# Módulo I

# Introducción al desarrollo web y aplicaciones

Te damos la bienvenida al primer módulo de #YoProgramo, en el encontraremos las respuestas de ¿por qué es importante programar y de qué se trata? Conocerás aspectos técnicos de cómo viaja la información por internet, cuáles son los elementos conceptuales que componen una aplicación web, también veremos cómo gestionar el tiempo de forma eficaz y conocerás las herramientas junto con la metodología de trabajo que se utilizan en las empresas para desarrollar un proyecto.

**Conocimientos previos**

* Fundamentos de la programación
* Programación imperativa y estructuras de datos
* Programación con Objetos

# ¿Qué es la programación?

Es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un proyecto, preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados o elaborar programas para su empleo en computadoras.

En la actualidad, la noción de programación se encuentra muy asociada a la creación de aplicaciones de informática y videojuegos. En este sentido, es el proceso por el cual una persona desarrolla un programa, valiéndose de una herramienta que le permite escribir el código (el cual puede estar en uno o varios lenguajes, como C ++, Java y Python, entre otros) y de otra que sea capaz de “traducirlo” a lo que se conoce como lenguaje de máquina, que puede "comprender" el microprocesador.

Pensamiento computacional: Proceso que permite abordar un problema complejo, comprenderlo y desarrollar las soluciones más adecuadas. Etapas:

* Descomposición: Desarticular el problema complejo en unidades más chicas para que sea mas fácil de resolver
* Patrón: Buscar similitudes dentro o fuera de la situación problemática
* Abstracción: Simplificar un problema complejo, basandose en lo importante
* Algoritmo: Pasos ordenados para el desarrollo del problema

Permite ampliar habilidades de comunicación, trabajo en equipo, poner en juego la creatividad, desarrollar pensamiento critico y razonamiento lógico.

¿Cómo se puede aplicar el pensamiento computacional a un problema concreto?

1. Analizar el problema
2. Tener en cuenta las variables mas significativas
3. Tomar deciisiones

¿Qué es un algoritmo?

Un algoritmo informático es un conjunto de instrucciones definidas, ordenadas y acotadas para resolver un problema o realizar una tarea. En programación, supone el paso previo a ponerse a escribir el código. Primero debemos encontrar la solución al problema (definir el algoritmo informático), para luego, a través del código, poder indicarle a la máquina qué acciones queremos que lleve a cabo. De este modo, un programa informático no sería más que un conjunto de algoritmos ordenados y codificados en un lenguaje de programación para poder ser ejecutados en un ordenador. Siempre tiene un input, un proceso dentro para llegar al resultado, que es el output

# Habilidades Digitales

## Habilidades digitales para programadores

**¿A qué nos referimos con habilidades digitales?**  
  
Las Tecnologías de información y comunicación (TIC) se han convertido en herramientas que intervienen en la mayor parte de las actividades laborales, académicas y recreativas de la vida actual. En nuestros días nos enfrentamos cotidianamente a situaciones de interacción social mediadas por las TIC: relaciones sociales, transacciones comerciales, trámites, consulta, intercambio y producción de información, situaciones de estudio, recreación, etc.  
  
Saber moverse en este mundo con alto uso de tecnologías de información y participar en los variados tipos de intercambios mediados por las TIC puede definirse como estar integrado a la cultura digital. Para esto es necesario contar con habilidades digitales. Entendemos por habilidades digitales el conjunto de saberes (saber hacer y saber sobre el hacer) relacionados con el uso de herramientas de comunicación, acceso, procesamiento y producción de la información.  
  
De manera resumida, las habilidades digitales no sólo remiten a aspectos operacionales, sino a aspectos formales, como cuál es la mejor solución a determinado problema, o la forma adecuada de encarar una dificultad tecnológica. Por ahora, haremos foco en habilidades de búsqueda.

**Uso de Keywords:**  
  
Lo primero y más importante es pensar bien los términos de búsqueda que vamos a usar. Cuando usamos algún motor de búsqueda tenemos algunos indicadores de eficiencia. Por ejemplo, generar menor cantidad de resultados de búsqueda implica que nuestra búsqueda ha sido más precisa. Pero menos no siempre es mejor. También podemos obtener menos resultados si no expresamos de manera correcta lo que estamos buscando.  
  
Por ejemplo, nos encontramos aprendiendo a usar [Githhub](https://developer.mozilla.org/es/docs/Learn/Common_questions/Using_Github_pages" \t "_blank), y queremos saber cómo clonar un repositorio, pero no de la rama principal, sino de una rama en específico.

