Full-Stack Developer

Hello world

##### **BLOQUE 1 - ENCUENTRO 1**

# Introducción a Desarrollo Web Full Stack - Tecnología web - Servidor web

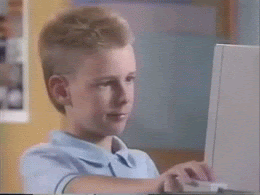
**Introducción a la tecnología web**

***“Cualquier persona sin importar sus estudios previos puede aprender a programar, quizás lo único que se requiere es la paciencia y tener un problema real que resolver”. —Mario Alberto Chávez, Rubyist and software engineer.***[***Aprender a Programar o desarrollar.***](https://medium.com/@mario_chavez/aprender-a-programar-o-desarrollar-ff16ff5d4906)

**Los primeros pasos como Full Stack Developer**

¡Bienvenido/a a tu primera bitácora! Aquí encontrarás disponibles los conceptos clave y las herramientas que serán necesarias para prepararte para cada encuentro con tu equipo docente y compañeros/as. Además, motivaremos tu acceso a los canales de comunidad: te propondremos interactuar con otros/as Developers para disipar dudas, **hacer** consultas y, por qué no, conocer y recomendar cosas interesantes sobre los temas que trabajaremos.

En este primer bloque de aprendizaje (que consta de 16 bitácoras para 16 encuentros), aprenderás a estructurar un sitio web mediante HTML y a aplicar los estilos que desees con CSS. En otras palabras, en este bloque trabajaremos en lenguajes de programación para el Front-End, i.e. aquello que observan los/as usuarios/as cuando ingresan en un sitio web. Además, aprenderás buenas prácticas de programación y practicarás con un proyecto real: Podcast Channel.



En particular, abordaremos los siguientes temas:

* **Desarrollo web (conceptos fundamentales para entender qué hacemos cuando programamos)** : Entorno / Internet / Web / Cliente - Servidor / Front-end / Sitios web estáticos y dinámicos / Aplicaciones web / Desarrollo / Programación / Lenguajes de la web / Programación lógica / Maquetado / Estilado / Herramientas de desarrollo.
* **HTML (la arquitectura de nuestro sitio)** : Introducción al lenguaje / Sintaxis / Estructura del documento / Etiquetas semánticas / Etiquetas estructurales / Atributos / Etiquetas funcionales / Rutas relativas y absolutas / Vinculación de recursos / Buenas prácticas.
* **CSS (el estilo estético de nuestro sitio):**Introducción al lenguaje / Sintaxis / Integración con HTML / Selectores por etiqueta / Selectores a través de atributos / Propiedades y valores / Sistema de cascada / Estilos por defecto / Posicionamiento de elementos / Modelo de cajas / Relatividad / Debugging en el explorador / Responsive web design / Unidades de medida relativas / Flexbox / Media-queries / Pseudo clases / Pseudo elementos / Transformaciones / Animación / Recursos externos (Google Fonts, FontAwesome) / Framework y librerías (Bootstrap) / Pre-procesadores (SASS).

**¿Qué es programar?**

Si nos apuramos, podríamos decir que programar es pasar **de ser usuarios a creadores.** Cuando buscamos definiciones acerca de qué es la programación, nos encontramos con respuestas tales como “dar instrucciones al ordenador” o bien “crear software usando un lenguaje de programación”. Según [Wikipedia,](https://es.wikipedia.org/wiki/Programaci%C3%B3n#cite_note-2) es el proceso por el cual una persona desarrolla un programa valiéndose de una herramienta que le permita escribir el código (nosotros usaremos HTML, CSS y Javascript) y de otra que sea capaz de “traducirlo” a lo que se conoce como lenguaje de máquina, que puede comprender el ordenador.

Agustín Quetto lo explica de una manera más simple:

*?”Caminamos hasta que las piernas nos duelan, dormimos hasta que no conciliamos más el sueño y buscamos dinero en la billetera hasta encontrar la cantidad suficiente. Todo posee un cambio de estado, ¿no crees? Si las piernas no duelen, caminamos. Si estamos cansados, dormimos hasta no estarlo y buscamos dinero en la billetera hasta dar con la cantidad necesaria (ojalá siempre fuese así, ¿no?). Programar no es más que lo que hacemos a diario: un conjunto de instrucciones (...) de manera “digital”. No hay diferencia. Programar es como hablar con una computadora: decirle qué hacer, dónde y cómo. Para esto podemos poseer valores que cambian, para saber si aún tenemos sueño o no, ciclos para repetir las cosas como ir al trabajo todos los días, preguntar el valor de algo para saber si estamos o no en situación de hacer cierta acción o tomar una decisión.”*[*(*](https://medium.com/@agustinquetto/qu%C3%A9-es-programar-vivimos-programando-6fcd523ad882) [*“¿Qué es programar? ¿Vivimos programando?”*](https://medium.com/@agustinquetto/qu%C3%A9-es-programar-vivimos-programando-6fcd523ad882) [*)*](https://medium.com/@agustinquetto/qu%C3%A9-es-programar-vivimos-programando-6fcd523ad882)

Desarrollar un software supone un conocimiento en programación, pero además, comprender las implicancias de ciertas decisiones a la hora de implementar una solución. Por eso, los/las desarrolladores/as formamos parte -directa o indirectamente- de todo **ciclo de desarrollo de software: la planificación, el análisis, el diseño de la solución, el testing, la integración y el mantenimiento de un producto.**

Diagram

Description automatically generated

**Las 5 claves que hacen que los/las desarrolladores/as seamos todo terreno**

El desarrollo web full stack **,**de acuerdo con la Encuesta de [Stack Overflow 2019](https://insights.stackoverflow.com/survey/2019) [,](https://insights.stackoverflow.com/survey/2019) actualmente la ocupación de Full Stack Developers es la más demandada. Iniciarnos en este camino de un [Full Stack Developer](https://medium.com/@jdevmanzo/como-ser-desarrollador-web-full-stack-sin-morir-en-el-intento-42c0f8e605c2) significa que vamos a saber cómo agregar valor en las interfaces visibles para los/as usuarios/as (Front-end) como en la lógica que opera detrás (Back-end).

