

Modulo 1

Introducción a Desarrollo Web y Aplicaciones

Bienvenida al Módulo I

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.

¿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.

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:

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 Github, 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.

OPERADOR	EJEMPLO	QUÉ HACE
()	("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	Te busca los resultados dentro de una web que hayas especificado.
INFO	info: www.argentina.gob.ar/	Te muestra resultados donde se ofrezca información sobre una página web.
RELATED	related: 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.

¿Cómo la levantas? 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.

Para poder utilizar el traductor debes activar el plugin del navegador de Chrome (<https://chrome.google.com/webstore/detail/google-translate/aapbdbdomjkkjkaonfhkkikfgjllcleb>) o Firefox (<https://addons.mozilla.org/es/firefox/addon/to-google-translate/>) También dando click derecho en la página en la que te encuentres y luego "Traducir a español".

Otra opción recomendable es DeepL que te permite traducir texto con mayor calidad, pero no puedes hacerlo con una página completa. <https://www.deepl.com/translator>

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. En 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.

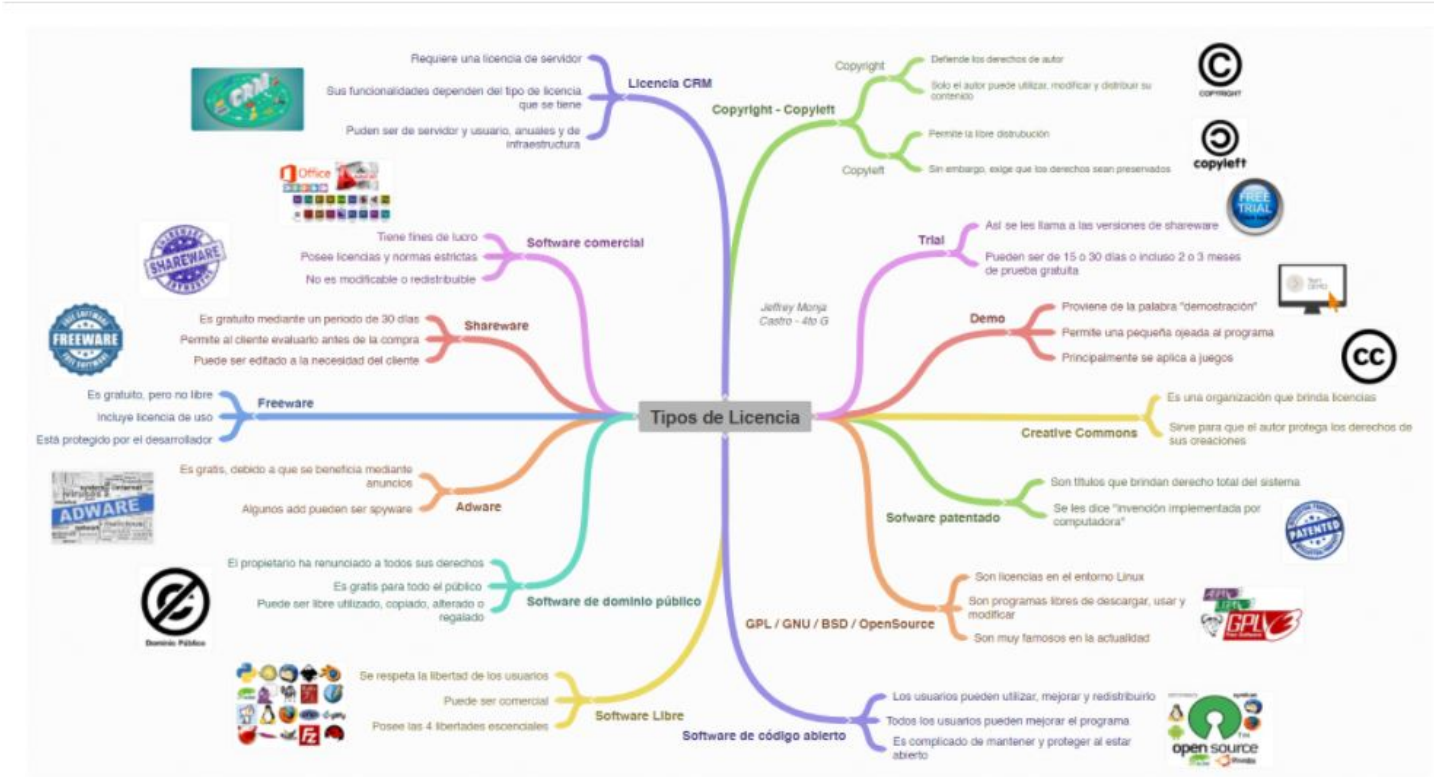
Suele ocurrir con el software libre que, al ser un conjunto de aportes de distintos desarrolladores, cada “parte” de un software tenga licencias distintas. Así ocurre, por ejemplo, con el sistema operativo móvil Android y la iniciativa de código abierto que lo construye y mantiene, el Android Open Source 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.





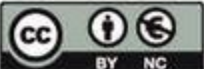

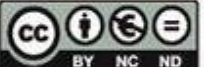


En el caso de las licencias celebradas por otros medios válidos de expresión del consentimiento, cuya aceptación de los términos de la licencia es tácita, se distinguen dos casos:

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.

LAS COMBINACIONES, DE UN VISTAZO

	SI VEO ESTA LICENCIA...	YO PUEDO ...	SI...
MÁS PERMISIVA	 PUBLIC DOMAIN	Domínio Público (CC0): Europeana, Figshare, Open Goldberg V.	<ul style="list-style-type: none">• Compartir• Copiar• Remezclar• Ganar dinero <ul style="list-style-type: none">• Menciono al autor (en algunas jurisdicciones)
	 CC BY	Reconocimiento (by): PLOS, Saylor.org.	<ul style="list-style-type: none">• Compartir• Remezclar• Ganar dinero <ul style="list-style-type: none">• Menciono al autor
	 CC BY SA	Reconocimiento – Compartirlgual (by-sa): Wikipedia, Wikimedia, Arduino, P2PU	<ul style="list-style-type: none">• Compartir• Remezclar• Ganar dinero <ul style="list-style-type: none">• Menciono al autor• Mantengo la misma licencia (by-sa)
	 CC BY ND	Reconocimiento – SinObraDerivada (by-nd): Drupal, Behance, GNU, Free Software Foundation.	<ul style="list-style-type: none">• Compartir• Ganar dinero <ul style="list-style-type: none">• Menciono al autor• No hago remezclas
	 CC BY NC	Reconocimiento – NoComercial (by-nc): Brooklyn Museum, Wired.com Photography	<ul style="list-style-type: none">• Compartir• Remezclar <ul style="list-style-type: none">• Menciono al autor• No gano dinero
	 CC BY NC SA	Reconocimiento – NoComercial – Compartirlgual (by-nc-sa): MIT Open CourseWare	<ul style="list-style-type: none">• Compartir• Remezclar <ul style="list-style-type: none">• Menciono al autor• No gano dinero• Mantengo la misma licencia (by-nc-sa)
	 CC BY NC ND	Reconocimiento – NoComercial – SinObraDerivada (by-nc-nd): Videos TED Talks, Propublica	<ul style="list-style-type: none">• Compartir <ul style="list-style-type: none">• Menciono al autor• No hago remezclas• No gano dinero• Mantengo la misma licencia (by-nc-nd)
MÁS RESTRICTIVA			

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 trabajarás 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/javase/11/docs/tutorial/index.html)
- 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/>

FreelImages: <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/#Cocoaalce>

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/>

JavaScript-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:

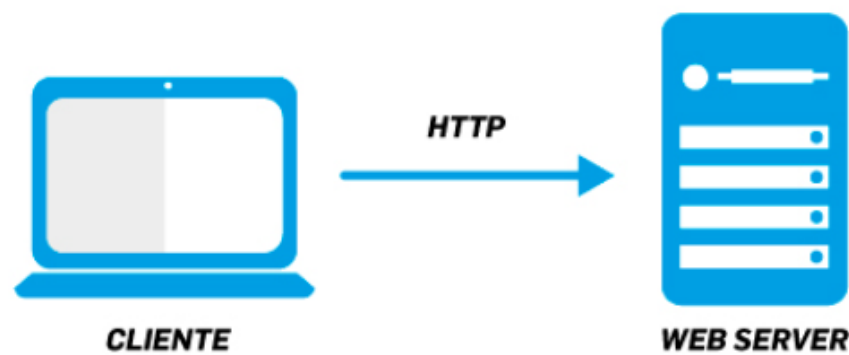
ionic (angular react vim)

react native + expo, flutter

ARQUITECTURA WEB

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 abandona el CERN y funda la W3C, en inglés, "World Wide Web Consortium", organismo internacional que propone recomendaciones y estándares web que aseguran el crecimiento de la World Wide Web.

TECNOLOGÍAS

HISTORIA



HTML | URL
HTTP



HISTORIA 2



ARQUITECTURAS 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 implemente, pero es importante mencionar que en tecnología la mayoría de estructuras está 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.

