprender a manejar la **línea de comandos**, también conocida como "DOS", "consola" o "terminal". La línea de comandos era la única forma de interactuar con un computador cuando no existían interfaces gráficas, pero los programadores la seguimos utilizando.

Una herramienta importante que debes aprender a manejar es un **sistema de control de versiones**, preferiblemente Git. Un sistema de control de versiones guarda el historial de cambios del código y te permite trabajar sobre el mismo proyecto con más personas de forma simultánea.

Para crear aplicaciones Web tipo [Twitter](#), [Facebook](#), etc. necesitas escoger y aprender un **framework Web** para el servidor (backend). Tu decisión dependerá del lenguaje de programación que hayas escogido. Por ejemplo, el framework Web más popular de Ruby es [Ruby on Rails](#), el más popular de Python es [Django](#) y el más popular de Node.js (JavaScript) es [Express.js](#). Más adelante profundizaremos sobre estas opciones. También necesitarás aprender a manejar una o varias Base de Datos como [MySQL](#), [PostgreSQL](#), [MongoDB](#), etc.

En el lado del navegador (frontend) la tendencia es crear aplicaciones que nunca se refrescan llamadas SPA's (Single Page Applications). Muchos argumentan que ofrecen una mejor experiencia al usuario y se han creado innumerables frameworks con ese fin. Todos son basados en JavaScript, claro. Entre los más populares actualmente están [React](#), [Vue](#) y [AngularJS](#). Si quieres especializarte en front-end deberás aprender al menos dos de estos frameworks. En cualquier caso es buena idea aprender al menos sobre [jQuery](#), que es la librería de JavaScript más básica para manipular HTML, escuchar

eventos (clicks, teclas, etc.), hacer animaciones e interactuar con servidores remotos a través de AJAX desde el navegador.

Por último, es importante que conozcas las opciones que existen para desplegar tus aplicaciones en Internet como [Heroku](#), [AWS](#), [Netlify](#), etc.

¿Mucha información? Profundicemos un poco más en cada una de estas tecnologías y el rol que desempeñan en la creación de aplicaciones Web.

HTML y CSS

HTML y CSS son quizá las tecnologías más básicas que necesitas aprender como Web Developer. Con HTML (Hyper Text Markup Language) se define la estructura de una página Web y con CSS (Cascading Style Sheets) los estilos (color de fondo, márgenes entre los elementos, tamaño de las fuentes, bordes, etc.).

Hay muchas formas de hacer lo mismo con HTML y CSS, y la estrategia de muchos es "hacerlo funcionar". Es una buena estrategia. Pero si quieres especializarte en frontend concéntrate en aprender CSS muy bien.

Adicionalmente, deberás aprender sobre un framework de HTML y CSS como [Bootstrap](#), [Foundation](#) o [Semantic UI](#). Si no sabes cuál escoger empieza por [Bootstrap](#) que es por mucho el más popular.

¿Qué te ofrece un framework HTML y CSS como [Bootstrap](#)? Primero, estilos básicos para los elementos de HTML como tipografía, tablas, formularios, imágenes, etc. Segundo, una grilla para que organices los elementos en la página fácilmente y que se adapten al tamaño de la pantalla. Por último, un

catalogo de componentes que la mayoría de sitios necesitan como barras de navegación, pestañas, mensajes de alertas, modales, barras de progreso, entre muchos otros componentes.

Una desventaja de utilizar un framework HTML y CSS como [Bootstrap](#) es que todos los sitios se empiezan a verse igual y no es tan fácil personalizarlo.

Un lenguaje de programación

Aprender un lenguaje de programación no es como aprender HTML y CSS. Esta es la parte en que más personas se frustran porque lo más difícil no es aprender la sintaxis (las reglas) del lenguaje, lo difícil es aprender la lógica del programador, es decir, cómo usar el lenguaje para solucionar problemas.

Si este es tu primer acercamiento con la programación te recomendamos los siguientes recursos que están diseñados para niños pero te van a ayudar a desarrollar la lógica del programador:

- <http://code.org/>
- <https://scratch.mit.edu/>

Veamos ahora algunos lenguajes con sus fortalezas y debilidades.

JavaScript

La ventaja de JavaScript es que está en todas partes. JavaScript es el único lenguaje de programación que entienden actualmente los navegadores, así

que muchas veces no hay opción. También lo han adaptado para el backend con una plataforma llamada [Node.js](#).