Te contamos cuáles serán las habilidades necesarias para el rol que desempeñes como Full Stack Developer. Como cualquier habilidad, no se adquieren de un día para el otro, sino que se ejercitan en el tiempo y nuestra propuesta es que las trabajes a lo largo de la carrera.

1. **Resolvemos problemas:** convertimos los requerimientos -es decir, las necesidades de la vida real con las que estamos trabajando- en soluciones. Sabemos qué problema resuelve el software y en cada parte del código orientamos nuestro desarrollo específico a la solución del problema general.
2. **Ejercitamos la creatividad:** si bien es cierto que existen buenas prácticas y patrones para resolver problemas clásicos, no hay una sola forma de llegar al mismo resultado. Para satisfacer las necesidades del software que desarrollamos es imprescindible que pongamos nuestra creatividad a disposición de la solución. Una buena forma de hacerlo es buscar diferentes caminos y analizar cuál es el que resolverá nuestro requerimiento con mayor eficiencia. A veces, también es necesario ir un poco más allá  de las soluciones que están al alcance y ampliar nuestra estructura de pensamiento, siendo profundamente críticos/as con el camino conocido y poniendo en práctica nuestra capacidad para desafiar nuestros propios horizontes.
3. **Ejercitamos el pensamiento abstracto:**los lenguajes de programación nos proveen símbolos y estructuras para poder representar información y la forma en que se va a procesar dentro de nuestro programa. Estos símbolos y estructuras son parte de un marco de trabajo que aceptamos para poder implementar nuestras soluciones. Este marco es una abstracción.
4. **Aplicamos el pensamiento lógico:** las bases del pensamiento crítico con las que fuimos educados directa o indirectamente en las distintas etapas de la educación formal nos dotaron de la lógica proposicional para poder argumentar y sacar conclusiones a partir de los datos que tenemos acerca de un problema dado. Cuando realizamos operaciones de comparación para procesar información que recibe nuestra aplicación, estamos ni más ni menos que aplicando las tablas de verdad que nos provee esta disciplina. Si no sabes de qué hablamos, ¡no te preocupes! Lo vas a aprender programando :)
5. **Nos actualizamos:** no debe existir una frase más trillada que “la tecnología avanza a pasos agigantados”. Pero ¡no por eso deja de ser cierta! Los/as Full Stack Developers contamos con una ventaja en ese sentido. Vivimos trabajando con las últimas tecnologías, y si bien es nuestra responsabilidad intentar estar al tanto de lo que pasa allá afuera, estar activos profesionalmente nos da las herramientas para conocer el contexto.

Ahora bien, antes de comenzar con los temas de esta carrera, queremos darte tres reglas que cumpliremos a lo largo de la carrera, y que son fundamentales para maximizar tu aprendizaje:

1. **Vale decir “no sé”.** Los mejores Full Stack Developers saben reconocer cuando no saben qué significa algo, qué implica, o cómo resolver situaciones que se les presentan. Esta habilidad es fundamental, ya que es el primer disparador para ir a buscar ayuda (ya sea pedirle a alguien que sepa más, que lo haya vivido antes, que esté dispuesto a debatir o buscar en internet).
2. **Si sabes lo que tienes que ‘codear’ pero no sabes cómo hacerlo, entonces no tienes un problema: tienes un desafío.**No saber cómo programar algo puede ser una situación técnica que se puede resolver preguntando a otros o buscando en internet. En múltiples situaciones, verás que la clave no es saber cómo escribir las líneas de código, sino saber o entender el código que hay que escribir. Eso significará que, más allá de tu capacidad técnica, sabes a dónde debes llegar y, por ende, estarás mejor posicionado para buscar las respuestas o caminos posibles.
3. **A programar se aprende, valga la redundancia, programando.** Deberás sentarte a programar para aprender a hacerlo. Te acompañaremos en ese trayecto y trataremos de que sea lo mejor posible, pero no te garantizamos que todo será disfrute. Habrá momentos difíciles, otros en los que no tendrás ganas de volver a programar, e incluso algunos en los que te plantees “¿quién me mandó a hacer esto?”. Pero esperamos acompañarte a sobreponerte y continuar intentándolo, valga la repetición, programando.

**Sitios web: del dicho al hecho hay servidores, clientes, dispositivos, y navegadores**

Ya te contamos con qué desafíos te encontrarás en el camino hacia convertirte en Full Stack Developer de aplicaciones que generan nuevas oportunidades de vida para otros/as usuarios/as. ¡Adentrémonos en el mundo que hay detrás así lo conocemos mejor!

[Ethan Marcotte,](https://alistapart.com/article/responsive-web-design/) diseñador web, y creador del libro Responsive Web Design, hace referencia a la arquitectura:

*“Los cimientos de un edificio definen su plano, que define su estructura, que da forma a la fachada. Cada fase del proceso arquitectónico es más inmutable, más invariable que la anterior. Las decisiones creativas modelan, literalmente, un espacio físico, definiendo la forma en que la gente se moverá a través del mismo por décadas o incluso siglos”.*

Como Full Stack Developer es importante que conozcas la estructura de internet y cómo funciona para poder crear sitios web. Sin darnos cuenta, todos los días estamos en contacto con esta estructura cuando usamos nuestros dispositivos como el celular o la computadora para visitar páginas web de interés o subimos fotos a redes sociales como Instagram, Facebook o Twitter.

Las primeras preguntas que son fundamental hacernos como Full Stack Developers son:

* ¿Cómo funciona internet?
* ¿Cómo es que podemos visualizar un sitio web?
* ¿De dónde sale la información? ¿Qué sucede detrás de las aplicaciones?

Internet? es un conjunto descentralizado de redes de comunicación interconectadas que utilizan ciertos protocolos comunes (TCP/IP) para poder “entenderse”. Esto garantiza que las redes físicas heterogéneas que la componen sean parte de una red lógica única de alcance mundial.

Si bien abordaremos el concepto de internet y su historia en la bitácora 2, es importante que sepas en este punto que el World Wide Web (WWW) no es lo mismo que internet, sino que se trata de uno de los servicios que más éxito ha tenido en internet. La WWW es un conjunto de protocolos que permite, de forma sencilla, la consulta remota de archivos de hipertexto.