VARIANTES DE
ARQUITECTURA



PATRÓN DE
CAPAS



ARQUITECTURA



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 otro Servidor. Esto da lugar a un concepto muy importante en POO el DESACOPLOAMIENTO, en este caso el diseño del Front End, Back End y Base de Datos puede desacoplarse. Pero eso lo veremos más adelante...

**ARQUITECTURA
DISTRIBUIDA**



**ARQUITECTURA
MICROSERVICIO
API REST**



ELEMENTOS DE LA ARQUITECTURA WEB

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

ELEMENTOS:



**La infraestructura
de red**

Servidor DNS



Isp

Hosting



Cliente Web

Servidor Web



**Nombre de
Dominio**

**Contenedor de
aplicaciones Web
(o servidor de
aplicaciones web)**



URL



Sitio web

**Servidor de
Bases 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 Interconexión de Sistemas Abiertos (en inglés Open Systems Interconnection). Es un modelo de comunicación de 7 capas, y es la base por el cual viaja toda la información por la redes (internet global, internet local, internet celular, etc.)

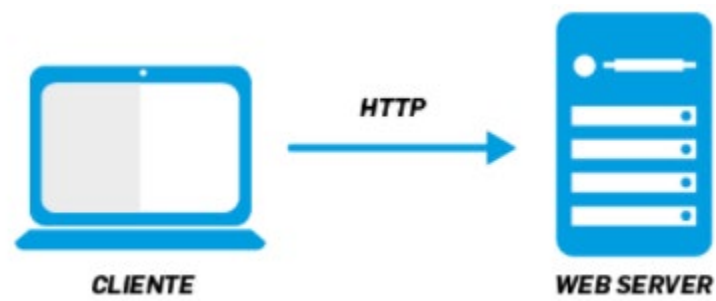
Breve reseña

A partir del desarrollo de ARPANET en 1969 empieza un crecimiento vertiginoso del uso de la internet. En 1990 [Tim 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](#), [intranet](#) o [extranet](#)) 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.



Arquitectura cliente / servidor básico

En 1994 (1 de octubre) Tim Berners-Lee abandona el CERN y funda la [W3C](#), 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](#) 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 + base de datos en un mismo servidor

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 y de datos separados

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



Servidor web + servidor de aplicaciones + base de datos en un mismo servidor

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.

ARQUITECTURA

Pag Web



1 Página

HTML, CSS
JavaScript

1

Sitio Web



+ de 1 Página

HTML, CSS
JavaScript

2

App Web



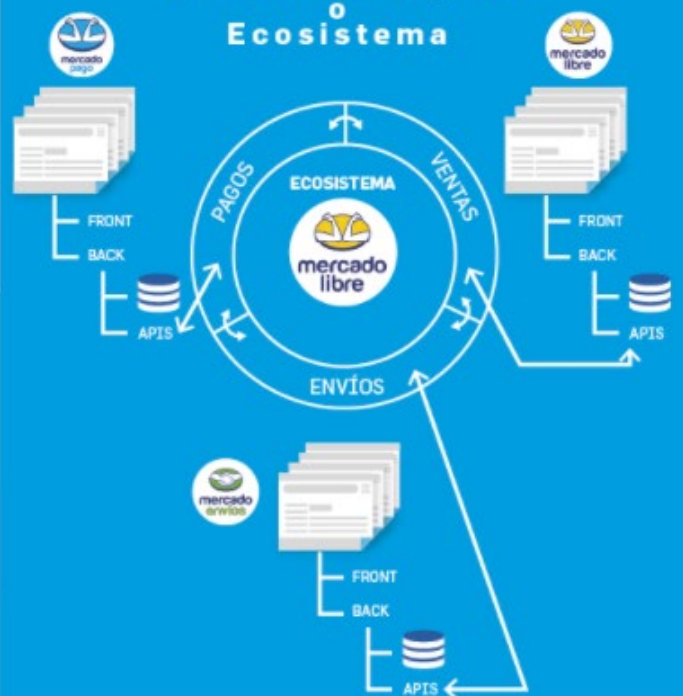
1 o + Pag Dinamicas

Con posibilidad de calculos,
Procesos, Acceso a datos, Etc.
Previamente Diseñada y
Programada.

HTML, CSS, JavaScript
+ Leng. Prog. Front End
+ Leng. Prog. Back End
+ Conexión a DB
+ Apis

3

Plataforma Digital o Ecosistema



4

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 conectarnos 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, notarás 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.

Arquitectura Web distribuida

Para comenzar tengamos presente que hemos visto que una aplicación web tiene varios conceptos como arquitectura cliente servidor, Front End, Back End, Base de datos y Apis. Pero qué pasa cuando una aplicación sabemos que va a recibir muchas consultas de nuestra cliente.

Para poder dar respuesta a ello y antes de entender qué es una arquitectura distribuida hagamos el ejercicio de organizar cada concepto en su lugar y veamos cómo podemos distribuir nuestra aplicación.

Supongamos que nos juntamos varias compañeras del curso y tenemos un sistema web o aplicación web con la arquitectura Cliente / Servidor, en ella debemos organizar los conceptos que ya vimos como Front End, Back End, Base de datos y Api. Esta organización quedaría de esta manera:

Arquitectura Cliente / Servidor:

- **Cliente:**
 - Front End: El código o programación que muestra información al navegador web llamado cliente.
- **Servidor:**
 - Back End: El código o programación que ejecutas las acciones en el servidor y conecta con la base de datos.
 - Base de dato o DB: Donde se almacena la información.

- Api: es quien nos permite conectar a otros sistemas.

¿Qué es una arquitectura centralizada?

Es cuando tenemos todos los elementos de nuestra aplicación web de arquitectura de Cliente / Servidor en un solo lugar equipo o servidor, es decir, tener el Back End, Front End, Bases de datos y APIs en el mismo equipo. Esto hace que en el caso de una falla del equipo toda nuestra aplicación también fallará.

¿Qué es la arquitectura distribuida?

Es tener la posibilidad y capacidad de separar nuestro sistema en distintos servidores de la red (sea red local o internet). Ya sabemos que cuando hablamos de arquitecturas estamos refiriéndonos a una estrategia de cómo construir nuestro sistema dependiendo de lo grande que sea, de las funcionalidades que tenga, esto es mas bien una forma de pensar en cómo escalar nuestro sistema para que soporte más usuarios o más transacciones.

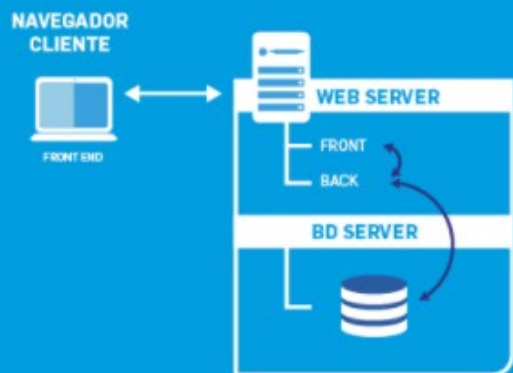
¿Pero cómo te das cuenta cuando una aplicación es distribuida?

En este punto depende de que estés mirando el proyecto o la aplicación, te compartimos solo 2 enfoques:

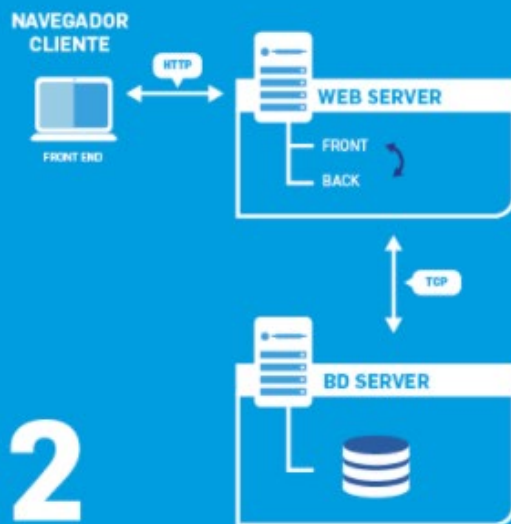
- **Como usuario:** No te darías cuenta porque si el sistema está distribuido funciona como un conjunto único y sincronizado.
- **Como Programador:** Cuando te asignen a un proyecto o cliente en base a preguntas concretas podrás ir conociendo cómo se implementó o distribuyó el sistema, pero te compartimos algunas formas en las que puedes entender que estás frente a una arquitectura distribuida:
 1. **Por la Líder del Proyecto:** Cuando se comienza a trabajar en un proyecto generalmente la líder del mismo hace una explicación del tipo de aplicación con la que se está trabajando, además de indicarnos en qué parte del proyecto estaremos trabajando.
 2. **Por el perfil asignado:** Cuando nos asignan el trabajo en una empresa nos especifican si trabajaremos en el Front End, en el Back End o en ambos Full Stack, de esa manera podemos inferir que la arquitectura es distribuida, igualmente siempre es mejor preguntar para estar seguros.
 3. **Por un diagrama:** Generalmente se utilizan diagramas de aplicación para documentar un sistema, en el se puede ver la separación del sistema y si esta distribuido en 1 o varios servidores.