La desventaja de JavaScript es que es un lenguaje que ha ido evolucionando en el tiempo y esto puede causar confusión al principio. Por otro lado la sintaxis es más compleja comparada con Ruby y Python, así como el manejo de código asíncronico.

Ruby y Python

La sintaxis de estos dos lenguajes es muy parecida. Es clara y muy amigable para el principiante. Son lenguajes pensados en la productividad del programador, más que en el rendimiento.

Ruby fue muy popular algunos años pero ha perdido fuerza en relación con JavaScript. Python, por otro lado, es muy popular para realizar tareas con servidores y ahora para análisis de datos e inteligencia artificial, aunque también es posible crear aplicaciones Web con Python.

La desventaja de estos lenguajes es que son relativamente lentos y en Windows no son tan fáciles de configurar como en Linux y Mac (que muchas veces ya vienen instalados).

PHP

PHP fue uno de los primeros lenguajes enfocados en la Web y aún existen muchas aplicaciones sobre este lenguaje. Wordpress, que es el CMS (Sistema de Administración de Contenidos) más popular de la actualidad está hecho

sobre PHP. Facebook, una de las aplicaciones más usadas actualmente también está hecha con PHP.

La desventaja de PHP es que no es un buen lenguaje de programación y muchos empleos de PHP son para mantener sitios muy antiguos.

Java y C-Sharp (.NET)

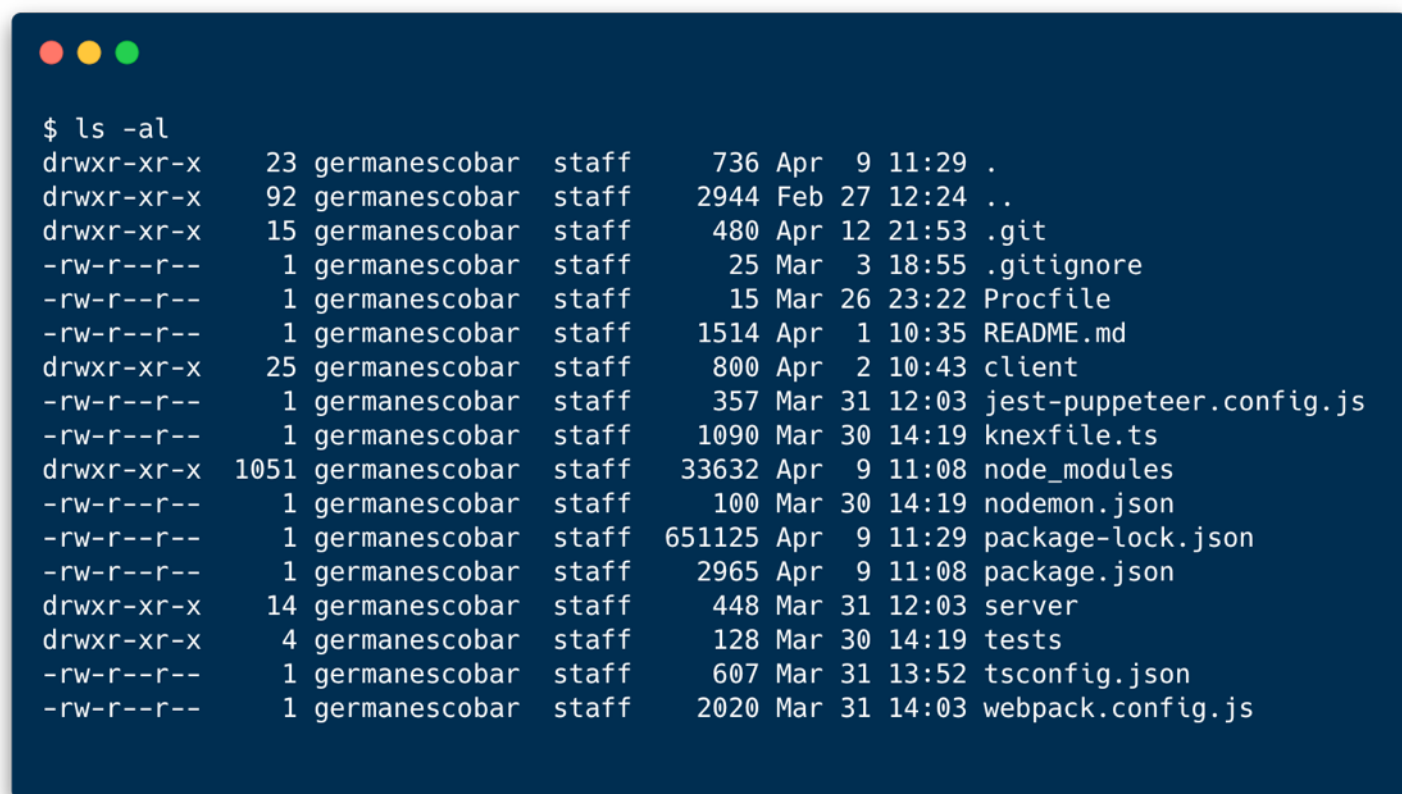
Estos son lenguajes muy populares a nivel empresarial. También son los lenguajes que hoy siguen enseñando las universidades. La ventaja es que, a diferencia de Ruby y Python, son lenguajes con un muy buen rendimiento.