Nosotros/as los/as usuarios/as, cuando accedemos a los sitios web realizamos solicitudes como “clientes” a través de dispositivos (computadoras, teléfonos, tablets).

Los servidores son quienes reciben estas solicitudes y nos dan una respuesta en diferentes formatos según lo que pidamos (imágenes publicadas, nuestro correo electrónico o la página web que queremos ver).

Por lo tanto, **un** **servidor** **se encuentra disponible para otros dispositivos y será quien suministre la información requerida por los clientes.**

A close up of electronics

Description automatically generated

Todos nuestros dispositivos se conectan a través de internet con otras computadoras conformando una red que aloja sitios web y funcionan como servidores para que todo el mundo pueda ver lo que solicita.

Existen varios [tipos de servidores.](https://www.lifeder.com/tipos-de-servidores/) ¡No te preocupes por aprenderte todos los términos y definiciones, de a poco nos vamos a ir familiarizando! Pero **si tu intención es desarrollar un sitio web debes buscar este alojamiento y el primer al lugar que tienes que acudir es un servidor web.**

**¿Qué es un servidor web?**

Cada vez que un cliente acceda desde el navegador a nuestro sitio web, el servidor web transferirá todos los archivos necesarios para atender la solicitud y resolverla. Nos conectamos a otras computadoras y no hacemos más que descargarnos una serie de archivos que desde otra computadora conectada a la red actúa de servidor web.

En otras palabras, **un**[**servidor web**](https://hostingsaurio.com/que-es-un-servidor-web/) **almacena todos los archivos que componen una aplicación web y visualiza nuestro producto a los clientes a través del navegador.**

[Emmanuel Orozco](https://medium.com/laboratoria-how-to/entendiendo-como-funciona-el-internet-parte-http-6c8c5e078303) nos explica: *"sin embargo, tiene que existir algún tipo de estándar para que mi navegador, tu navegador, todos los navegadores y servidores del mundo se comprendan. Ese estándar, se llama protocolo.”*Mediante el [**protocolo HTTP**](https://www.definicionabc.com/tecnologia/http.php) **en el puerto 80 el cliente recibe los archivos que allí están alojados.**

Diagram

Description automatically generated

Podemos simplificar los pasos de cliente/servidor de la siguiente manera:

1. Inicio de petición por parte del cliente (ingresa [www.google.com).](http://www.google.com/)
2. El servidor web recibe la petición y la resuelve.
3. El cliente recibe la respuesta.
4. Cierre de transacción (visualización de la página web).

Para tener tu propio **servidor web** podrás basarte en esta estructura clásica de cliente-servidor, y en el momento en que comiences a desarrollar **aplicaciones web** verás que será necesario replicar este escenario.

**Qué debemos saber de las aplicaciones web**

Para comenzar a interactuar en este mundo de la web debemos conocer cómo funciona a través de internet y también qué lenguajes se utilizan.

Chart, funnel chart

Description automatically generated

Hay tres lenguajes básicos que van a tener un rol específico en una aplicación web:

* **HTML** nos permite estructurar el contenido que vamos a brindarle al usuario.
* Con **CSS** vamos a trabajar con la presentación visual del contenido.
* **JavaScript** nos da la posibilidad de interactuar con las acciones del usuario y actualizar el contenido sin necesidad de recargar el sitio.

Los documentos que componen una aplicación web son archivos de texto plano con una extensión específica que los hace reconocibles para los navegadores en cuanto al rol que cumplen dentro del sitio.

¡Ya te contaremos más en el encuentro que viene y en las siguientes bitácoras!