En los siguientes diagramas o esquemas de una aplicación que fue creada y pensada en forma distribuida separando el código en Front End , Back End. Podemos ver cómo se puede ir escalando o distribuyendo en distintos servidores y que en cualquier caso seguirá funcionando.

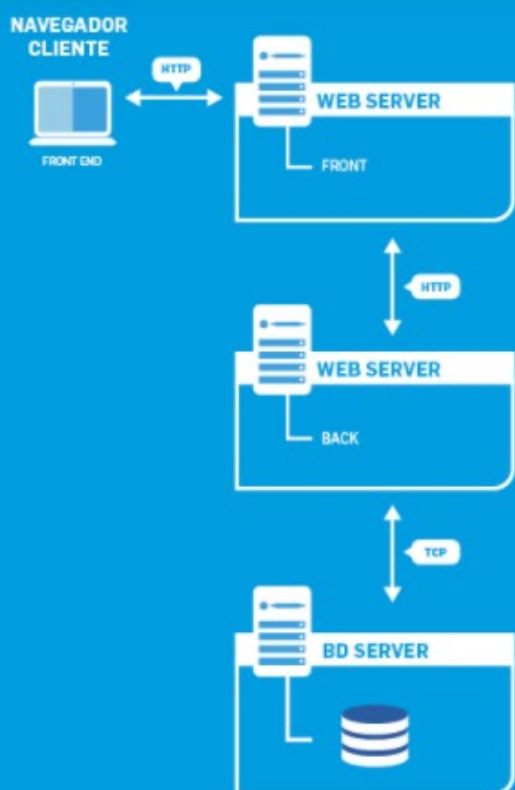
ARQUITECTURA DISTRIBUIDA



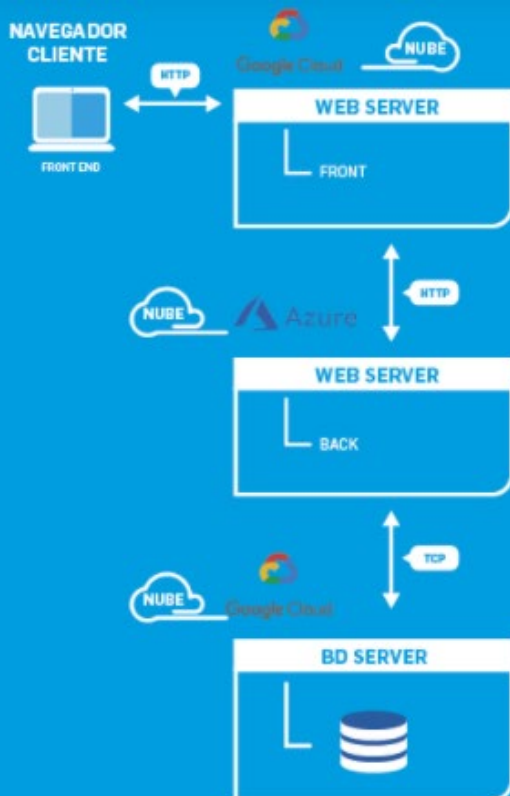
1



2



3



4

Analicemos juntos los escenarios de la imagen anterior:

- **Escenario 1:** En este caso podemos ver que la aplicación está separada en Front End, Back End y bases de datos están en el mismo servidor. Notemos que la aplicación fue diseñada de forma modular o separada (para poder distribuirla) todas las partes del sistema están en un mismo servidor, es decir, en caso de falla del servidor afecta a todo el sistema.
- **Escenario 2:** En este caso podemos ver que se ha separado la base de datos y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida. Pero la parte del Front End y Back End aun están en un mismo servidor.
- **Escenario 3:** En este caso podemos ver que cada parte del sistema Front End, Back End y Base de datos esta en un servidor diferente. Con esto comenzamos a ver los beneficios del diseño con arquitectura distribuida en los sistemas.

- **Escenario 4:** En este caso podemos ver que cada parte del sistema está en la nube de distintas empresas y nuestro sistema sigue funcionando por su diseño modular o distribuido.

¿Hasta dónde puedo modularizar o distribuir mi sistema?

El cómo distribuir el sistema es algo que se analiza en el diseño de la aplicación o se va cambiando a medida que el sistema va creciendo, normalmente cuando llegamos a un trabajo las aplicaciones ya están funcionando y necesitan de nuestro conocimiento para mantenerlas y agregarles mejoras.

Cuando un aplicación se hace más grande, compleja y con más usuarios necesitamos seguir modularizando el sistema, dado que no nos alcanza con separar en Front End, Back End y Base de datos. En esta situación ya debemos pensar en modularizar o separar algunas funcionalidades del sistema, algunos motivos pueden ser:

- **Por alta demanda:** Cuando el sistema tiene una funcionalidad que es compleja, consume mucho recurso del servidor o es muy demandada por distintas partes del sistema.
- **Por interconexión:** Cuando un sistema tiene funcionalidades que se necesita dar acceso a otros sistemas para consumir ese proceso, función o datos.
- **Por segregación de roles:** Cuando es necesario separar roles o funciones por motivos de seguridad o aspectos técnicos, también puede ser porque negocio lo requiere, por ejemplo si se decide por seguridad separar el proceso de autenticación del sistema para reforzar la seguridad.
- **Por escalamiento:** Cuando las proyecciones indican que en un periodo de tiempo la demanda aumentará considerablemente, será necesario agregar más servidores en la red con la misma funcionalidad para que satisfaga la demanda.
-

Existen varias **formas de separar estas funcionalidades que llamaremos API REST o Microservicios**, si bien a medida que avancemos iremos aprendiendo más sobre las API REST, ahora veremos cómo la arquitectura distribuida puede aplicarse para pasar de un gran sistema que tiene todo en un solo lugar a separarlo en pequeñas y que todo siga funcionando.

Para ejemplificar tomaremos un sistema que tiene lo siguiente:

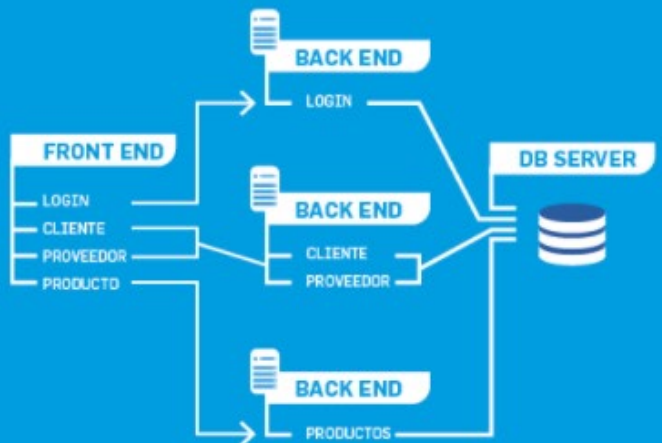
- **Front End:** Todos fue diseñado pensando en una arquitectura distribuida, las funcionalidades son las siguientes.
 - Login: Se conecta con el Back End para validar el usuario y clave.
 - Cliente: Se conecta con el Back End para consultar, editar, crear y eliminar clientes
 - Producto: Se conecta con el Back End para consultar, editar, crear y eliminar productos
 - Proveedor: Se conecta con el Back End para consultar, editar, crear y eliminar proveedores
 -
 -
-
- **Back End:** Todos fue diseñado pensando en una arquitectura distribuida, las funcionalidades son las siguientes,
 -
 - Login: Recibe la petición, consulta la base de datos y valida si el usuario existe.
 - Cliente: Recibe la petición, consulta la base de datos y devuelve los datos de cliente.
 - Producto: Recibe la petición, consulta la base de datos y devuelve los datos del producto.
 - Proveedor: Recibe la petición, consulta la base de datos y devuelve los datos del proveedor.