La desventaja es que no están pensados en la productividad del programador. Grandes compañías (Oracle y Microsoft respectivamente) esperan lucrarse de alguna forma u otra de estos lenguajes (generalmente cobrándoles a grandes empresas por soluciones sobre dimensionadas que no necesitan).

En todos estos lenguajes vas a encontrar trabajo. Para Java y C-Sharp es posible que te pidan títulos o certificados por el tipo de empresas que contratan para estos lenguajes. Pero en general, vas a notar que las empresas valoran más lo que has creado y si verdaderamente tienes un interés por la programación.

La línea de comandos

La línea de comandos es un aplicación que nos permite escribir y ejecutar comandos sobre el sistema operativo sin necesidad de usar el mouse. Los programadores la siguen usando porque permite automatizar tareas rápidamente sin necesidad de crear interfaces gráficas. También es la forma de interactuar con servidores remotos. Muchas de las aplicaciones que usan los Desarrolladores Web son aplicaciones para la línea de comandos.



```
$ ls -al
drwxr-xr-x  23 germanescobar  staff    736 Apr  9 11:29 .
drwxr-xr-x  92 germanescobar  staff   2944 Feb 27 12:24 ..
drwxr-xr-x  15 germanescobar  staff    480 Apr 12 21:53 .git
-rw-r--r--   1 germanescobar  staff     25 Mar  3 18:55 .gitignore
-rw-r--r--   1 germanescobar  staff     15 Mar 26 23:22 Procfile
-rw-r--r--   1 germanescobar  staff   1514 Apr  1 10:35 README.md
drwxr-xr-x  25 germanescobar  staff    800 Apr  2 10:43 client
-rw-r--r--   1 germanescobar  staff    357 Mar 31 12:03 jest-puppeteer.config.js
-rw-r--r--   1 germanescobar  staff   1090 Mar 30 14:19 knexfile.ts
drwxr-xr-x 1051 germanescobar  staff  33632 Apr  9 11:08 node_modules
-rw-r--r--   1 germanescobar  staff    100 Mar 30 14:19 nodemon.json
-rw-r--r--   1 germanescobar  staff  651125 Apr  9 11:29 package-lock.json
-rw-r--r--   1 germanescobar  staff   2965 Apr  9 11:08 package.json
drwxr-xr-x  14 germanescobar  staff    448 Mar 31 12:03 server
drwxr-xr-x   4 germanescobar  staff    128 Mar 30 14:19 tests
-rw-r--r--   1 germanescobar  staff    607 Mar 31 13:52 tsconfig.json
-rw-r--r--   1 germanescobar  staff   2020 Mar 31 14:03 webpack.config.js
```