##### **BLOQUE 1 - ENCUENTRO 2**

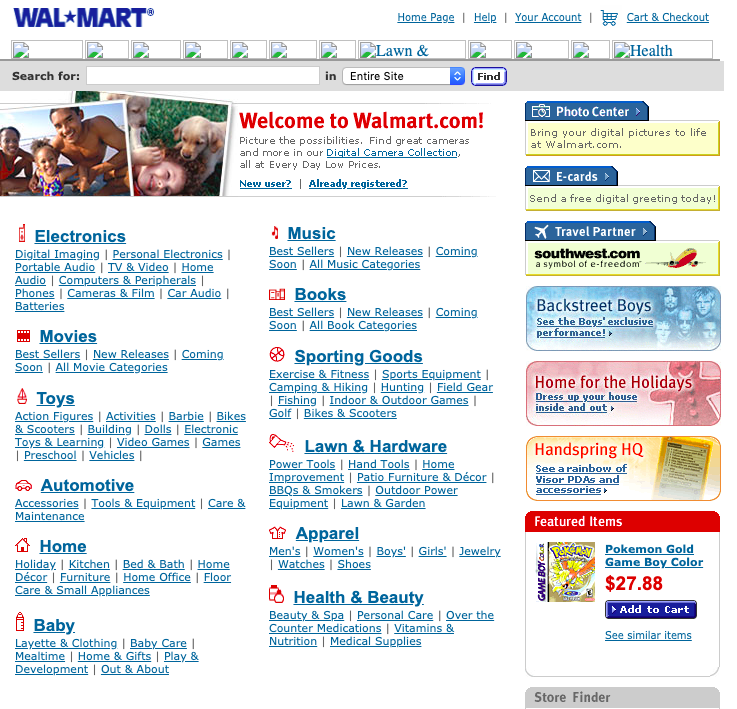
# HTML y Arquitectura Web

## Del big bang de la web hasta la actualidad

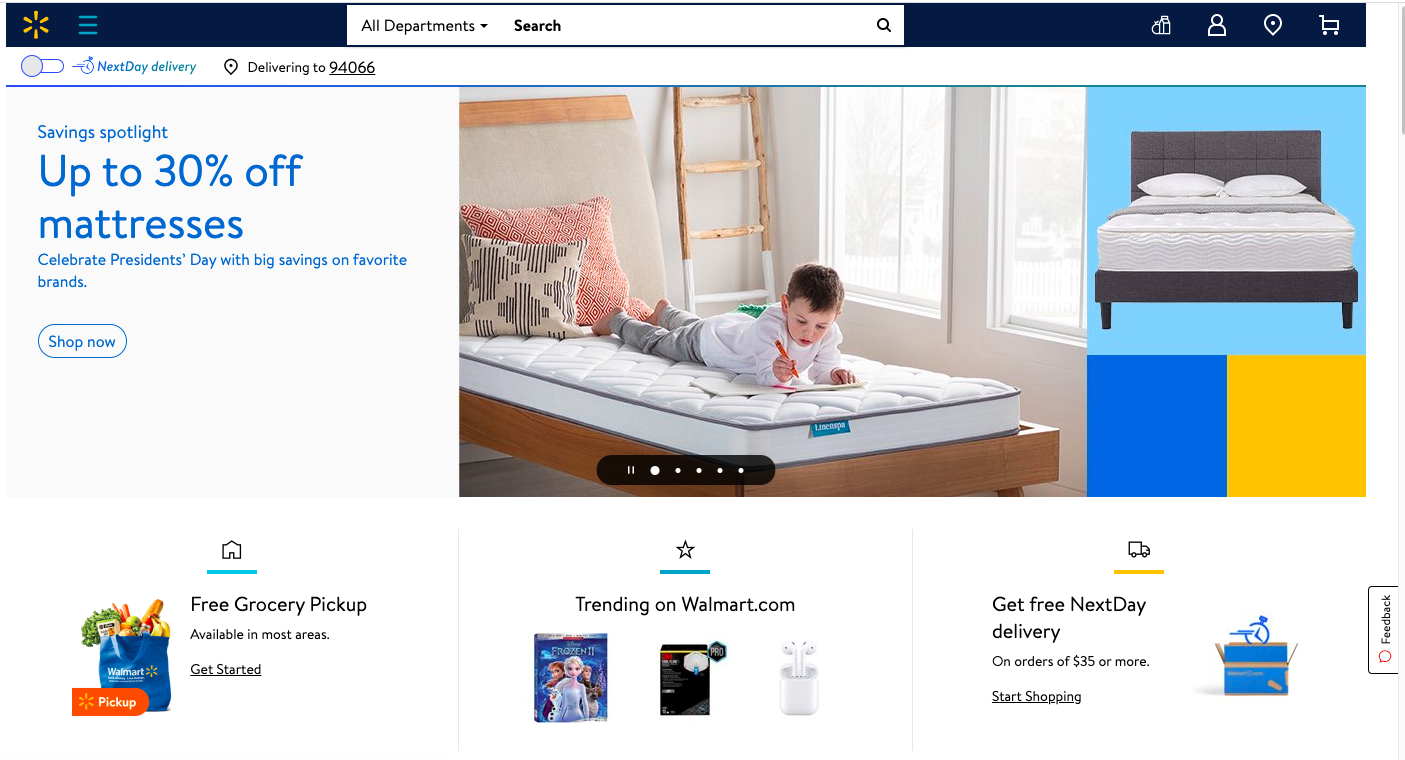
**“La web es más una creación social que técnica. La diseñé para un efecto social —para ayudar a las personas que trabajan juntas— y no como un juguete técnico. El objetivo último de la web es apoyar y mejorar nuestra existencia en la telaraña mundial.” —Tim Berners-Lee.**

La web no siempre fue tal como la conocemos hoy en día: grandes modificaciones se han producido desde sus comienzos. Por ejemplo, observa cómo cambió el sitio web de Wal-Mart en los últimos 20 años:

www.walmart.com en 2000:



www.walmart.com en 2020:



Puedes buscar más ejemplos en esta gran biblioteca pública de archivos web actuales e históricos: [Wayback Machine.](https://archive.org/web/)

**Dato:**Seis mujeres —Kathleen McNulty Mauchly Antonelli, Frances Bilas Spence, Ruth Lichterman Teitelbaum, Marlyn Wescoff Meltzer, Frances Snyder Holbertson and Betty Jennings Bartik— permitieron que internet se desarrolle como se desarrolló, ya que programaron la primera computadora 100% eléctrica en 1945 para un proyecto de la Armada de EE.UU. Puedes leer [aquí](https://www.shethepeople.tv/news/5-women-who-made-the-internet-possible) otras contribuciones de mujeres en la computación que produjeron grandes transformaciones.

Los orígenes de internet se remontan a la Guerra Fría, cuando el Departamento de Defensa de EEUU creó [una Agencia de Investigación de Proyectos de Defensa](https://es.wikipedia.org/wiki/Agencia_de_Proyectos_de_Investigaci%C3%B3n_Avanzados_de_Defensa) (ARPA , por sus siglas en inglés) que nucleaba a centros de investigación de distintas universidades.

[John Lickinder](https://es.wikipedia.org/wiki/Joseph_Carl_Robnett_Licklider) comenzó a mediados de los ‘60 a pensar en una red que permitiera transmitir datos entre las universidades y poder centralizar las investigaciones. La llamó “Intergalactic Computer Network” pero no pudo concretarla. [Robert Taylor](https://es.wikipedia.org/wiki/Robert_Taylor_(inform%C3%A1tico)) tomó esta idea y formó un equipo junto a [Lawrence Roberts,](https://es.wikipedia.org/wiki/Lawrence_G._Roberts) [Leonard Kleinrock](https://es.wikipedia.org/wiki/Leonard_Kleinrock) y [Donald Davies](https://en.wikipedia.org/wiki/Donald_Davies) para llevarla a cabo. El nombre de la red sería [ARPANET.](https://es.wikipedia.org/wiki/ARPANET)

Para desarrollar ARPANET, tuvieron en cuenta una tecnología existente (la red de telefonía) y desarrollaron una forma de enviar paquetes a través de la misma, basada en la teoría de colas proveniente de la matemática y desarrollada por [Agner Erlang.](https://es.wikipedia.org/wiki/Agner_Krarup_Erlanghttps:/es.wikipedia.org/wiki/Agner_Krarup_Erlang)Incluyeron además el concepto de multiplexación para poder hacer transmisiones simultáneas entre diferentes equipos.

Los equipos que desarrollaron se llamaron “Interface Message Processor”, y en **octubre de 1969 realizan la primera conexión de manera exitosa entre UCLA y Stanford.**

El uso masivo de la red hizo que gobiernos y universidades del mundo se sumaran y se crearan protocolos. Como con cualquier lenguaje o comunicación entre personas, entes u organizaciones, es necesario establecer algunas normas de convivencia para facilitar la comprensión. ¡Era necesario estandarizar la red!