Ahora veamos en un diagrama o esquema cómo estas funcionalidades se pueden ir distribuyendo en distintos servidores:

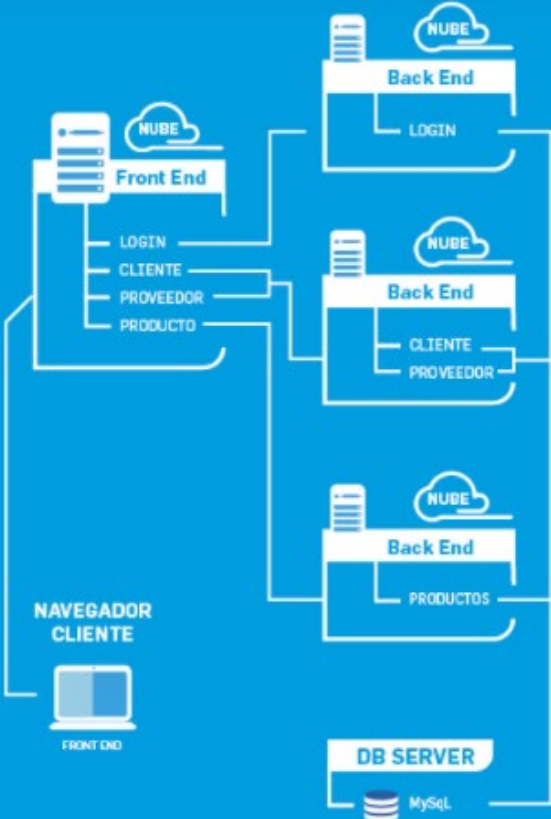
ARQUITECTURA MICRO SERVICIO API REST



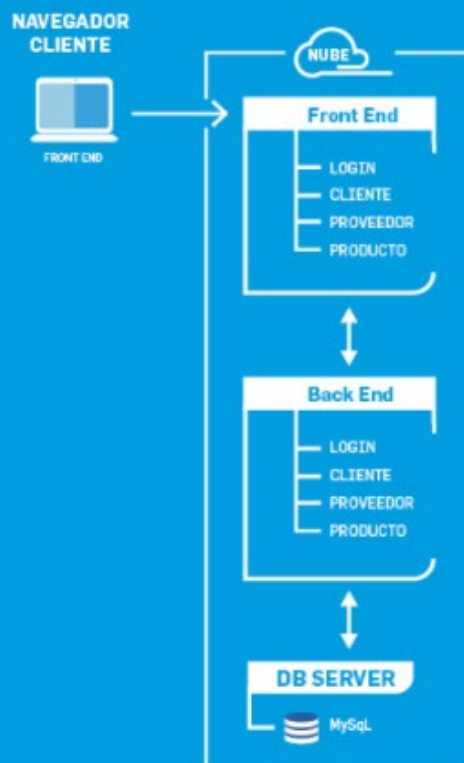
1



2



3



4

Analicemos juntos los escenarios de la imagen anterior:

- **Escenario 1:** En esta caso tenemos el sistema que está en 3 servidores para cada separación
 1. Front End: Todas las funcionalidades están juntas y en el mismo servidor.
 2. Back End: Todas las funcionalidades están juntas y en el mismo servidor.
 3. Base de datos: Todos los datos en una base de datos y en el mismo servidor.
- **Escenario 2:** En este caso podemos ver que cada parte del sistema está distribuido en 5 servidores y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida.
 1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor.
 2. Back End: La funcionalidad de *login está en un servidor exclusivo*.

3. Back End Las funcionalidades de *cliente y proveedor están en un servidor*.
4. Back End: La funcionalidad *producto está en un servidor* exclusivo.
5. Base de datos: Todos los datos en una base de datos y en el mismo servidor.

- **Escenario 3:** En este caso **podemos ver que cada parte del sistema está distribuido en 5 servidores en nubes diferentes** y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida.
 1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor de la nube de Argentina.
 2. Back End: La funcionalidad de *login está en un servidor de la nube* de España.
 3. Back End Las funcionalidades de *cliente y proveedor están en un servidor* de la nube de Canadá.
 4. Back End: La funcionalidad *producto está en un servidor en la nube* de Nueva Zelanda.
 5. Base de datos: Todos los datos en una base de datos y en un servidor propio.
- **Escenario 4:** En este caso **podemos ver que el sistema está en la misma nube**, a pesar que las funcionalidades están todas juntas como el en escenario 1 y nuestro sistema sigue funcionando.
 1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor de la nube de Argentina.
 2. Back End: Todas las *funcionalidades están juntas* y en la nube de Argentina.
 3. Base de datos: Todos los datos en una base de datos y en la nube de Argentina.

Como hemos visto hay varios niveles de como modularizar, distribuir o separar en capas una aplicación con sus funcionalidades, en algunos casos la separación solo puede ser Font End, Back End. Pero existen otras formas de separar en partes más pequeñas la aplicación y eso lo hacemos con la ayuda de las APIs REST.

Elementos de la arquitectura Web

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

1- La infraestructura de red

Si bien es cierto que, en fase de desarrollo, para probar nuestra aplicación web, no necesitaríamos esta infraestructura, una vez nuestra aplicación se instale en el hosting definitivo, será necesario una red ethernet y todos los componentes que hacen posible la conectividad de los equipos informáticos. Con esto nos referimos a cables de red, sea [utp](#) o fibra óptica, placas de red, sea wifi o cableada, [switch](#), [routers](#), [modem](#), etc. Y estos componentes físicos son necesarios tanto del lado del cliente como del servidor. Con esto se quiere decir que no es posible implementar una aplicación web sino existe una infraestructura de red preexistente o se diseña e implementa una nueva. Esta puede ser: internet, intranet o extranet.

2.- Isp

Es el proveedor del servicio de internet.

3.- Cliente Web

Es el navegador web. Ejemplos: Chrome, Safari, Firefox, etc. Pero ya no se restringe solo a estos dispositivos sino que podría ser, por ejemplo, un sistema embebido ejecutándose en una [SBC](#) (small board computer). Incluso podría no estar ejecutando un navegador convencional, como por ejemplo un reloj inteligente, o un dispositivo vinculado a una máquina de producción seriada como los surgidos de la mano del concepto de [Industria 4.0](#).

4.- Nombre de Dominio

Dicho de forma sencilla, el nombre de dominio (o simplemente "dominio") es el nombre de un sitio web. Es decir, es lo que aparece después de "@" en una dirección de correo electrónico o después de "www." en una dirección web. Si alguien te pregunta cómo te puede encontrar en Internet, tendrás que decirle tu nombre de dominio. Las computadoras para comunicarse utilizan direcciones [IP](#) (números únicos en la red). Un ejemplo de una dirección IP de un servidor es 173.194.121.32. Para nosotros es imposible recordar tantos números y saber qué servicio o qué aplicación se encuentra en esa dirección IP o servidor. Para resolver estos problemas se usan palabras que las personas pueden leer, que son intuitivas, fáciles de recordar y dicen mucho sobre el servicio web que ofrecen, se denominan nombres de dominio. ¿Puedo comprar un nombre de dominio? No, los nombres no se pueden comprar, solo se puede pagar por el derecho a usarlo por cierto periodo de tiempo. Para registrar un dominio a tu nombre debes hacerlo por medio de una empresa que se encarga de administrar las registraciones de nombres de dominio. En el caso de Argentina es <https://nic.ar>

Aquí tienes algunos ejemplos de nombres de dominio: google.com, wikipedia.org, youtube.com https://domains.google/intl/es_es/learn/web-terms-101/

5.- URL

Una URL (o localizador uniforme de recursos) es una dirección web completa que se utiliza para encontrar una página web específica. Mientras que el dominio es el nombre del sitio web, la URL es una dirección que remite a los usuarios a una de las páginas de ese sitio web. Cada URL contiene un nombre de dominio y otros componentes necesarios para localizar una página o un contenido concreto. Aquí tienes algunos ejemplos de URL: <http://www.google.com> - <https://es.wikipedia.org/wiki/umami> - <https://www.youtube.com/feed/trending>; [https:// dominios. google / intl / es es / learn / web-terms-101 /](https://dominios.google/intl/es_es/learn/web-terms-101/)

6.- Sitio web

Aunque una cosa lleve a la otra, comprar un nombre de dominio no implica tener un sitio web. El dominio es el nombre del sitio web, la URL es la forma de encontrarlo y el sitio web es lo que los usuarios ven en su pantalla y con lo que interactúan. Es decir, cuando compres un dominio, habrás adquirido el nombre de tu sitio web, pero te faltará crear el sitio web en cuestión. https://domains.google/intl/es_es/learn/web-terms-101/

7.- Servidor DNS

Sistema de Nombre de Dominio. Se ocupa de la administración del espacio de nombres de dominio. Este servidor se encarga de hacer las conversiones de nombres de dominio a direcciones IP. Cuando el cliente realiza una petición web, por ejemplo google.com, una de las primeras acciones del sistema es invocar un servidor DNS para que devuelva la dirección IP del / o de alguno de los servidores de google. Por ejemplo devolverá la ip 172.217.162.14.

8.- Hosting

Es el nombre que se le da al servicios de alojamiento en la web a nuestras paginas, aplicaciones, bases de datos (los hosting son servidores que están siempre encendidos y conectados a internet). Los programadores una vez terminado el trabajo suben su aplicación web al Hosting para que todo el mundo pueda acceder.

9.- Servidor Web

También llamado servidor HTTP, es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

10.- Contenedor de aplicaciones Web (o servidor de aplicaciones web) ,

En el módulo que permite la ejecución de aplicaciones web. Por ejemplo el módulo PHP o Python del Servidor Web. Componente ASP o ASPX de IIS. Servidor o Contenedor de Aplicaciones Web Java: Tomcat, Weblogic, Websphere, JBoss, Geronimo, etc.