1.      **Primera opción:** “como clonar un repositorio de github de una rama secundaria” (49,400 resultados)

2.      **Segunda opción:** “Clone secondary Branch github” (295,000 resultados)

3.      **Tercera opción:** “github clonar una rama específica” (43,600 resultados)

4.      **Cuarta opción:** “github clonar rama” (12,100,000 resultados)

Como vemos, la cantidad de resultados puede ser un indicador, pero no es el único… Por supuesto si utilizamos términos en inglés tendremos más resultados. Algunos consejos que te podemos dar son:

1.      Analiza tu problema, planea el trabajo de búsqueda.

2.      Piensa y anota qué posibles términos de búsqueda usar y vete ampliando la lista.

3.      Considera qué herramientas de búsqueda emplear, usa varias herramientas (veremos más adelante).

4.      Examina cómo se utilizan las herramientas de búsqueda, aprende a usarlas.

5.      Usa sistemas de búsqueda avanzada (campos, frases, limitaciones, combinación…).

6.      Busca ordenada, sistemáticamente, en varios pasos lógicos, sin precipitación.

7.      Guarda resultados provisionales, analízalos después, conserva los definitivos, etc.

8.      Valora críticamente los resultados, piensa si son relevantes, de un nivel adecuado…

Recuerda que las búsquedas relacionadas en programación regularmente suelen hacerse en Ingles por lo que es importante que siempre trates de hacer la búsqueda en Ingles por ejemplo : “How to create an array javascript”

## Uso de Operadores de búsqueda

Los operadores de búsqueda son una serie de símbolos y comandos que sirven para acotar los resultados de las búsquedas que realizas. Algunos de ellos incluyen símbolos como el de puntuación, que Google siempre ignora en el caso de que no pertenezcan a un operador concreto.

Son como filtros añadidos, sólo que en vez de estar en las opciones que ves debajo de la barra de búsqueda de Google cuando te muestra los resultados tienes que incluirlos en la propia búsqueda acompañando al término que quieras encontrar. Hay incluso algunos que sirven como enlace para poder utilizar más de uno a la vez.

La forma en la que funcionan es muy sencilla. Lo único que tienes que hacer es escribir un texto en la barra de búsqueda de Google en el que se incluya el operador que quieres utilizar. Habrá veces en las que este operador estará entre comandos de búsqueda o al final de cada uno, dependiendo de lo que quieras hacer con él.

Operadores de búsqueda de Google

| **OPERADOR** | **EJEMPLO** | **QUÉ HACE** |
| --- | --- | --- |
| **OR** | Pelota OR palo OR paso | Te muestra resultados que contengan cualquiera de las palabras que hayas incluido. |
| **AND** | JavaScript and Sintaxis | Busca páginas que incluya los dos términos especificados. |
| **" "** | "Github flow" o "Subversion" | Te muestra resultados donde aparece el término o los términos exactos que hayas añadido entre los ". |
| **-** | Fullstack -MEAN | Te muestra resultados donde se excluya la palabra que hayas puesto detrás del -. |
| **\*** | "SCRUM \*meeting" | Un comodín que puede coincidir con cualquier palabra en la búsqueda. |
| **#..#** | Celular 20000..50000 pesos | Te muestra resultados donde donde se añade un intervalo de números que tú especificas. |
| **(  )** | ("redes sociales" OR "plataformas sociales") -Twitter | Te permite combinar operadores. En el ejemplo buscarás redes sociales o plataformas sociales, pero excluyendo Twitter de los resultados. |
| **AROUND** | Trucos around(3) Instagram | Resultados donde aparecen las dos palabras especificadas, pero con el número que determines de términos entre ellas. |
| **EN** | 300 dólares en euros | Sirve para convertir unidades de un mismo tipo de medida. |
| **MAP** | map:BuenosAires | La búsqueda te devuelve resultados con mapas del sitio donde le digas. |
| **DEFINE** | define:Lunfardo | Busca la definición de una palabra que no conozcas |
| **SITE** | Cursos site:[www.argentina.gob.ar](http://www.argentina.gob.ar/) | Te busca los resultados dentro de una web que hayas especificado. |
| **INFO** | info:[www.argentina.gob.ar/](http://www.argentina.gob.ar/) | Te muestra resultados donde se ofrezca información sobre una página web. |
| **RELATED** | related:[www.argentina.gob.ar/](http://www.argentina.gob.ar/) | Te muestra en los resultados otras páginas similares a la que has escrito. |
| **LINK** | Teléfonos link:<https://developer.mozilla.org/> | Te muestra en los resultados páginas que tienen enlaces a la web que hayas especificado. |
| **CACHÉ** | cache:<https://developer.mozilla.org/> | Te muestra la copia de la página que hay en el caché de Google. |
| **FILETYPE** | filetype:pdf presupuestos 2021 | Busca resultados que contengan archivos con el formato que hayas especificado |
| **ALLINTEXT O INTEXT** | allintext:"desarrollador fullstack" | Encuentra páginas que incluyan en su texto algunos o todos los términos que hayas incluido en el comando. |
| **ALLINTITLE O INTITLE** | allintitle:recursos desarrollador web | Te muestra páginas que tengan algunos o todos los términos que hayas incluido en el comando en su título |
| **INURL O ALLINURL** | inurl:"apple iphone" allinurl:"apple sfera" | Te muestra páginas con algunos o todos los términos que hayas incluido en el comando |
| **ALLINANCHOR O INANCHOR** | allinanchor:"desarrollador fullstack" | Resultados con páginas donde se incluya un enlace con un texto anclado donde se incluya uno o varios términos especificados. |
| **STOCKS** | stocks:Facebook | Busca el estado actual de la empresa que busques en bolsa. |
| **WEATHER** | weather:Rosario,ar | El tiempo en la ciudad elegida. Mira que después del nombre de la ciudad puedes poner una coma y el país para ser más concreto. |
| **TIME** | time:Nueva York | Te muestra la hora en la localidad que decidas. |
| **MOVIE** | movie:Avengers | Te muestra resultados relacionados con una película que establezcas. |
| **@** | @ArgentinaPrograma | Busca etiquetas sociales asociadas con Twitter. |
| **#** | #ArgentinaPrograma | Busca términos publicados con hastags en redes sociales que tengan sistema de hashtags. |

## Alternativas Motores de Búsqueda

¿Dónde buscar?