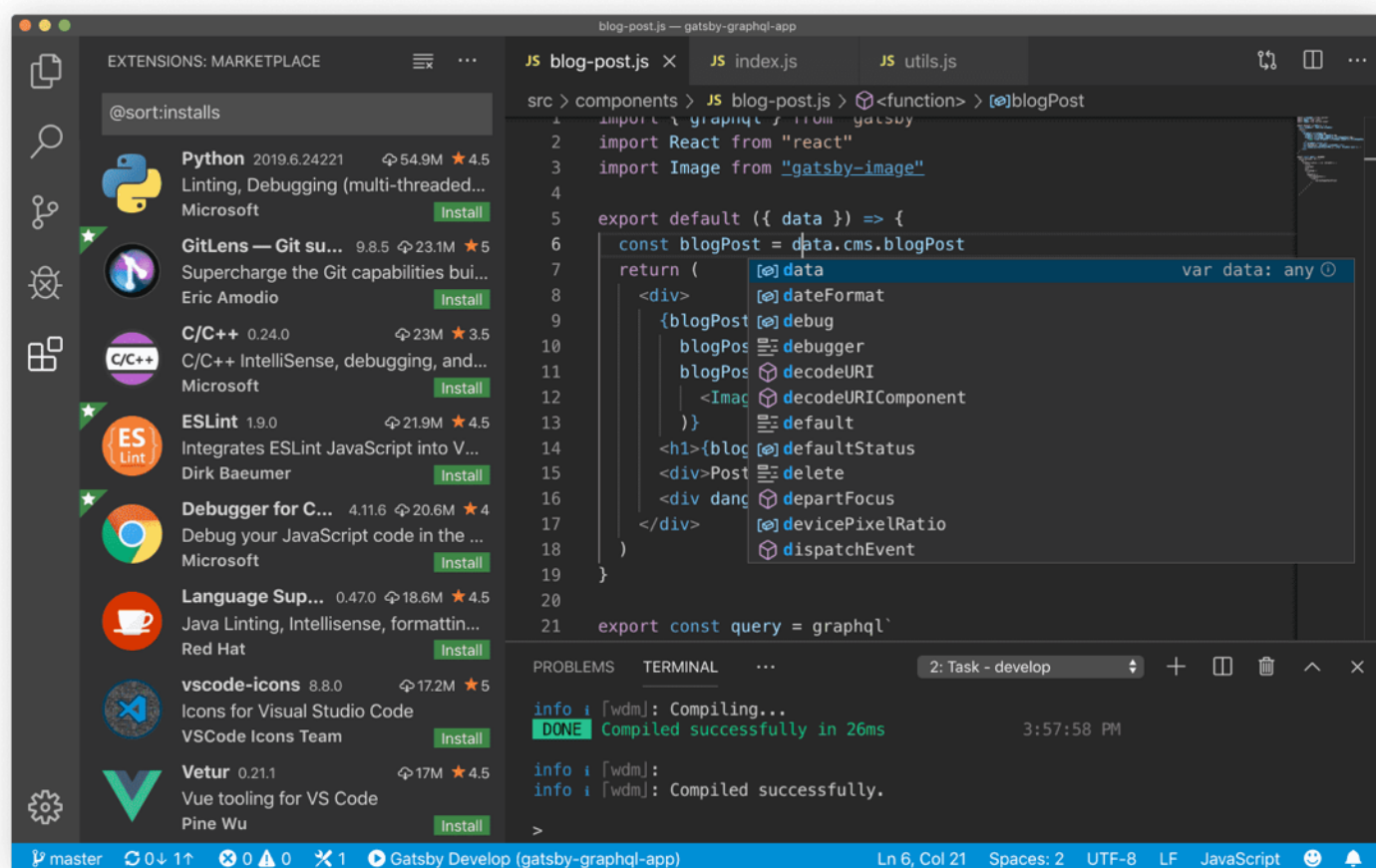
Ejemplo de la línea de comandos.

Un editor de texto

Para crear tus aplicaciones Web vas a necesitar un editor de texto en donde escribir el código. Los editores de texto más básicos son Bloc de Notas en Windows y TextEdit en Mac. Pero el problema de estos editores de texto es que no están diseñados para los programadores.

Un editor de texto para programadores debe tener al menos las siguientes dos características básicas: debe resaltar el código con colores (recuerda que el código es texto plano sin formato) y debe auto completar el código a medida que lo escribas.

Aunque existen muchos editores de texto en el mercado, el más popular ahora entre los programadores es [VSCode](https://code.visualstudio.com/), que se puede descargar y usar gratuitamente.



El editor de texto. Imagen tomada de <https://code.visualstudio.com/>.

Un sistema de control de versiones

Un sistema de control de versiones tiene dos objetivos principales:

- Guardar el historial de cambios sobre el código de un proyecto.

- Permitir a varias personas contribuir simultáneamente sobre un mismo proyecto.