En 1981 nació el [**protocolo TCP/IP,**](https://es.wikipedia.org/wiki/Modelo_TCP/IP) **que es la tecnología que hoy en día se utiliza para interconectar a las computadoras del mundo, a través de la red de INTERNET.**De esta manera se estandarizó la comunicación punto a punto para que todas las computadoras pudieran “hablar” entre sí en el mismo lenguaje.

Pasamos de una comunicación así:

A picture containing table, room, kite, bed

Description automatically generated

A una comunicación así:

A picture containing text, light, map, traffic

Description automatically generated

Durante los ‘80, la red de internet se estandarizó y se empezó a utilizar en la mayoría de los centros de conocimiento, entre ellos, el [CERN](https://home.cern/)(Comisión Europea de Energía Atómica), que tiene el nodo de internet más grande de Europa.

**Dato:** Desde dentro del CERN, el físico [Tim Berners-Lee](https://es.wikipedia.org/wiki/Tim_Berners-Lee) comenzó a pensar que era necesario que se pudiera acceder a las investigaciones de las distintas sedes desde cualquier punto de la red. En 1989 presentó ante el CERN [Cómo funciona internet,](https://www.acamica.com/clases/10793/que-es-internet-y-como-funciona) el proyecto de [World Wide Web.](https://es.wikipedia.org/wiki/World_Wide_Web)  Creó para esto un protocolo, un lenguaje, una arquitectura de comunicación y un intérprete de documentos HTML (Navegador).

En este [artículo](https://okdiario.com/ciencia/grandes-frases-tim-berners-lee-3536759) del diario digital “OKDIARIO” se destaca sobre Tim Berners-Lee: “Tanto él como su grupo de trabajo crearon las siglas conocidas como lenguaje HTML, es decir, las etiquetas de hipertexto, así como el sistema de localización URL y el protocolo HTTP. Así que, de no ser por él y por sus proyectos Memex y Xanadú, tal vez ahora no estarías leyendo este texto.”

A screenshot of a cell phone

Description automatically generated

La **arquitectura web** que crea Tim Berners-Lee es la de cliente-servidor (que vimos en la bitácora anterior).

## De los sitios web a las aplicaciones web

En 1991 se lanzó la **World Wide Web** de uso público. Comenzaron a aparecer los primeros sitios web, y distintos navegadores que interpretaban **HTML** **.** El problema era que cada navegador tenía su propia forma de interpretar el lenguaje. Nuevamente, el mismo problema que había surgido en 1981: ¡había que estandarizar!

A medida que se masificó el uso de internet para fines comerciales, empezó a ser necesario agregar **estilos** y **diseños** con fines publicitarios.

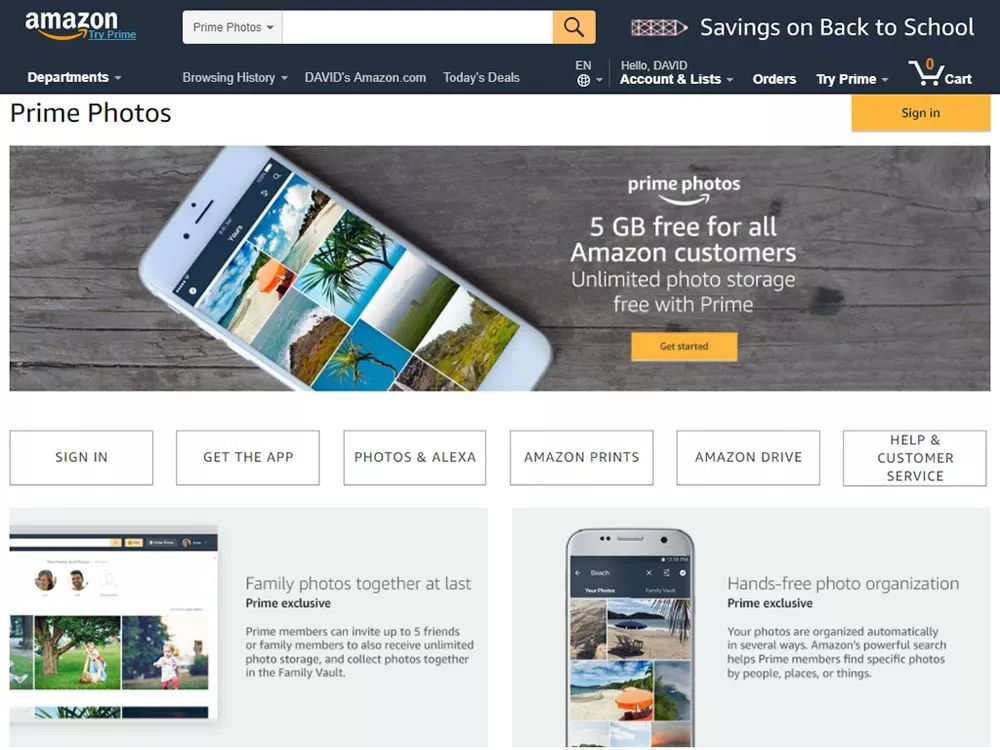
HTML proveía sólo algunas **etiquetas** y **atributos** para tal fin, pero provocaba que fuera necesario trabajar con cada etiqueta de forma individual.

Graphical user interface, text, application

Description automatically generated

Nace entonces CSS , **para poder aplicar estilos a diferentes documentos HTML y por lo tanto a diferentes etiquetas, de manera centralizada.**De esta manera, se abstrae la capa de estructura de la capa de presentación de los sitios web.

¡Gracias CSS!



La siguiente necesidad fue la de poder modificar los documentos HTML y sus estilos desde el navegador, sin la necesidad de traer un nuevo documento HTML con los cambios necesarios.

Para tal fin, en 1995 [Netscape,](https://es.wikipedia.org/wiki/Netscape_Navigator)a través de [Brendan Eich,](https://es.wikipedia.org/wiki/Brendan_Eich)creó el lenguaje de programación **JavaScript. El objetivo primario del lenguaje fue interceptar eventos que el usuario realiza sobre el sitio web (por ejemplo “un click”), y realizar una acción sin necesidad de recargar todo el documento HTML.**

# ¡Bienvenidos/as a la web interactiva!

Tim Berners-Lee crea entonces el **World Wide Web Consortium,** el cual hoy sigue presidiendo, y es el encargado día a día de mantener el estándar de los lenguajes de la web. Pasamos de desarrollar páginas web a aplicaciones web que puedan responder a peticiones de usuarios personalizadas.