Empecemos por un pequeño detalle. Conviene tener presente que Google no es el único buscador generalista, es decir, orientado a rastrear y localizar sitios y páginas web de cualquier clase. Hay otros, como, por ejemplo:

1. Yahoo search
2. Bing
3. Exalead
4. Startpage
5. Duckduckgo
6. Ask
7. Search encrypt
8. Brave

Continuemos con el ejemplo anterior... En la mayoría de búsquedas que realizamos no pasamos de la primera hoja de resultados. Además, como ya mencionamos, los criterios de indexación de los resultados son diferentes, por lo que los resultados serán diferentes, o estarán presentados en un orden diferente.

Google es el buscador más conocido y utilizado. Quizá es el que más información rastrea de internet y también el que en líneas generales mejor lo hace. Pero no hay que menospreciar la capacidad de sus rivales de encontrar resultados preferibles para algunos problemas, ni las prestaciones especiales de algunos, como Exalead o StartPage.

Pedir a veces una segunda opinión, tener otros buscadores de reserva a la hora de explorar la web, es buena idea. Puede resultarnos de utilidad.

También en caso de necesitarlo en algún momento, hay motores de búsqueda especializados en contenido académico, por ejemplo:

* Google Scholar
* Microsoft academic
* Base
* World Wide Science

## Herramientas de desarrollador

#### **¿Cuáles son las herramientas de desarrollo del navegador?**

Todos los navegadores web modernos incluyen un potente conjunto de herramientas para desarrolladores. Estas herramientas hacen una variedad de cosas, desde inspeccionar HTML, CSS y JavaScript actualmente cargados, hasta mostrar qué archivos ha solicitado la página y cuánto tiempo tardaron en cargarse. Vamos a explicar cómo utilizar las funciones básicas de las herramientas de desarrollo del navegador.

##### **Cómo abrir devtools en tu navegador**

Las herramientas para desarrolladores (devtools) viven dentro de tu navegador en una subventana que se ve más o menos así, dependiendo del navegador que estés utilizando:

##### **Levantar DevTools: Existen tres distintas maneras:**

* ***Teclado:*** *Ctrl*+*Mayús*+*I*, excepto en
  + **Internet Explorer y Edge:**F12
  + **macOS:***⌘*+*⌥*+*I*
* ***Barra de menú:***
  + **Firefox**: Menú➤ Desarrollador web ➤ Alternar herramientas, o ➤ Herramientas ➤ Alternar herramientas del desarrollador web
  + **Chrome:** Más herramientas ➤ Herramientas del desarrollador
  + **Safari:** Desarrollador ➤ Mostrar el inspector web. Si no puedes ver el menú Desarrollar, ve a Safari ➤ Preferencias ➤ Avanzado y marca la casilla de verificación Mostrar menú desarrollador en la barra de menú.
  + **Opera**: Desarrollador ➤ Herramientas para desarrolladores
* **Menú contextual:** Presiona y mantén presionado / haz clic con el botón derecho en un elemento en una página web (Ctrl-clic en Mac) y elige Inspeccionar elemento en el menú contextual que aparece. (Una ventaja adicional: este método, inmediatamente resalta el código del elemento en el que hiciste clic con el botón derecho).

Las herramientas del desarrollador, generalmente se abren de forma predeterminada en el inspector. Esta herramienta muestra cómo se ve el HTML en tu página en tiempo de ejecución, así como qué CSS se aplica a cada elemento de la página. También te permite modificar instantáneamente el HTML y CSS y ver los resultados de tus cambios reflejados en vivo en la ventana del navegador.

Además podemos encontrar el depurador de JavaScript, el cual te permite observar el valor de las variables y establecer puntos de interrupción, lugares en tu código en los que deseas pausar la ejecución e identificar los problemas que impiden que tu código se ejecute correctamente.

Otro componente importante es la consola de JavaScript  que es una herramienta increíblemente útil para depurar JavaScript que no funciona como se esperaba. Te permite ejecutar líneas de JavaScript en la página actualmente cargada en el navegador e informa los errores encontrados cuando el navegador intenta ejecutar tu código.

## Herramientas para programadores

Empecemos por algunos puntos que nos van a ser útiles…

#### **Traductor**

El clásico es GoogleTranslate. Revisemos algunos detalles de su uso.

El Traductor de Google es un sistema multilingüe de traducción automática, desarrollado y proporcionado por Google, para traducir texto, voz, imágenes o video en tiempo real de un idioma a otro.  El Traductor de Google posee la capacidad de traducir en más de 100 idiomas en distintos niveles, el sistema provee un servicio gratuito y diariamente es utilizado por más de 200 millones de personas.

#### **Licenciamiento y versionado**

El objetivo de este apartado será un repaso superficial respecto los tipos de licenciamiento de software, si quieres profundizar más puedes investigarlo como “derecho informático” dado que es una disciplina diferente a la de desarrollo de software.