El sistema de control de versiones más popular actualmente es Git. Git fue diseñado y desarrollado inicialmente por Linus Torvalds (creador del sistema operativo Linux) en el 2005 cuando BitKeeper, el sistema de control de versiones que utilizaban para el desarrollo de Linux, cambiara su licencia y no permitiera su uso libre.

Si no lo has hecho, ve y crea una cuenta en [Github](#). [Github](#) es la red social de los programadores. Ahí puedes ver el código de varios proyectos Open Source, colaborar en esos proyectos o publicar tus propios proyectos.

El protocolo HTTP

Cada vez que entras a algún sitio Web (p.e. Google o Facebook), tu navegador está intercambiando mensajes según unas reglas establecidas por HTTP (Hyper Text Transfer Protocol).

Muchas personas prefieren aprender primero un framework Web antes de HTTP, pero nuestra experiencia nos dice que aprender HTTP antes es mucho más valioso porque te va a permitir adaptarte a cualquier framework Web. Además, aprender sobre el protocolo HTTP te va a preparar para entender Web API's más adelante, que es un mecanismo para conectar aplicaciones de forma automática.

Un framework Web (backend)

Para crear aplicaciones Web en el servidor (backed) es recomendable usar un framework Web. Para cada lenguaje vas a encontrar varias opciones, aunque siempre hay uno que es más popular que los demás. Como decíamos, el framework Web más popular de Ruby se llama [Ruby on Rails](#). El de Python se llama [Django](#), el de Node.js (JavaScript) se llama [Express.js](#). Y la verdad es que todos estos frameworks son parecidos porque siguen un patrón llamado MVC (Modelo Vista Controlador). PHP también tiene un framework MVC llamado [Laravel](#).

¿Cuál aprender? La respuesta es **cualquiera** siempre y cuando sea un framework MVC. Si empezaste aprendiendo Python intenta [Django](#). Si empezaste con JavaScript sigue con [Express.js](#). Y así.

Si no has empezado con ningún lenguaje en particular nuestra recomendación es que escojas algo que tus amigo(a)s, o en tu empresa, usen (asegúrate de que les puedas preguntar todo lo que necesites sin que se enojen).

Si no tienes amigo(a)s Desarrolladores Web y tu empresa no es de tecnología, nuestra recomendación es que empieces con Ruby y [Ruby on Rails](#) o Node.js (JavaScript) y [Express.js](#).

Hace un tiempo [Ruby on Rails](#) estaba tomando mucha fuerza pero hoy Node.js (JavaScript) se ha convertido en la opción dominante, especialmente para encontrar un empleo.

Bases de Datos

La información que genera una aplicación Web se almacena en una base de datos. Las bases de datos se pueden categorizar en dos: relacionales y no relacionales. La forma de interactuar con una base de datos relacional es a través de SQL (Structured Query Language). A las bases de datos no relaciones también les llaman NoSQL y en los últimos años han tomado bastante fuerza.

Ejemplos de bases de datos relacionales incluyen [MySQL](#), [PostgreSQL](#), [Oracle](#) y [SQLServer](#) (Microsoft). Ejemplos de bases de datos no relacionales incluyen [MongoDB](#), [CouchDB](#), [Cassandra](#) y [Redis](#).

Nuestra recomendación es que no te preocupes por la base de datos en este momento. Cuando aprendas un framework Web seguramente aprenderás sobre bases de datos. Si aprendes [Ruby on Rails](#) seguramente vas a escoger [PostgreSQL](#). Si aprendes [Express.js](#) seguramente vas a escoger [MongoDB](#), y así.

jQuery

[jQuery](#) es una librería escrita en JavaScript que soluciona las diferencias entre los navegadores al manipular HTML, escuchar eventos (clicks, movimientos del mouse, teclas, etc.) y hacer llamados AJAX a servidores remotos.

Aunque se dice que en un futuro no va a ser necesario [jQuery](#) (porque las diferencias entre los navegadores son cada vez menor), la realidad es que hoy sigue siendo una tecnología fundamental en el desarrollo Web. No solo eso, si quieres especializarte en front-end deberías convertirte en un maestro [jQuery](#).

AJAX (Asynchronous JavaScript and XML) es una tecnología que permite intercambiar información con un servidor remoto sin necesidad de refrescar la página. Hoy también es posible hacer llamados AJAX con la misma simplicidad de [jQuery](#) utilizando el [Fetch API](#).