11.- Servidor de Bases de Datos

Estos son contenedores de bases de datos que permiten organizar y administrar los datos que deben permanecer en un medio de almacenamiento permanente. Resuelven problemas de: seguridad, mecanismos de comunicación, concurrencia, inconsistencias de los datos, respaldo, entre otros. Hay varios tipos de bases de datos, por ejemplo, las relaciones que organizan los datos en forma de tablas, en filas y columnas. Otro tipo son los orientados a objetos u orientados a documentos donde el concepto de tablas se cambia por la colección con formatos similares a "json". JavaScript Object Notation ([JSON](#)) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. Es utilizado para transmitir datos en aplicaciones web (por ejemplo: enviar algunos datos desde el servidor al cliente, así estos datos pueden ser mostrados en páginas web, o viceversa).

Full Stack – Infografía



DESARROLLO WEB FULL STACK

El desarrollador web full stack puede crear aplicaciones web dinámicas. Esto se logra en base a los estándares web y utilizando tecnologías web que pueden variar dependiendo del stack de desarrollo. Tiene por objetivo la creación de aplicaciones web dinámicas.

Como vimos anteriormente, las aplicaciones web dinámicas modifican su contenido en función del usuario que accede permitiendo mostrar uno u otro contenido dependiendo del usuario y de su interacción para con la aplicación web (ej. facebook, instagram, etc.) mientras que, las aplicaciones web estáticas muestran siempre el mismo contenido independientemente del usuario visitante (ej. un blog).



STACK

Lo que denominamos stack tecnológico, o también denominado stack de soluciones o ecosistema de datos, es un conjunto de todas las herramientas tecnológicas utilizadas para construir y ejecutar una sola aplicación, en este caso web.

Sitios como la red social Facebook, han sido desarrolladas por una combinación de frameworks de codificación y lenguajes, entre los que se incluyen JavaScript, HTML, CSS, PHP y ReactJS. Así podemos decir que este es el "stack tecnológico" de Facebook.

Otros Stacks

Hay otros stacks, y probablemente surjan nuevos en el futuro cercano. Una solución completa involucra no solo el stack, que en ocasiones conlleva la inversión en las correspondientes licencias, sino también las consideraciones referidas al costo de hardware, entre ellos el del servidor físico. Así por ejemplo, en torno al lenguaje de programación Java y de la mano del servidor web JBoss, se pueden implementar en servidores con sistemas operativos RedHat soluciones Web. El hardware puede ser por ejemplo de Dell, HP, entre otros. Otra posible configuración es el frontend desarrollado en torno a React, el backend en Python y Django y usando como base de datos SQLite.

¿Desarrollo Web full stack?

El desarrollador web full stack puede crear aplicaciones web dinámicas. Esto se logra en base a los [estándares web](#) y utilizando tecnologías web que pueden variar según la pila de desarrollo. Tiene por objetivo la creación de aplicaciones web dinámicas. Como vimos anteriormente, las aplicaciones web dinámicas modifican su contenido en función del usuario que acceden permitiendo mostrar uno u otro contenido dependiendo del usuario y de su interacción para con la aplicación web (ej. Facebook, Instagram, etc.) mientras que las

¿Qué son los estándares web?

¿cuáles existen?

MÁS USADOS

LAMP

MEAN

OTROS

.NET

BFF

Django

Ruby on Rails

LEMP

MEER

aplicaciones web estáticas muestra siempre el mismo contenido independientemente del usuario visitante (ej. un blog).

Pero ¿qué son los estándares web y qué buscadores existen? Los estándares web son tecnologías que se utilizan para crear aplicaciones web. Los mismos son creados por organismos de estándares - instituciones que invitan a grupos de personas de diferentes compañías de tecnología a unirse y acordar cómo deben funcionar las tecnologías de la mejor manera posible para cumplir con todos sus casos de uso. El [W3C](#) es el organismo de estándares web más conocido, pero hay otros como [WHATWG](#) (responsables de la modernización del lenguaje HTML), [ECMA](#) (publica el estándar para ECMAScript, en el que se basa JavaScript), [Khronos](#) (publica tecnologías para gráficos 3D, como WebGL) y otras [MDN Web Docs](#).

¿ Pero qué son los Stacks?

Lo que denominamos ***stack tecnológico***, o también denominado ***stack de soluciones*** o ecosistema de ***datos***, es un conjunto de todas las herramientas tecnológicas utilizadas para construir y ejecutar una sola aplicación.

Recordemos lo que ya hemos visto en la arquitectura Cliente /Servidor a su vez al código que navega el cliente por el navegador le llamamos Front End y la parte de código que accede a los datos se le llama Back End , si no recuerdas has clic [aquí](#),

Cada una de estas partes de la arquitectura puede crearse con distintos lenguajes de programación, el que quieras, por citar algunos ejemplo:

Front End: Para el desarrollo del Front End se necesita.

- Estructura y estilos: HTML, CSS o framework como Bootstrap
- Lenguaje programación: JavaScript, java, PHP o bien pueden ser framework como Angular

Back End: Para el desarrollo de Back End se necesita un web server, una base de datos y un lenguaje de programación.

- Base de datos: puede ser MySQL, PostgreSQL, etc
- Lenguaje: Java, PHP, etc
- Web server: Apache, NGINX, etc

Como son muchas las opciones de lenguaje de programación, bases de datos y frameworks. Se ha creado el concepto de Stacks o pila que agrupa las tecnologías (que funcionan bien en conjunto o mas utilizadas) para crear una aplicación desde el Front End y Back End usando ese Stack y ahorrarnos tener que investigar al detalle que tecnología podemos utilizar, es como un combo de tecnologías. Esto también ayuda a las empresas a identificar el conocimiento que tiene un programador al nombrar el Stack tecnológico que utiliza.

Sitios como la red social Facebook, han sido desarrolladas por una combinación de frameworks de codificación y lenguajes, entre los que se incluyen JavaScript, HTML, CSS, PHP y ReactJS. Así podemos decir que este es el "stack tecnológico" de Facebook.

Podríamos preguntarnos cuáles son los stacks que usan Netflix, Whatsapp, Instagram...

Uno de los stacks o pila de tecnologías más utilizado por los desarrolladores es el que se conoce por **LAMP** : **L**inux, **A**pache, **M**ySQL y **P**HP. Cualquier web hecha con Wordpress, Drupal o Prestashop, por ejemplo, están hechas sobre estos cuatro pilares. Pero se pueden hacer las variaciones que se crean convenientes, puesto que muchas de estas tecnologías son intercambiables por otras similares. Por ejemplo, NginX en lugar de Apache, PostgreSQL en lugar de MySQL o Ruby on Rails en lugar de PHP.

Otro stack muy utilizado es el llamado **MEAN** , que se compone de **M**ongoDB, **E**xpress, **A**ngular y **N**odeJS. te dejamos un imagen ilustrativa del Stack MEAN.



Algunos ejemplos de Stacks

- **LAMP:** Linux - Apache - MySQL - PHP. Es la más antigua y el formato de aplicaciones predominante se denomina "**Multi Page Application o MPA**" o aplicación de múltiples páginas.
- **MEAN:** MongoDB - Express - AngularJS - Node.js. Es la más moderna y el formato de aplicaciones predominante se denomina "**Aplicación** de una sola página o **SPA**" o aplicación de una sola página.
- **.NET:** .NET + WebApi + IIS - SQL Server. Utilizan Microsoft . [Núcleo neto](#) como plataforma de desarrollo.
- **Patrón BFF:** Se puede usar una arquitectura Backend for Frontend (BFF) para crear backends para aplicaciones web o móviles orientados al cliente. Los BFF pueden ayudar a respaldar una aplicación con múltiples clientes al mismo tiempo que mueven el sistema a un estado menos acoplado que un sistema monolítico. El [acoplamiento](#) es "es el grado en que los módulos de un programa **depende** unos de otros" . Una aplicación monolítica es una unidad cohesiva de código. Este patrón de código ayuda a los equipos de desarrollo a iterar las funciones más rápido y tener control sobre los backends para aplicaciones móviles sin afectar la experiencia de la aplicación móvil o web correspondiente.
- **Django:** Python - Django - MySQL
- **Ruby on Rails:** Ruby - SQLite - Rails
- **LEMP:** Linux - Nginx - MySQL - PHP
- **MERN:** MongoDB - Express - React - Node.js.
- **Otros Stacks:** Existen otros stacks y probablemente surjan nuevos en el futuro cercano. Una solución completa involucra no solo el stack, que en ocasiones conlleva la inversión en las correspondientes licencias, sino también las consideraciones referidas al costo de hardware, entre ellos el del servidor físico. Así por ejemplo, en torno al lenguaje de programación Java y de la mano del servidor web JBoss, se pueden implementar en servidores con sistemas operativos RedHat soluciones Web. El hardware puede ser por ejemplo de Dell, HP, entre otros. Otra posible configuración es el frontend desarrollado en torno a React, el backend en Python y Django y usando como base de datos SQLite.

¿Todas las empresas usan si o si estos Stacks?