Existen varios criterios de clasificación para dividir los tipos de licencias de software. El la imagen de mapa de licencia facilita la comprensión e incorpora ejemplos para cada tipo de licencia. En el caso del software propietario, las licencias de software van a depender del titular de los derechos de autor del software en cuestión, que normalmente va a ser quien lo crea o quien lo ofrece. Veamos una pequeña descripción de las más importantes:

**Licencias de software Shareware**

Corresponden a un tipo de distribución de aplicaciones que consiste en liberar gratuitamente una versión con funcionamiento limitado. Esa limitación puede ser temporal (después de determinada cantidad de días deja de operar), por funciones (desde el comienzo, o a partir de determinado momento, hay funciones que el programa deja de realizar) o una combinación de las mencionadas (el programa empieza con todas sus funciones y deja de realizar algunas al cabo de cierto tiempo).

**Licencias de software libre**

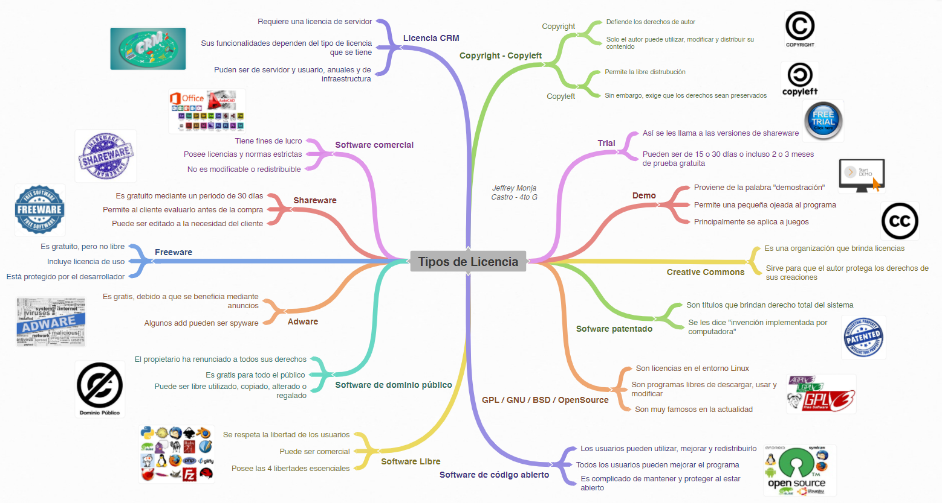
En el caso del software libre, si bien cada desarrollador puede utilizar la licencia que desee, como ocurre con el software propietario, está más extendida la práctica de licenciar un software libre bajo determinadas licencias creadas principalmente por organizaciones, como la licencia General Public License (GNU GPL), creada y promovida por la Free Software Foundation, la licencia Apache, la licencia Mozilla Public License, creada y promovida por la Mozilla Foundation y usada en su producto más conocido, el navegador web Mozilla Firefox, entre otras.

Suele ocurrir con el software libre que, al ser un conjunto de aportes de distintos desarrolladores, cada “parte” de un software tenga licencias distintas. Así ocurre, por ejemplo, con el sistema operativo móvil Android y la iniciativa de código abierto que lo construye y mantiene, el Android Open Source Project (AOSP), en castellano Proyecto de Código Abierto de Android. En su sitio web, el AOSP recomienda el uso de la licencia Apache para liberar el software relacionado con Android. Sin embargo, hay partes de Android que se liberan bajo otras licencias, como la licencia GPL para el núcleo (kernel) del sistema, puesto que es el núcleo del sistema operativo Linux, utilizado no solamente como base de Android sino también como base para sistemas operativos de escritorio, de servidores, de sistemas embebidos (por ejemplo, cajeros automáticos o navegadores GPS), que está liberado bajo la licencia GPL.

**Otros tipos de licencias de software**

Además de esta distinción entre licencias de software libre y licencias de software propietario, existen otras clasificaciones de las licencias:

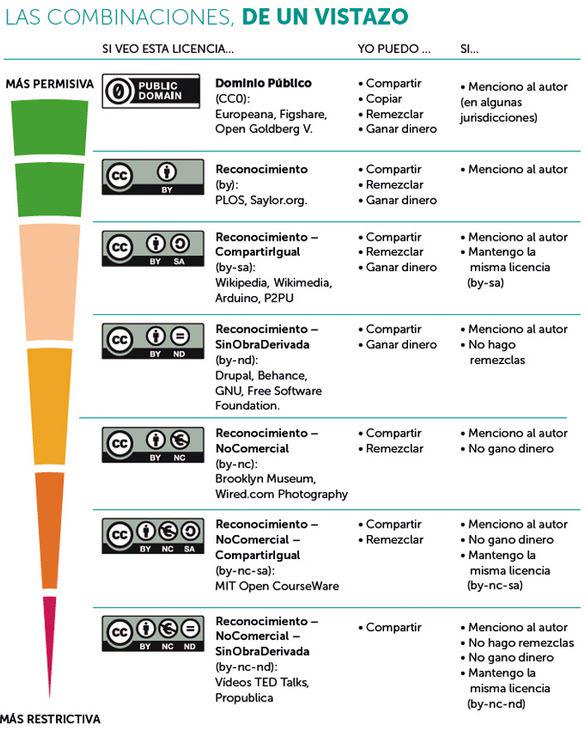
* • Según el grado de libertad de uso que se le entrega al licenciatario.
* • Según el grado de estandarización de los términos de la licencia.
* • Según la forma de celebración del contrato.
* • Según el grado de estandarización, licencias de software genérico (empaquetado) con contratos de adhesión y licencias de software personalizado. Las licencias otorgadas mediante contratos de adhesión se llaman también licencias shrink-wrap.
* • Según la forma de celebración del contrato, distingue entre licencias celebradas por escrito; licencias celebradas por otros medios válidos de expresión del consentimiento y licencias celebradas por medios electrónicos. Hay que señalar que las licencias celebradas por escrito sería un contrato consensual, que se perfecciona por el solo consentimiento de las partes.