Un framework Web (frontend)

Antes un desarrollador Web escribía tanto el frontend como el backend. Sin embargo, a medida que el desarrollo frontend empezó a evolucionar se separaron los roles.

Crear una aplicación compleja en el navegador utilizando JavaScript puro o [jQuery](#) no es fácil. Es por eso que han surgido frameworks como [React](#), [Vue](#) y [AngularJS](#), por nombrar algunos, que permiten escribir código más mantenible en el tiempo. La desventaja es que la curva de aprendizaje es mucho más empinada. Trabajar con estos frameworks requiere un cambio de pensamiento en comparación a [jQuery](#).

Antes [AngularJS](#) era el framework más popular pero ha sido desplazado en los últimos años por [React](#). [Vue](#) inició como una alternativa a [React](#) más simple pero no ha logrado la misma acogida, aunque es muy popular en la comunidad de PHP y en China (el país).

Hoy también es posible crear aplicaciones móviles utilizando una librería utilizando HTML, CSS y JavaScript a través de una librería llamada React Native que utiliza los mismos conceptos de [React](#) y después compila la aplicación a iOS y Android. Es otra ventaja de aprender [React](#).

Despliegue de aplicaciones

"Desplegar" es un término que utilizamos los programadores cuando nos referimos a publicar una aplicación Web (por primera vez o con cambios que hemos hecho) en Internet.

La verdad es que este es otro tema del que no te deberías preocupar en este momento. Al fin y al cabo, primero debes aprender a construir las aplicaciones, pero igual lo mencionamos acá para que lo tengas de referencia.

El tema de servidores ha evolucionado muy rápido en los últimos años. Antes eran muy populares los "hosting compartidos", en donde se publicaban varias aplicaciones sobre un mismo servidor y todas compartían el sistema operativo y los mismos recursos de la máquina. Era la única opción en muchos casos porque alquilar un servidor exclusivo era muy costoso.

Hoy, gracias a la virtualización, es posible simular un servidor exclusivo sin necesidad de que físicamente sea exclusivo. Además, hoy es posible alquilar servidores virtualizados por horas y el proceso para hacerlo toma minutos. A esto se le llama IaaS (Infrastructure as a Service) y es muy popular. Antes solicitar un servidor exclusivo tomaba horas y el cobro era mensual.

La compañía líder en IaaS es Amazon con su servicio [AWS](#) (Amazon Web Services). [AWS](#) ofrece varios servicios, pero el más importante es EC2 (Elastic Cloud Compute). Con EC2 puedes solicitar uno o más servidores y pagar por hora según las necesidades de memoria y procesamiento que necesites. Toda la infraestructura de [Netflix](#), [Pinterest](#) y [AirBnB](#), entre otras compañías

reconocidas, están sobre [AWS](#). Existen varias alternativas a EC2 como [Digital Ocean](#), [Microsoft Azure](#) y [Google Compute Engine](#), entre otros.

Pero desplegar una aplicación Web sobre cualquier IaaS es engorroso. Necesitas solicitar un servidor, seleccionar las características y el sistema operativo, esperar a que inicie e ingresar remotamente para instalar la aplicación y sus dependencias (lenguaje de programación, base de datos, etc.). Es por eso que ahora existen los PaaS (Platform as a Service).

Los PaaS son ideales para principiantes porque no necesitas ingresar a la máquina a instalar tu aplicación. El PaaS se encarga de eso. Ejemplos de PaaS incluyen [Heroku](#) y [Google App Engine](#), entre otros. La forma de desplegar tu aplicación varía entre cada PaaS, pero en la mayoría puedes hacerlo directamente con Git ejecutando un simple comando. La desventaja de los PaaS es que son más costosos y no tienes control sobre la infraestructura, aunque muchas veces eso es precisamente lo que estás buscando.

¿Cómo consigo un empleo como desarrollador Web?

Es cierto que existe una oportunidad enorme para los Desarrolladores Web, y en general para los programadores. La demanda sigue aumentando al igual que los salarios. Muchos trabajan remotamente, con gran flexibilidad horaria. El estilo de vida de muchos Desarrolladores Web es bastante envidiable para los estándares de la actualidad. Pero eso no quiere decir que conseguir un empleo así sea fácil. Veamos qué necesitas realmente para conseguir un empleo como Desarrollador Web.