Diagram

Description automatically generated

HTML, CSS y JavaScript —tres lenguajes que habíamos mencionado en nuestra primera bitácora (Introducción a Desarrollow Web Full Stack- Tecnología web- Servidor web)—componen el stack de tecnologías que utilizamos hasta hoy para desarrollar un sitio web y las que veremos en esta carrera como parte del Front-End. Lo que ves a la derecha en el gráfico (node.js y MySQL son lenguajes que se utiliza en Backend, y que abordaremos en el Bloque 3).

# Un lenguaje para diseñar y desarrollar páginas web

Ahora que ya sabes sobre el nacimiento de internet y sus funciones, nos vamos a sumergir en HTML.

**HTML es un lenguaje de Marcas de Hipertexto de la web .**Se basa en etiquetas y atributos que luego los browser se van a encargar de interpretar para mostrar el contenido jerarquizado de determinada manera.

En [este artículo](https://medium.com/@danvao/qu%C3%A9-es-html-7427a3bdcb6a) Dan nos aclara: “HTML no se considera un lenguaje de programación porque no “programa” comportamientos, simplemente se encarga de estructurar información, ni siquiera se encarga de manejar su propia imagen, de eso se encarga el CSS.”

Podemos verlo en un ejemplo de Roy Batty en su artículo [“Qué es HTML?”:](https://medium.com/@danvao/qu%C3%A9-es-html-7427a3bdcb6a)

Imaginemos una web donde sólo se nos plantea una pregunta: “¿Quieres ir a Google.com?” Y debajo, dos posibles respuestas: SÍ y NO; y dependiendo de la opción que seleccionemos, que el hipervínculo de rigor nos lleve a Google.com o a otra web con otra lista de navegadores. Nuestro documento tendría esta estructura:

<body>

¿Quieres ir a Google.com?</br>

- <a href="http://www.google.com">Sí</a></br>

- <a href="listanavegadores.html">No</a></br>

</body>

¿Es esto un programa? No, ¿verdad?

Las líneas que hemos escrito en HTML tan sólo son “formas de mostrarnos las cosas”, nada más, no ejecutamos ninguna rutina, subrutina, programa, subprograma, ni nada… nuestro código HTML no tiene que interpretar funciones simples o complejas, sólo llevarnos a uno u otro sitio, que hemos definido previamente.

Si aún no viste [**Introducción a HTML**](https://www.youtube.com/watch?v=xkqBvv1EZhk) ¡estás a tiempo de hacerlo! Nos vamos a encargar primero de aprender a escribir y entender este lenguaje que nos permite estructurar el contenido que vamos a brindarle a los/as usuarios/as.

# Cómo escribir en un lenguaje tecnológico

La sintaxis HTML se basa en **etiquetas** que son las encargadas de comunicarle al browser qué contenido debe mostrar. HTML dispone de etiquetas para imágenes, linkear documentos, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.

¿Cómo comenzar a escribir en HTML? Existen dos formas:

1. Las etiquetas que tienen un **inicio** y **cierre**

Se escriben abriendo el símbolo “menor que” (<) y se cierran con el símbolo “mayor que” (>):

<etiqueta>

Y se cierran con una etiqueta igual a la de apertura pero con una barra hacia la derecha (/) delante:

<etiqueta>

Contenido

</etiqueta>

1. Otras etiquetas no tienen cierre, son únicas, y tienen **atributos.**

Se cierran con una barra inclinada hacia la derecha al final (/) luego del último signo:

<img src:"imagen.jpg" alt="esto es una imagen" />

Los atributos son elementos que al completarse dan una funcionalidad extra a la etiqueta.

TIP: Es una buena práctica escribir las etiquetas HTML y los atributos en minúscula. Aunque pueden funcionar igual, no es costumbre escribirlas en mayúsculas.

# ¿Qué hay dentro de HTML?

Para comenzar, nuestra página web o aplicación web va a tener como archivo central al documento HTML que se complementa con los otros lenguajes. Es importante prestarle mucha atención y aprender a manejarlo bien, ya que es la forma en la que “encontrarán” el archivo que creamos.

Vamos a conocer la **estructura básica** de un documento HTML que se mantiene siempre igual.

<!DOCTYPE html>

<html>

 <head>

<title> Título aquí </title>

</head>

<body>

Contenido del sitio web aquí.

</body>

</html>

¿Qué son todas estas palabras **en azul** que estamos viendo? ¡No te asustes! Al principio suenan extrañas, pero se van a ir aclarando. Estos elementos son una serie de etiquetas que el navegador interpreta y da forma en la pantalla.

Nuestro documento inicia con la declaración <!DOCTYPE  html> que se escribe muy diferente de las demás. DOCTYPE le va a comunicar al browser la manera en la que el navegador procesará el documento.

La etiqueta <html> contendrá todo el documento. Observa que todas las etiquetas tienen una apertura, que se escribe <etiqueta> y un cierre, que se escribe </etiqueta> . Adentro hay dos etiquetas fundamentales:

* La etiqueta body es una de las más importantes ya que contendrá todas las etiquetas que compondrán el documento, y por lo tanto, la información que visualizará el usuario en la ventana del navegador.
* La etiqueta head por otro lado, tiene el rol de contener toda información, pero que no visualizará el usuario. La utilizamos para que el navegador pueda tomar decisiones sobre cómo presentar el documento al usuario. Dentro de esta etiqueta se encuentra la etiqueta title que va a mostrar en el browser el título de nuestro sitio. Dentro del head también podemos utilizar la etiqueta meta , y allí podremos describir nuestro sitio para ayudar a los buscadores a encontrarnos.

Hay que diferenciar elementos en línea y bloque de HTML para no generar confusiones:

* Las etiquetas inline no afectan al flujo de los demás elementos. Dejan que el texto y otros fluyan en la misma línea, colocándose uno al lado del otro.