En el caso de las licencias celebradas por otros medios válidos de expresión del consentimiento, cuya aceptación de los términos de la licencia es tácita, se distinguen dos casos:

1. Cuando existen actos positivos por parte de quien adquiere el software.
2. Cuando existe silencio o inactividad por parte de quien adquiere el software.

En algunas doctrinas, se establece que la aceptación tácita debe ser manifestada por hechos inequívocos de ejecución del contrato propuesto.



#### **Sitios de proyectos y comunidades:**

Todas las tecnologías tienen y cuentan con documentación oficial, para acceder a ella es importante que conozcas el origen de ella. Aquí te damos algunos ejemplos con las tecnologías que trabajaras en este curso y que te ayudarán durante toda tu vida como programador:

• Documentación oficial HTML - <https://developer.mozilla.org/es/docs/Web/HTML>

• Documentación oficial CSS - <https://developer.mozilla.org/es/docs/Web/CSS>

• Documentación oficial JavaScript - <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference>

• Documentación oficial Angular - <https://angular.io/docs>

• Documentación oficial Java -  [JDK 11 Documentation - Home (oracle.com)](https://docs.oracle.com/en/java/javase/11/)

• Documentación oficial SpringBoot - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

• Página Oracle University (Java) - <https://education.oracle.com/software/java/pFamily_48>

#### **Stackoverflow:**

La descripción que hace la propia página web del servicio nos parece lo suficientemente concreta y precisa, por lo que no hace falta añadir demasiado a la descripción.

“***Stack Overflow*** es una comunidad abierta para cualquiera que codifique. Lo ayudamos a obtener respuestas a sus preguntas de codificación más difíciles, compartir conocimientos con sus compañeros de trabajo en privado y encontrar el próximo trabajo de sus sueños”

<https://es.stackoverflow.com/>

Si eres un desarrollador, puedes acceder a diferentes funciones y características de gran utilidad. Entre ellas, se encuentran las siguientes:

1. Obtener respuestas a más de 16.5 millones de preguntas
2. Retribuir el apoyo compartiendo conocimiento con otros desarrolladores
3. Compartir información de manera privada con compañeros de trabajo
4. Encontrar trabajo adecuado a través de listados de alta calidad, filtrando mediante título, tecnología, salario, ubicación y otras variables