No siempre, tengamos en cuenta que los stacks son un grupo de herramientas tecnológica sugeridas para crear una aplicación web, por ende, las empresas pueden elegir las herramientas que necesiten sin tener en cuenta los Stack antes nombrados, Pero recuerda cuando una empresa solicita un programador LAMP y te presentan ellos asumirán que sabrás manejar ese grupo de tecnologías. Existen otras empresas que buscan perfiles con conocimiento solo en Front End o en Back End, estas empresas focalizan al programador en un solo área independientemente del stack.

Como has notado las empresas utilizan distintas tecnologías

¿Con qué tecnología trabajaremos en el curso?

Luego de investigar y consultar la demanda requerida de las empresas vs las tecnologías mas utilizadas en Argentina, utilizaremos las siguientes tecnologías:

Para Front End:

- Estructura y estilos: HTML, CSS y framework como Bootstrap
- Lenguaje programación: TypeScript y el framework como Angular ambos basados en Javascript

Para Back End:

- Base de datos: Usaremos MySQL
- Lenguaje: Java con Framework Spring Boot
- Web server: Apache Tomcat

Ahora que sabemos con las tecnologías que vamos a trabajar podemos decir que será ya tenemos nuestro stack tecnológico.

¿Qué significa ser un desarrollador full stack?

El desarrollador full stack es un perfil que tiene conocimiento de lenguajes de programación de Front End, Back End, APIs y Bases de datos. Esta amplitud de conocimiento es muy buscada hoy en día por las empresas, esto significa que al momento de trabajar en un proyecto podrías estar asignado a cualquiera de esas áreas.

Recuerda que tener el conocimiento de un programador full stack te amplía las posibilidades de trabajo, por ello, vale la pena el esfuerzo de aprender y practicar. En el procesos podrás descubrir que tienes facilidad o te gusta mas alguna parte del Full Stack (Front End, Back End o Bases de datos) y posiblemente te conviertas en experto en alguno. Por eso a seguir estudiando y practicando.

Procesos de petición Infografía



PROCESO DE UNA PETICIÓN WEB

- 1** **Cliente Web:** Solicita la resolución de nombres al servidor DNS. Por ejemplo: google.com
- 2** **Servidor DNS:** Recibe y trata la solicitud. Una vez recibida la petición realiza las consultas necesarias para resolver y obtener la dirección IP.
- 3** **Servidor DNS:** Devuelve al navegador Web la dirección IP que corresponde al Servidor Web.

VER GRÁFICO

En resumen, si hay algo que nos queda claro entonces, es el potencial que ha tenido y tiene la web, su capacidad de escalabilidad, por lo que todos los mercados, incluso la cultura y el arte, se manifiestan con gran libertad.

- 4** **Cliente Web:** Conecta con el servidor web mediante la dirección IP y el puerto. Realiza la petición mediante una URL (Método GET) o un formulario (Método POST). Dicha solicitud incluye: la dirección IP del servidor web, el puerto del servidor web, URL y parámetros.
- 5** **Servidor Web:** Control de Acceso, Análisis de la petición y localización del recurso. Como detecta que es el acceso a un fichero o ruta de aplicación tiene que traspasar el control al Contenedor de aplicaciones Web
- 6** Paso de la petición del servidor web al contenedor de aplicaciones web
- 7** El contenedor analiza la petición y en base a la ruta traspasa el control a la aplicación web.
- 8** Paso del control de la petición desde el CAW a la aplicación.
- 9** La aplicación recibe la petición y decide qué hacer en base a ella, es decir, elegir la funcionalidad que se encargará de gestionar esa petición, normalmente en base a la ruta, el método HTTP y los parámetros de entrada por URL. Una vez elegida ejecutará esa funcionalidad.
- 10** La aplicación realiza una petición SQL a la base de datos.
- 11** La Base de Datos recibe la petición SQL y la procesa realizando los cambios que tenga que hacer, si corresponde.
- 12** Una vez procesada la petición devuelve los datos a la aplicación web, normalmente un conjunto de datos. Ej. los 10 últimos clientes.
- 13** La aplicación web recibe estos datos y tiene que generar una salida, normalmente HTML, donde estructura el contenido de los datos devueltos por la BBDD en etiquetas HTML.

Pero a medida que ampliamos los horizontes aparecen nuevos problemas y estos generan nuevas soluciones. Es que ya no son simples páginas web coloridas y dinamizadas, sino sistemas completos, distribuidos, multiplataformas, para usos generales y específicos, como por ejemplo una plataforma de ecommerce.

VER GRÁFICO



-
- 14** La aplicación web devuelve una respuesta al Contenedor de Aplicaciones Web

 - 15** El contenedor procesa la respuesta, para controlar la ejecución de la aplicación por si esta falla.

 - 16** El Contenedor de Aplicaciones Web devuelve el fichero al servidor web.

 - 17** El servidor Web devuelve los datos dentro de la respuesta HTTP al navegador web.

 - 18** **Cliente Web:** Presenta (renderiza) el contenido HTML resultante.

Proceso de una petición Web y modelo OSI

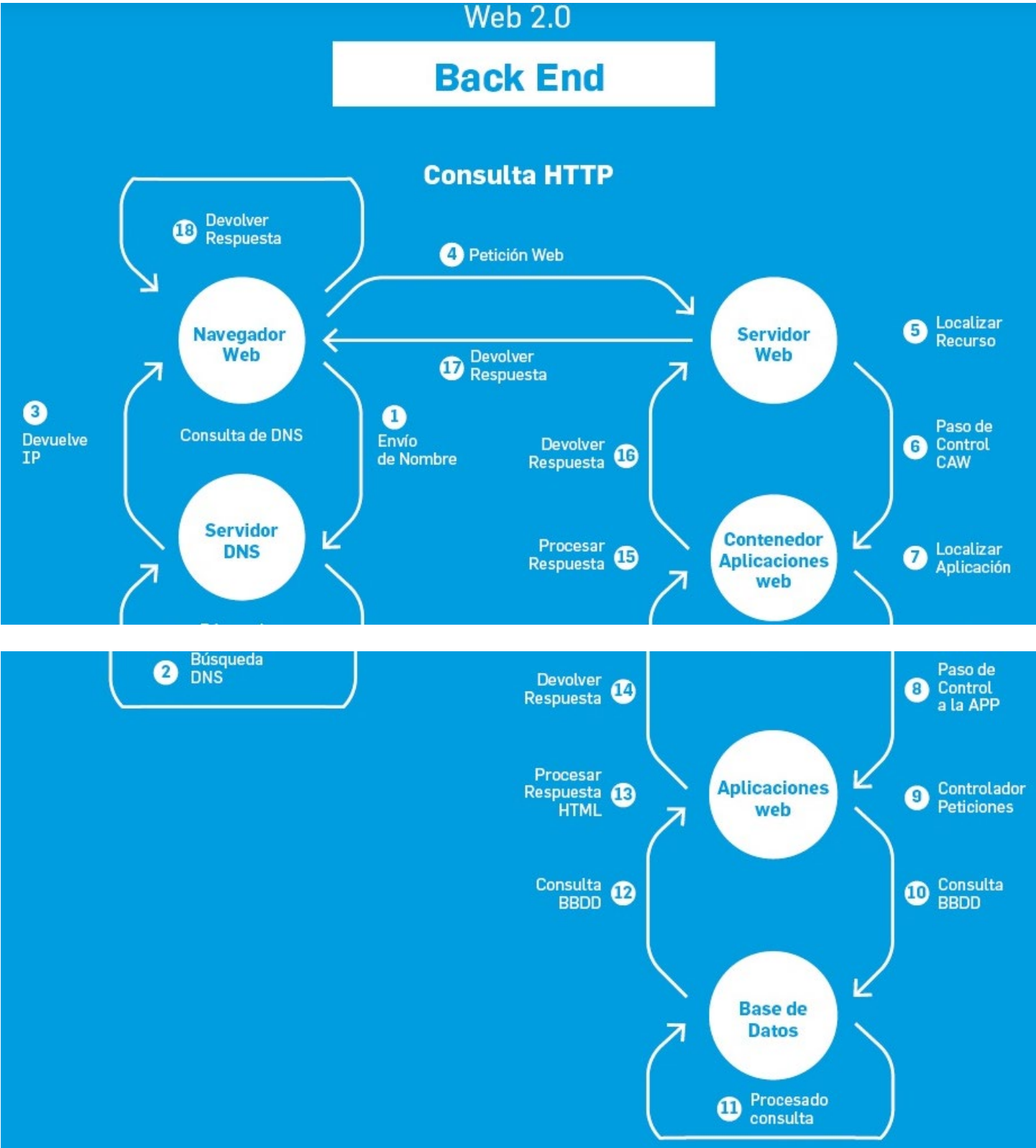
Ahora que entendemos los principios básicos de la arquitectura WEB y algunos de sus elementos, veamos el Modelo OSI.

El Modelo OSI o Modelo de Interconexión de Sistemas Abiertos (en inglés **Open Systems Interconnection**). 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.), te invitamos a profundizar más haciendo clic [aquí](#).