<div>hola <span>mundo</span></div>

* Las etiquetas block cortan el flujo y se renderizan en el navegador en líneas independientes.

<div>hola <div>mundo</div></div>

No te preocupes por entender en detalle esto, pero es importante que lo vayas viendo para cuando lo trabajemos en los encuentros.

¿Conoces otras etiquetas? El conocimiento que traes es importante, puedes investigar hasta nuestro próximo encuentro. ¡Ya estás más cerca de tu primer sitio web!

**3 most useful extensions for Visual Studio Code**

**Live Server**

**Live SASS compiler:** Sirve para compilar archivos de SCSS a CSS.

**ESLint**

##### **BLOQUE 1 - ENCUENTRO 3**

# Metadatos - Formulario

Unirme

El encuentro comienza

**Hoy a las 19:00hs.**

## Lo que hay detrás de una solicitud de datos

**“La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.”—Aristóteles, Filósofo.**

## ****La importancia de crear formularios****

A close up of a computer

Description automatically generated

Como **usuarios/as** de internet, ¿cuántas veces nos encontramos llenando formularios para generar una cuenta nueva, ingresar a un sitio o simplemente pedir información en una página web para que te contacten?

Como **desarrolladores/as,** ¿qué pasa si los/as usuarios/as quieren enviarnos algún tipo de información o nosotros/as necesitamos datos para mejorar el servicio?

En la bitácora anterior (HTLM y Arquitectura Web) nos detuvimos en el rol que tiene HTML en la creación de páginas web y en la estructuración del contenido que vamos a brindarle al/la usuario/a. Dentro de HTML hay una variedad de etiquetas que podemos utilizar (¿recuerdas? <head>, <body>, <title> ), algunas de ellas dan funcionalidades tales como imágenes o formularios.

Según la definición de [MDN Web Docs de Mozilla:](https://developer.mozilla.org/es/docs/Learn/HTML/Forms/Your_first_HTML_form)

Los formularios web son uno de los principales puntos de interacción entre un usuario y un sitio web o aplicación. Los formularios permiten a los usuarios insertar datos, que generalmente se envían a un servidor web para su procesamiento y almacenamiento (....) o se usan en el lado del cliente para actualizar inmediatamente la interfaz de alguna manera (por ejemplo, agregar otro elemento a una lista, o mostrar u ocultar una función de interfaz de usuario).

Si estás pensando en cómo se va a ver no te apresures, todavia no importa la estética de la página. Lo importante en este punto es estructurar el contenido con las etiquetas correspondientes y entender cómo lo manipula el navegador. A lo largo de la carrera te encontrarás varias veces diseñando **formularios** dentro de tu sitio web o en una aplicación, por lo que es importante que entiendas las bases de cómo se estructuran.

## ****¿Cómo crear tus propios formularios?****

Al crear formularios, tendrás que pensar en tus usuarios finales y en cómo hacer para que les resulte sencillo completarlo, que no resulten confusos, y que los campos a llenar sean consistentes y coherentes.

Existen dos **etiquetas** básicas que te permiten generar **formularios:**

* Etiqueta form **:** esta etiqueta representa un formulario, y es necesario agregarle otras como la etiqueta action .
* Etiqueta input **:** lleva varios **atributos,** por ejemplo:
  + type
    - text: input de texto
    - checkbox: permite múltiples selecciones
    - radio: permite seleccionar solamente uno de una lista
    - email: input de texto que valida que exista un @
    - file: permite subir un archivo
    - hidden: input que no se visualiza
    - password: input de texto que no visualiza los caracteres
    - url: input de texto que corrobora una URL válida
    - submit: genera un botón para enviar el formulario
  + required : si es requerido o no
  + id : identifica la etiqueta HTML de otras y es para cualquiera, no sólo para formularios. Puede dar funcionalidades extra cuando trabajemos con formularios en JavaScript.
  + value : valor por defecto del input
  + name : identifica el campo en el formulario para poder manipularlo una vez que se envía.
* Etiqueta select : genera un menú desplegable, cada opción es creada con la etiqueta option .

El formulario se verá así:

Graphical user interface, text, application, email

Description automatically generated

¿Conoces otros formularios? Puedes buscar en internet más ejemplos para compartir en el próximo encuentro. Hay más etiquetas para generar formularios aunque suelen tener una estructura repetitiva, también las pueden investigar. En el encuentro practicaremos con ejemplos concretos con cada una de las etiquetas, sus componentes y para qué las utilizamos.

Prepara cualquier duda que tengas, los encuentros están para escucharte y ayudarte durante el proceso.

HTML Input Types

Here are the different input types you can use in HTML:

* <input type="button">
* <input type="checkbox">
* <input type="color">
* <input type="date">
* <input type="datetime-local">
* <input type="email">
* <input type="file">
* <input type="hidden">
* <input type="image">
* <input type="month">
* <input type="number">
* <input type="password">
* <input type="radio">
* <input type="range">
* <input type="reset">
* <input type="search">
* <input type="submit">
* <input type="tel">
* <input type="text">
* <input type="time">
* <input type="url">
* <input type="week">

OCT

#### **15**

##### **BLOQUE 1 - ENCUENTRO 4**

# Introducción a CSS

[Unirme](https://acamica.zoom.us/j/97799976702)

El encuentro comenzó hace

**02:53:46**

## El lenguaje puede darte tu estilo propio en la web

**“El lenguaje es la ciudad para cuya edificación cada ser humano ha aportado una piedra” —Ralph W. Emerson. Escritor, filósofo y poeta**

Graphical user interface, website

Description automatically generated

En la bitácora 2 (Del big bang de la web hasta la actualidad) vimos que los documentos HTML habían sido creados para divulgar información científica y con el uso masivo de internet fue necesario incorporar estilos y diseños con fines de expandir el alcance más allá de la comunidad científica.