## Glosario de recursos útiles

##### **Cursos de Programación (y más):**

Udacity: <https://www.udacity.com/>

Coursera: <https://es.coursera.org/>

Canal Youtube MIT: <https://www.youtube.com/c/mitocw>

EdX: <https://www.edx.org/es/>

##### **Herramientas CSS:**

CSS Gradient: <https://cssgradient.io/>

CSS Layout: <https://csslayout.io/>

Getwaves: <https://getwaves.io/>

WebGradients: <https://webgradients.com/>

Cubic-bezier: <https://cubic-bezier.com/#.17,.67,.83,.67>

CSS3: <https://www.css3.me/>

CSS-Minifier: <https://cssminifier.com/>

##### **Herramientas BootStrap:**

Icons.getbootstrap: <https://icons.getbootstrap.com/>

Themes.getbootstrap: <https://themes.getbootstrap.com/>

Bootswatch: <https://bootswatch.com/>

Getbootstrap: <https://getbootstrap.com/>

##### **Herramientas HTML:**

HTML-minifier: <https://html-minifier.com/>

HTML ColorCode: <https://htmlcolorcodes.com/es/>

W3schools: <https://www.w3schools.com/>

##### **Herramientas e Imágenes Alta resolución:**

Freepik: <https://www.freepik.es/>

Pexels: <https://www.pexels.com/es-es/>

MotionArray: <https://motionarray.com/>

UnDraw: <https://undraw.co/>

UnSplash: <https://unsplash.com/>

FreeImages: <https://www.freeimages.com/es>

Pixabay: <https://pixabay.com/es/>

Thenounproject: <https://thenounproject.com/>

Tinypng: <https://tinypng.com/>

Manypixels: <https://www.manypixels.co/>

Humaans: <https://www.humaaans.com/>

Uigradients: <https://uigradients.com/#CocoaaIce>

Flaticon: <https://www.flaticon.es/>

Boxicons: <https://boxicons.com/>

##### **Herramientas FrontEnd:**

Codepen: <https://codepen.io/>

Modernizr: <https://modernizr.com/>

BrowserShots: <https://browsershots.org/>

CDNJS: <https://cdnjs.com/>

Waybackmachine: <https://archive.org/web/>

Dummyimage: <https://dummyimage.com/>

Colorzilla: <https://www.colorzilla.com/>

Caniuse: <https://caniuse.com/>

Spritecow: <http://www.spritecow.com/>

Figma: <https://www.figma.com/>

JavasScript-minifier: <https://www.minifier.org/>

##### **Fuentes:**

fonts.google: <https://fonts.google.com/>

Es.lipsum: <https://es.lipsum.com/>

FontAwesome: <https://fontawesome.com/>

##### **Editores de código:**

VScode: <https://code.visualstudio.com/>

##### **VCS:**

Subversión: <https://subversion.apache.org/>

Mercurial: <https://www.mercurial-scm.org/>

Git: <https://git-scm.com/>

Github: <https://github.com/>

Gitlab: <https://about.gitlab.com/>

Bitbucket: <https://bitbucket.org/>

##### **Programas para VCS:**

Github-desktop: <https://desktop.github.com/>

Git-kraken: <https://www.gitkraken.com/>

##### **Sistema de persistencia:**

SQLite: <https://www.sqlite.org/index.html>

MariaDb: <https://mariadb.org/>

MongoDb: <https://www.mongodb.com/es>

Oracle: <https://www.oracle.com/ar/index.html>

PostgreSQL: <https://www.postgresql.org/>

Redis: <https://redis.io/>

MySQL: <https://www.mysql.com/>

DynamoDb: <https://aws.amazon.com/es/dynamodb/>

AmazonRelationDataBaseService: <https://aws.amazon.com/es/rds/>

##### **Navegadores:**

Chrome: <https://www.google.com/intl/es-419/chrome/>

Brave: <https://brave.com/es/>

Chrome Canary: <https://www.google.com/intl/es-419/chrome/canary/>

Chromium: <https://www.chromium.org/>

Firefox Developer Edition: <https://www.mozilla.org/es-AR/firefox/developer/>

##### **Servicios web:**

Rest API

Graphql API – Serverless Firebase

AWS Amplify

Cloudinary

##### **Desarrollo API:**

Postman: <https://www.postman.com/product/graphql-client/>

Rest Client de VSCode: <https://marketplace.visualstudio.com/items?itemName=humao.rest-client>

##### **Servicios cloud:**

Azure: <https://azure.microsoft.com/es-es/>

Amazon Web Services: <https://aws.amazon.com/es/>

GoogleCloud: <https://cloud.google.com/>

Docker - para empaquetar: <https://www.docker.com/>

##### **CMS:**

Wordpress: <https://wordpress.com/es/>

Woocommerce: <https://woocommerce.com/>

Magento: <https://magento.com/>

Blogger: <https://www.blogger.com/about/>

Ghost: <https://ghost.org/>

##### **Framework de Backend:**

GO (Gorilla, Buffalo, goji)

Python (django, flask)

Typescript/JavaScript/Node (Loopback, Nest, Next.js, Nuxts.js)

##### **Servers:**

Nginx: <https://www.nginx.com/>

Apache: <https://www.apache.org/>

Windowsiis: <https://www.iis.net/>

Tomcat: <http://tomcat.apache.org/>

##### **Mobiledev**

ionic (angular react vim )

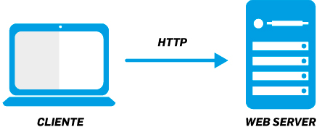
react native + expo, flutter

# Arquitectura Web

## Arquitectura Web Infografía

**Breve reseña**

Se trata entonces de una arquitectura cliente-servidor en la que cada dispositivo electrónico en la red (internet, intranet o extranet) actúa como cliente o servidor lo que implica la comunicación entre procesos hacen peticiones (clientes) y procesos que responden a esas peticiones (servidores). Esta comunicación es posible gracias al protocolo HTTP.



En 1994 (1 de octubre) Tim Berners-Lee abadona el CERN y funda la W3C, en ingles, “World Wide Web Consortium”, organismo internacional que propone recomendaciones y estándares web que aseguran el crecimiento de la World Wide Web.

**Arquitectura de las aplicaciones web**

Las aplicaciones web se basan en una arquitectura cliente/servidor. Es decir que, por un lado esta el cliente (navegador) y por otro lado el servidor. Existen diferentes variantes de la arquitectura básica según como se implemente, pero es importante mencionar que en tecnología la mayoría de las estructuras esta compuesta por capas.

Generalmente las historias se escriben en lenguaje que el usuario pueda entender y que refleje una descripción sintetizada de lo que este desea. En lo posible se debe tratar de eliminar ambigüedades y malas interpretaciones.

Un grupo de paginas web dinámicas se concepctualiza como Front End (pensadas para que el cliente aceeda) y el otro grupo de paginas dinámicas web como Back End (pensadas para el procesamiento y acceso a datos), además de que la base de datos puede existir otro Servidor. Esto da lugar a un concepto muy importante en POO el DESACOPLAMIENTO, en este caso el diseño del Front End, Back End y Base de Datos puede desacoplarse. Pero eso lo veremos mas adelante.

**Elementos de la arquitectura web**

A continuación, se enumeran los elementos de la arquitectura web (pueden variar según la arquitectura elegida):

* La infraestructura de red
* ISP
* Cliente Web
* Nombre de dominio
* URL
* Sitio Web
* Servidor DNS
* Hosting
* Servidor Web
* Contenedor de aplicaciones Web (o servidor de aplicaciones web)
* Servidor de base de datos.