En la siguiente animación te mostramos como viaja la información por internet de una computadora a otra pasando por las 7 capas del modelo OSI.

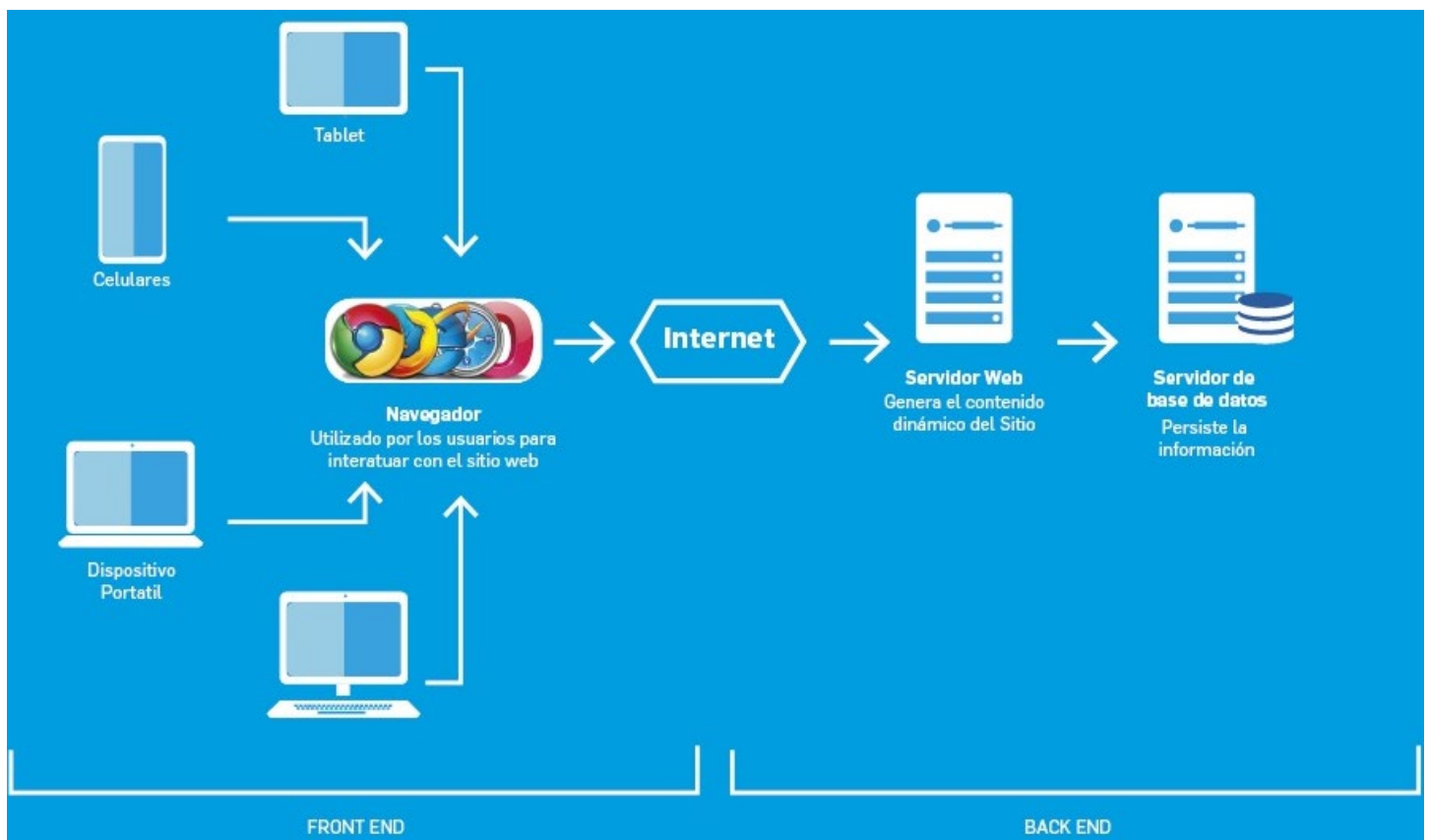
Proceso de una petición web

1. Cliente Web: Solicita la resolución de nombres al servidor DNS. Por ejemplo: google.com
2. Servidor DNS: Recibe y trata la solicitud. Una vez recibida la petición realiza las consultas necesarias para resolver y obtener la dirección IP.
3. Servidor DNS: Devuelve al navegador Web la dirección IP que corresponde al Servidor Web.
4. Cliente Web: Conecta con el servidor web mediante la dirección IP y el puerto. Realiza la petición mediante una URL (Método GET) o un formulario (Método POST). Dicha solicitud incluye: la dirección IP del servidor web, el puerto del servidor web, URL y parámetros.
5. Servidor Web: Control de Acceso, Análisis de la petición y localización del recurso. Como detecta que es el acceso a un fichero o ruta de aplicación tiene que traspasar el control al Contenedor de aplicaciones Web
6. Paso de la petición del servidor web al contenedor de aplicaciones web
7. El contenedor analiza la petición y en base a la ruta traspasa el control a la aplicación web.
8. Paso del control de la petición desde el CAW a la aplicación.
9. La aplicación recibe la petición y decide qué hacer en base a ella, es decir, elegir la función que se encargará de gestionar esa petición, normalmente en base a la ruta, el método HTTP y los parámetros de entrada por URL. Una vez elegida ejecutará esa función.
10. La aplicación realiza una petición SQL a la base de datos.
11. La Base de Datos recibe la petición SQL y la procesa realizando los cambios que tenga que hacer, si corresponde.
12. Una vez procesada la petición devuelve los datos a la aplicación web, normalmente un conjunto de datos. Ej. los 10 últimos clientes.
13. La aplicación web recibe estos datos y tiene que generar una salida, normalmente HTML, donde estructura el contenido de los datos devueltos por la BBDD en etiquetas HTML.
14. La aplicación web devuelve una respuesta al Contenedor de Aplicaciones Web
15. El contenedor procesa la respuesta, para controlar la ejecución de la aplicación por si esta falla.
16. El Contenedor de Aplicaciones Web devuelve el fichero al servidor web.
17. El servidor Web devuelve los datos dentro de la respuesta HTTP al navegador web.
18. Cliente Web: Presenta (renderiza) el contenido HTML resultante.



En resumen, si hay algo que nos queda claro entonces, es el potencial que ha tenido y que tiene la web, su capacidad de escalabilidad, por lo que todos los mercados, incluso la cultura y el arte, se manifiestan con gran libertad.

Pero a medida que ampliamos los horizontes aparecen nuevos problemas y estos generan nuevas soluciones. Es que ya no son simples páginas web coloridas y dinamizadas, sino sistemas completos, distribuidos, multiplataformas, para usos generales y específicos, como por ejemplo una plataforma de e-commerce.



Arquitectura Web de 3 niveles

En cuanto a los desarrolladores, es importante agregar que si bien, lo más habitual es que se especialicen en frontend o en backend, hoy existe un tercer perfil: desarrollador “Full-Stack” (muy solicitado por las empresas de desarrollo de software). Este perfil se caracteriza por tener una visión integral de toda la aplicación web (frontend + backend).

Transferencia de datos Infografía



CONCEPTUALIZACIÓN DE LAS ESTRUCTURAS INVOLUCRADAS EN LA TRANSFERENCIA DE DATOS ENTRE COMPUTADORAS

Para entender la programación web es necesario además conocer o tener alguna noción sobre las estructuras involucradas en la transferencia de datos entre computadoras. Esta transferencia de datos, es decir, la comunicación entre dos o más computadoras se lleva a cabo a través de lo que se llama comúnmente red de computadoras. La más conocida red es Internet o, como también se la denomina, la red de redes.

Vamos a ver conceptos de red de computadoras, de Internet y cómo funcionan los elementos que las constituyen, a modo introductorio para entender cómo funcionan los sistemas para la Web.

Para comenzar vamos a definir que

Una red de computadoras, también llamada red de ordenadores, red de comunicación de datos o red informática, es un conjunto de equipos nodos y software conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos con la finalidad de compartir información recursos y ofrecer servicios (...)” (Tanenbaum, 2003)

Finalidad principal para creación de red de ordenadores:

- * La de compartir recursos e información
- * A grandes distancias
- * Asegurar la confiabilidad y la disponibilidad de la información (características propias que definen lo que es información)
- * Aumentar la velocidad de transmisión de los datos y
- * Reducir los costos. (CENS, 2018)

Si bien este no es un curso de redes de computadora es necesario conocer los elementos físicos en los que trabajan los sistemas que vamos a realizar. Con ello queremos decir que si bien podemos abstraernos de esta tecnología es muy útil tener alguna noción de lo que pasa detrás de escena.

La comunicación por medio de una red se lleva a cabo en dos diferentes categorías: una capa denominada física y otra lógica.