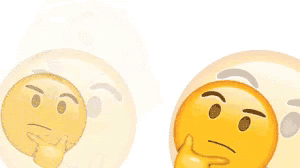
HTML sólo proveía algunas etiquetas y atributos básicos para darle estilo a los documentos pero, como interferían en su estructura, lo hacían poco escalable. Por estas necesidades surge **CSS (Cascading Style Sheets u Hojas de Estilo en Cascada).**

Veamos un ejemplo concreto de por qué necesitamos agregar CSS a nuestro sitio web:

Graphical user interface, text, application

Description automatically generated

## ¿Hojas de Estilo en Cascada?



Su nombre se basa en el concepto de reglas, que se van aplicando “en cascada” (porque se leen de arriba hacia abajo) para poder aplicar estilos a diferentes documentos HTML y por lo tanto a diferentes etiquetas, de manera centralizada (la fuente, el formato y la visualización).

Si te estás preguntando "¿cómo es que puedo utilizar CSS?" Aquí va una primera respuesta: deberás realizar algún tipo de conexión con el documento, y lo podrás hacer mediante **selectores** que determinan a qué conjunto de elementos se aplica la regla.

"...el problema más difícil en CSS: lograr que sus reglas coincidan con los elementos que desea, sin que coincidan accidentalmente con los elementos que no desea". (Philip Walton, ingeniero en Google)

¿Confuso? Vamos por partes.

## Modelo de caja

Si estamos por comenzar a intervenir nuestro documento para darle “vida estética”, es vital saber cómo funciona el lenguaje para poder **estilar** todos los elementos de HTML.

Imaginando al documento original como nuestro contenido, y teniendo en cuenta que existen tres elementos que lo rodean para definir el espacio que tendrán en una página web...

1. **margin (margen)**
2. **border (borde)**
3. **padding (relleno)**

...entonces, el documento web se verá así:

Graphical user interface, text, application, email

Description automatically generated

Para poder definir el aspecto de nuestro documento, primero debemos hacernos algunas preguntas elementales:

* ¿Qué tan ancho debería ser?
* ¿Cuán grueso debe ser el borde?
* ¿De qué color debería ser el fondo?
* ¿Qué tamaño de fuente deseamos/necesitamos?

**¡Ten cuidado!** Cada modificación que realices afectará a las demás.

El [modelo de cajas](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Modelo_Caja/Introducci%C3%B3n_al_modelo_de_caja_de_CSS) comprende un conjunto de propiedades además de las mencionadas. Éstas son algunas de las más usuales de CSS:

* background-color permite cambiar el color de fondo de cualquier elemento con esta característica.
* border permite cambiar los cuatro bordes de un elemento, y consta del tamaño del borde, el tipo y el color, por ejemplo: border:1px solid red se traduce como que todos los bordes serán de un 1 píxel de grosor, del tipo sólido, y de color rojo. Como sucede con el padding y el margin, se puede especificar cualquiera de los cuatro lados.
* box-sizing es una propiedad muy importante, ya que permite incluir el padding y el border dentro de las dimensiones totales del elemento, con el valor border-box.
  + En cambio, si usamos el valor content-box, el tamaño total del elemento sólo estará basado en su contenido, excluyendo el relleno y borde. Puedes ver una explicación visual [acá.](https://developer.mozilla.org/es/docs/Web/CSS/box-sizing)
* color cambia el color de la tipografía.
* width especifica el ancho de los elementos.
* height especifica el alto de los elementos.

Más adelante conoceremos propiedades que tienen que ver con el posicionamiento y presentación de los elementos, así como también propiedades específicas para la tipografía.

Hay una gran variedad de propiedades. Te dejamos algunos links para que te sumerjas y conozcas más sobre ellas: [**MDN, propiedades CSS,**](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference)[**Referencia CSS.**](https://cssreference.io/)

## ****El mundo de las unidades de medida****

En CSS es enorme la cantidad de unidades de medida. Por el momento vamos a aprender las básicas para, más adelante, adentrarnos en las más complejas.

Vamos a empezar mencionando las unidades de medida **absolutas** o **fijas.** Como su nombre lo indica, especifican una medida en forma exacta e independiente, que no varía en función del contenedor o del dispositivo desde el que se acceda.

* **pixel**(px) **.** Es la más usada de este tipo de unidades, y consta de la menor unidad de color en la que puede dividirse una imagen digital. La suma de muchos píxeles genera una imagen. Por ejemplo, si quisiéramos enmarcar una frase podríamos hacerlo de esta manera usando píxeles:

Graphical user interface, text, application

Description automatically generated

* **points**( pt ) **.** Esta medida tiene que ver con gráfica, donde cada uno de estos puntos es un punto que la impresora pinta en la hoja, por lo que no tiene sentido usarla para contenido que se verá en la pantalla de un dispositivo, pero sí en el caso de que lo que estamos maquetando vaya a ser impreso.

Luego, se encuentran otro tipo de unidades de medida que son **relativas,** ya que siempre se establecen en función de otros parámetros.

* **porcentaje**( % ). Lo que hace esta medida es tomar el porcentaje indicado según el tamaño de su elemento padre. Por ejemplo, si hay un contenedor que tiene 600px de ancho y dentro de él ponemos otro con un width del 50%, el contenedor hijo va a medir 300px.

## ****Le damos vida a las palabras****

Graphical user interface, application, website

Description automatically generated

La **tipografía** es una parte vital de las aplicaciones web, ya que gran parte del contenido es texto. Existen muchas propiedades que permiten modificar su estilo inicial: el color, el tamaño, el peso, el espacio entre las letras, el fondo y también la familia o tipo, entre otras.

Vamos a ejemplificar algunas de estas propiedades dando estilo a un elemento.

Para cambiar la familia de la tipografía, podemos usar la propiedad font-family , indicando el tipo de letra que queremos usar, por ejemplo ‘Helvetica’, o ‘Arial’.

TIP: una buena práctica para dar estilo a la tipografía incluye estos dos pasos:

* Definir primero la tipografía que sí o sí queremos mostrar.
* En la misma línea, y separadas por comas después de las comillas, incluimos otras dos alternativas: una con una tipografía en particular, y luego sans-serif o serif, según corresponda.

¿Por qué? En caso de que no existan o no se hayan podido cargar ninguna de las dos tipografías sugeridas, se usará cualquiera con serifas o sin serifas, según la indicación. El código de esta propiedad quedaría así:

Graphical user interface, text

Description automatically generated

Con este acercamiento a CSS estás listo/a para empezar a modificar estilos. En el encuentro, vamos a trabajar con la presentación visual del contenido y le daremos estilo.