Ahora que entendemos los principios básicos de la arquitectura WEB y algunos de sus elementos, veamos el Modelo OSI.

El modelo OSI o Modelo Interconexion de Sistemas Abiertas (en ingles Open Systems Interconection). Es un modelo de comunicación de 7 capas, y es la base por el cual viaja toda la información por las redes (internet global, internet local, internet celular, etc.)

**El viaje de la información por Internet**

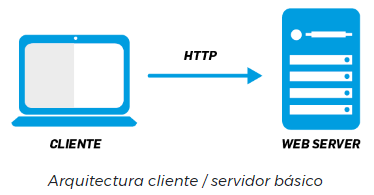
1. El usuario escribe una dirección web en el navegador
2. El protocolo HTTP es el que utilizan los servidores para enviar y recibir documentos a través de internet
3. El router enviara la solicitud al proveedor de acceso a internet, como Movistar o Jazztel
4. La solicitud se conecta con el servidor de nombres de dominio. Esta maquina traduce el nombre del dominio ([www.facebook.com](http://www.facebook.com)) a su IP (157.240.20.35)
5. Gracias al protocolo TCP, que divide el mensaje en paquetes, la información que enviamos y recibimos recorre su camino sin problemas.
6. El servidor host (Facebook) ejecuta la solicitud, y devuelve código HTML en el ordenador del usuario
7. Entonces el navegador web del usuario recibe la información y procesa el HTML para que se muestre como una página web visual

Breve reseña

 A partir del desarrollo de ARPANET en 1969 empieza un crecimiento vertiginoso del uso de la internet. En 1990 [Tim Berners-Lee](https://www.w3.org/People/Berners-Lee/)creó la **WWW**, la "WorldWideWeb" que realizó la primera conexión desde un navegador a un servidor web mientras trabajaba en el CERN desarrollando así, las tres tecnologías fundamentales de la web que son:

* ● **HTML**(lenguaje de marcado de hipertexto). Lenguaje de marcado o etiquetado que se emplea para escribir los documentos o páginas web.
* ● **URL**(localizador de recursos universal). El localizador de recursos uniforme, sistema de localización o direccionamiento de los documentos web.
* ● **HTTP**(Protocolo de transferencia de hipertexto) El lenguaje con el que se comunica el navegador con el servidor web y el que se emplea para transmitir los documentos web.

 Se trata entonces de una arquitectura cliente-servidor en la que cada dispositivo electrónico en la red ( [internet](https://dle.rae.es/internet), [intranet](https://definicion.de/intranet/)o [extranet](https://www.definicionabc.com/tecnologia/extranet.php)) actúa como cliente o servidor lo que implica la comunicación entre procesos que hacen peticiones (clientes) y procesos que responden a esas peticiones (servidores). Esta comunicación es posible gracias al protocolo HTTP.



En 1994 (1 de octubre) Tim Berners-Lee abandona el CERN y funda la [W3C](https://www.w3.org/), en inglés, "World Wide Web Consortium", organismo internacional que propone recomendaciones y estándares web que aseguran el crecimiento de la World Wide Web.

Haciendo [clic aquí](https://sites.google.com/site/elrendimientoscolar/evolucion-de-la-web/web-1-0-2-0-3-0-4-0)podrás ver la evolución de la web. Mencionaremos los hitos más relevantes:

* ● En 1991 surge **HTTP** definido como "protocolo de red para sistemas de información hipermedia distribuidos".
* ● Muy próximo aparece **HTML** 1, es el lenguaje de marcado predominante de las páginas web.
* ● En 1995, Netscape creó **JavaScript** , un lenguaje de secuencias de comandos basado en prototipos y "orientado a objetos". El objetivo de este lenguaje de programación fue darle capacidad de ejecución al cliente de esta arquitectura web, es decir, al navegador.
* ● En 1998 aparecen las hojas de estilo, en su versión 2. Se denominaron **CSS** , del inglés "Cascading Style Sheets", que es un lenguaje de hojas de estilo empleado para describir la semántica de presentación de un documento, en este caso un documento web.

## Arquitectura de las aplicaciones Web

Las aplicaciones web se basan en una arquitectura cliente / servidor. Es decir que, por un lado está el cliente (navegador) y por otro lado el servidor. Existen diferentes variantes de la arquitectura básica según como se implementa, pero es importante mencionar que en la tecnología la mayoría de las estructuras están compuestas por capas.

A continuación, enumeramos algunas de las arquitecturas más comunes:

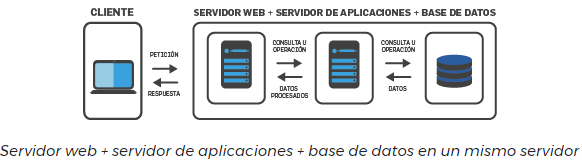
* Servidor web + base de datos en un mismo servidor (2 niveles o capas). En este caso el servidor gestiona tanto la lógica de negocio como la lógica de los datos y los datos



* Servidor web y de datos separados (3 niveles). En este caso se separa la lógica de negocio a la de datos en diferentes servidores.



* Servidor web + servidor de aplicaciones + base de datos en un mismo servidor (4 niveles).