Lo primero que debes saber es que los empleadores dividen a los Desarrolladores Web en dos: Junior y Senior. ¿Qué diferencia a uno del otro? Definitivamente no es un título ni un certificado. Para que te des una idea, un Desarrollador Web Junior necesita invertir mínimo entre 800 y 1,200 horas de aprendizaje para conseguir un empleo. Por otro lado, un Desarrollador Web Senior ha programado durante al menos 5 años tiempo completo (~10,000 horas) y es capaz de liderar un equipo de desarrollo.

En Estados Unidos un Desarrollador Web Junior puede empezar ganando entre \$4,000 a \$6,000 dólares al mes, pero en Latinoamérica ese rango es mucho más bajo: entre \$800 y \$1,500 dólares. Sería ideal encontrar empleo remoto como Desarrollador Web Junior en Estados Unidos pero no es fácil.

Para los Desarrolladores Web Senior el panorama es muy distinto. En Estados Unidos un Desarrollador Senior puede ganar entre \$6,000 y \$12,000 dólares al mes. En Latinoamérica el rango es mucho más bajo pero igual interesante:

entre \$2,000 y \$6,000 dólares. La ventaja es que un Desarrollador Senior, con un nivel de Inglés intermedio, puede encontrar un empleo remoto en Estados Unidos con relativa facilidad.

Como te puedes dar cuenta, la oportunidad es especialmente interesante para los Desarrolladores Senior, más que para los Junior. Si conoces a alguien que trabaje remotamente con un buen salario es muy probable que sea un programador Senior.

Sin embargo, encontrar un empleo como Desarrollador Web Junior es relativamente fácil si tienes un verdadero interés por la programación y te gusta enfrentarte a nuevos retos. Nuestro consejo es que pienses muy bien en tus razones y motivaciones, especialmente si estás pensando en cambiar de carrera.

Si crees que tu principal motivación es la programación y quieres convertirte en Desarrollador Web, el siguiente paso para encontrar un empleo es: **olvida el empleo y concéntrate en maximizar tu aprendizaje**. Algunos consejos son:

- Busca nuevos retos para trabajar (así no te paguen). Construye versiones simplificadas de tus aplicaciones Web favoritas o intenta resolver problemas en [Leetcode](https://leetcode.com/) por ejemplo.
- Busca inspiración en videos, blogs, etc.
- Asiste a las reuniones de programadores (Meetups) y a hackathons. También a Startup Weekends.
- Aprende sobre nuevos lenguajes y tecnologías.
- Fortalece los fundamentos de la programación: algoritmos, estructuras de datos, sistemas operativos, compiladores, etc.

- Escribe y comparte lo que estás aprendiendo.
- Lee libros de programación.
- Aplica a empleos. No pienses que vas a conseguir inmediatamente pero te va a dar una idea de lo que te falta por aprender.

Para concluir vale la pena reiterar que existe una gran oportunidad para los que aprendan a programar, pero nada es fácil en la vida. Vas a necesitar tu mayor concentración y dedicación para llegar a un nivel en el que puedas ser empleable, y mucho más para conseguir el empleo de tus sueños.

¿Dónde puedo aprender todo esto?

En [Make it Real](https://makeitreal.camp/) ofrecemos programas enfocados en JavaScript Full Stack. Puedes encontrar toda la información en nuestra página principal <https://makeitreal.camp/>.

Sin embargo, es importante aclarar que toda la información para aprender estos temas está disponible en Internet. Lo que ofrecemos en nuestros programas es una guía con mentores que te ayudarán a acelerar tu aprendizaje y acceso a una comunidad que está persiguiendo esos mismos intereses.

Conclusión

Espero que este libro te haya dado el panorama general de cómo convertirte en desarrollador Web. Como decía antes: la oportunidad es enorme pero eso no significa que sea fácil.

Sin embargo, sabemos que con la actitud y la guía adecuada es posible, así como lo han logrado cientos de nuestros alumnos que trabajan como desarrolladores Web y móvil utilizando estas tecnologías cada día.

Para terminar, te envío un gran saludo y espero algún día verte por [Make it Real](#), estamos ahí para ayudarte en este camino.

Germán Escobar
makeitreal.camp