Como vemos, la arquitectura web tiene un patrón de diseño en capas (arquitectura distribuida), y cada capa puede estar en un servidor diferente y aun así se pueden interconectar. El objetivo de separar las distintas funcionalidades en distintos servidores es aumentar la escalabilidad y el rendimiento. Por ejemplo, el servidor web al ofrecer servicios de http deberá tener una buena conexión a internet mientras que el servidor de base de datos requiere tener una buena capacidad de almacenamiento y procesamiento.

Un grupo de páginas web dinámicas se conceptualiza como Front End (pensadas para que el cliente acceda) y el otro grupo de páginas dinámicas web como Back End (pensadas para el procesamiento y acceso a datos), además de que la base de datos puede existir en otro Servidor. Esto da lugar a un concepto muy importante en POO el DESACOPLAMIENTO, en este caso el diseño del Front End, Back End y Base de Datos puede desacoplarse.

Ahora veamos distintos escenarios de arquitecturas web, debemos tener presente que nos referimos a como se dividen o distribuyen los distintos componentes de una aplicación en una red, en algunos casos dependiendo su complejidad del desarrollo necesita ser distribuida en varios servidores. La complejidad es determinada por las necesidades de la problemática que se pretende resolver o el diseño de la aplicación. Hagamos un repaso por las distintas arquitecturas que se nos pueden presentar en la siguiente imagen.

Analicemos juntos con más detalles cada escenario planteado.

* **En el escenario 1:**Vemos una páginas web con contenido estática, es decir, no tiene conexión con ningún servidor, significa que no actualiza su información y solo se puede navegar dentro de la misma página. Para crearla se utilizó HTML, CSS y JavaScript

****

* **En el escenario 2:**Vemos una arquitectura centralizada (todo está ubicado en el mismo sitio), contiene varias páginas web con contenido estática, a esto le llamamos sitio web estático, como en el caso anterior tampoco tiene conexión con otro servidor lo que significa que solo se puede navegar entre las mismas páginas. Para crearla se utilizó HTML, CSS y JavaScript.

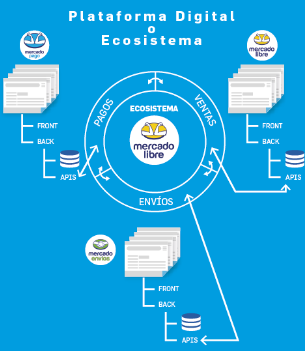


* **En el escenario 3:**Vemos una Aplicación Web o Web Dinámica, en este caso además de utilizar HTML, CSS y JavaScript se utilizaron lenguajes de programación para poder hacer intercambio de información con las distintas capas de la aplicación, hacer cálculos, crear nueva información en la base de datos, actualizarla o conectar con otros sistemas mediante API. Esta arquitectura tienen la siguiente separación conceptual o de arquitectura:
  + **Front End:** Es el nombre conceptual que se le da al código programado o la parte de la aplicación web que ve un usuario cuando entra desde el navegador a nuestra  aplicación. Por ejemplo cuando entras a una página como Mercado libre lo que ves en tu navegador es el Front End.
  + **Back End:** Es el nombre conceptual que se le da al código programado o la parte de la aplicación web *que no se ve a simple vista pero ejecuta acciones que pide el usuario desde el Front End*, es decir, al realizar una búsqueda de un producto determinado dentro de la web de Mercado Libre, el encargado de buscar el producto es el Back End y el encargado de mostrar el producto encontrado al usuario es el Front End.
  + **Conexión a BD:**Es el nombre que se le da a la conexión con la base de datos, allí se guardan todos los productos en el caso de Mercado Libre, de esta manera facilita el almacenamiento y búsquedas de los datos. Más adelante profundizaremos más sobre bases de datos.
  + **Apis:**Por el momento diremos que las Apis nos permiten conectemos con otros sistemas o bien que otros sistemas se conecten con el nuestro, más adelante iremos profundizando más en el tema.

Si no has visto en la graficas estos nombres conceptuales en el escenario 3, te invitamos a que vuelvas a mirarlo para ir asociando el concepto.



* **En el escenario 4:**Vemos una Plataforma Digital o un Ecosistema, en este caso podemos ver que consiste en muchos sistemas que trabajan en conjunto, colaborando para resolver una necesidad o problema. En el ejemplo de Mercado Libre se dedica a vender online, pero también tiene que cobrar y hacer envíos. Por eso han desarrollado aplicaciones independientes pero que saben cómo comunicarse a otros sistemas para pedir o enviar datos para realizar alguna tarea. De esta manera las aplicaciones pueden dar solución integral, comprar, pagar y enviar el producto sin tener que salir de la página. Esta arquitectura es más compleja porque son varios los sistemas que componen el ecosistema, pero si miras con atención el grafico, notaras que existen los mismos elementos que describimos anterior, revisemos:
  + **Front End**: Parte de la aplicación web que ve un usuario al entrar.
  + **Back End:** Parte de la aplicación web que no se ve y que realiza las acciones en el servidor.
  + **Conexión a BD:** En este caso se representa con un icono de una base de datos.
  + **Apis:**Parte de la aplicación que permite conectarse a otras aplicaciones y que otras aplicaciones se conecten con nuestro sistema.



Hasta aquí hemos conocido más sobre arquitecturas de aplicaciones web, te invitamos a que apliques tus habilidades de búsqueda para profundizar más del tema.