CAPA FÍSICA



CAPA LÓGICA



FORMACIÓN DE UNA RED

Para formar una red se requieren elementos de hardware, software y protocolos. Los elementos físicos se clasifican en dos grupos: los dispositivos de usuario final (llamados también Hosts) y dispositivos de red.

Entre los dispositivos de usuario final podemos enumerar computadoras, impresoras, escáneres, y demás elementos que brindan servicios directamente al usuario. Los segundos (dispositivos de red) son todos aquellos que conectan entre sí a los dispositivos de usuario finales posibilitando su intercomunicación. (Farías, 2013).

Internet es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen, constituyan una red lógica única de alcance mundial. (Guillamón, 2009)

Uno de los servicios que más éxito ha tenido en Internet ha sido **world wide web, WWW o la web**. La WWW es un conjunto de protocolos que permite de forma sencilla la consulta remota de archivos de hipertexto y utiliza Internet como medio de transmisión («Internet, n.». Oxford English Dictionary (Draft edición). Marzo de 2009).

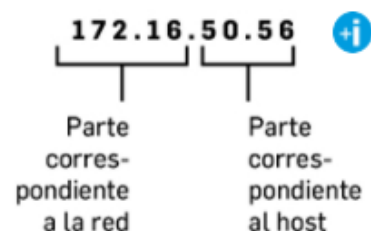
TCP (protocolo de control de transmisión)



VER GRÁFICO



protocolo IP (Internet Protocol)



Para terminar, una dirección IP es un número que identifica de manera lógica y jerárquica una interfaz de un dispositivo dentro de una red que utiliza el protocolo de internet.

VER MÁS



Para hacer más notable esta distinción vamos a nombrar algunos otros servicios y protocolos en Internet aparte de la web por ejemplo [para el envío de correo electrónico usamos el protocolo SMTP](#), para la [transmisión de archivos usamos el protocolo FTP](#), para nombrar algunos.

Esta familia de protocolos son un conjunto de protocolos de red en los que se basa Internet y permite la transmisión de datos entre computadoras como hemos dicho.

Entre los principales podemos nombrar el conjunto de protocolos TCP/IP que hace referencia a los dos protocolos más importantes que componen la Internet, que fueron los primeros en definirse y qué son los más utilizados.

CONCLUSIÓN

Se utiliza la combinación de estos dos protocolos para la comunicación en Internet, en dónde TCP aporta la fiabilidad entre la comunicación e IP la comunicación entre distintas computadoras ya que las cabeceras de IP (cabecera por ser una parte del protocolo, que aquí no tiene importancia) contienen las direcciones de las máquinas de origen y destino, llamadas direcciones IP. Estas direcciones serán usadas por los routers (routers) para decidir el tramo de red por el que se enviarán los paquetes.

Con esto concluimos una introducción de lo que es la comunicación entre computadoras en internet. Si bien parece algo complejo es de mucha importancia conocer estos mecanismos con los que trabajamos.

HYPERTEXT TRANSFER PROTOCOL (HTTP) (o Protocolo de Transferencia de Hipertexto en español)

es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también.

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados HTTP verbs.

Es muy importante saber qué HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. Luego veremos las implicancias de esto.

Para entender mejor esta distinción entre dos protocolos uno TCP y otro IP habría que analizar el modelo de capas OSI que no vamos a ver. Lo que sí vamos a decir es que hay una jerarquía entre capas y el protocolo IP pertenece a una capa denominada de red que está por encima de una capa denominada de transporte en dónde se encuentra TCP

EJEMPLO



Entendiendo la lógica de la programación de una App Web

Para que entendamos la forma de cómo se programa una aplicación web necesitamos en alguna medida entender los mensajes HTTP. Los mensajes HTTP son en texto plano lo que los hace más legible y fácil de depurar. Estos tienen la siguiente estructura:

VER ESTRUCTURAS



VER EJEMPLO



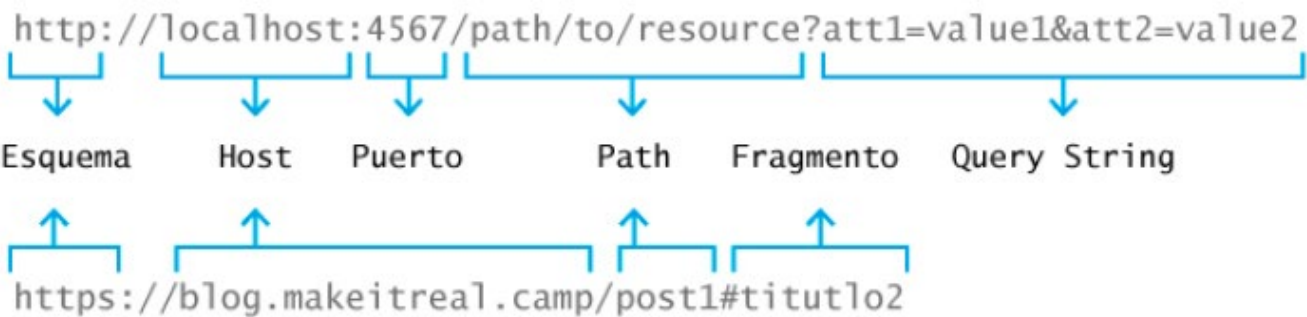
Una descripción importante del protocolo es que está orientado a transacciones y sigue el esquema de petición/respuesta entre un cliente y un servidor. El cliente realiza una petición enviando un mensaje con cierto formato al servidor. El servidor le envía un mensaje de respuesta. Para hacerlo más concreto un cliente podría ser un navegador web y un servidor podría ser una aplicación en un servidor web corriendo en Internet.

Los otros protocolos de TCP/IP son necesarios para configuraciones de conexiones, pero a nivel aplicación no los utilizaremos. Eso no significa que no haya que entenderlos pues nos facilitarán la solución de problemas, por ejemplo, al conectarnos a una base de datos desde la aplicación web del servidor.

METODOS	DESCRIPCIÓN	CODIGOS DE ESTADO
GET	El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.	<div>Cuando el servidor devuelve una respuesta se indica un código de estado:</div> <div>- 1xx: Mensaje Informativo.</div> <div>- 2xx: Exito</div> <div> * 200 Ok</div> <div> * 201 Created</div> <div> * 202 Accepted</div> <div> * 204 No Content</div> <div>- 3xx: Redirección</div> <div> * 300 Multiple Choice</div> <div> * 301 Moved Permanently</div> <div> * 302 Found</div> <div> * 304 Not Modified</div> <div>- 4xx: Error del Cliente</div> <div> * 400 Bad Request</div> <div> * 401 Unauthorized</div> <div> * 403 Forbidden</div> <div> * 404 Not Found</div>
HEAD	El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.	
POST	El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.	
PUT	El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.	
DELETE	El método DELETE borra un recurso en específico.	
CONNECT	El método CONNECT establece un túnel hacia el servidor identificado por el recurso.	

OPTIONS	El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.	- 5xx: Error del Servidor * 500 Internal Server Error * 502 Bad Gateway * 503 Service Unavailable
TRACE	El método TRACE realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.	
PATH	El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.	

COMPOSICIÓN DE UNA URL



Transferencia de datos

Conceptualización de las estructuras involucradas en la transferencia de datos entre computadoras

Para entender la programación web es necesario además conocer o tener alguna noción sobre las estructuras involucradas en la transferencia de datos entre computadoras. Esta transferencia de datos, es decir, la comunicación entre dos o más computadoras se lleva a cabo a través de lo que se llama normalmente red de computadoras. La más conocida red es Internet o, como se la denomina: la red de redes.

Vamos a ver conceptos de red de computadoras, de Internet y de cómo funcionan los elementos que las constituyen, a modo introductorio para entender cómo funcionan los sistemas para la Web.

Para comenzar vamos a definir que “ (...) una red de computadoras, también llamada red de ordenadores, red de comunicación de datos o red informática, es un conjunto de equipos nodos y software conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos con la finalidad de compartir información recursos y ofrecer servicios (...) ” (Tanenbaum, 2003)



La finalidad principal para la creación de una red de ordenadores es:

- compartir recursos e información a grandes distancias
- asegurar la confiabilidad y la disponibilidad de la información (características propias que definen lo que es información)

- aumentar la velocidad de transmisión de los datos y
- reducir los costos. (CENS, 2018)

Si bien este no es un curso de redes de computadora es necesario conocer los elementos físicos en los que trabajan los sistemas que vamos a realizar. Con ello queremos decir que si bien podemos abstraernos de esta tecnología, es muy útil tener alguna noción de lo que pasa detrás de escena.

La comunicación por medio de una red se lleva a cabo en dos categorías diferentes: una capa denominada física y otra lógica.

La **capa física** incluye todos los elementos de los que hace uso un equipo para comunicarse con otro equipo dentro de la red, como por ejemplo, tarjetas de red, los cables, las antenas, etc. Si te interesa conocer más podés leer autores como [Tanenbaum](#) en donde habla de redes y el modelo OSI (modelo de interconexión de sistemas, protocolos de comunicación, etc.) .

Con respecto a la **capa lógica** la comunicación se rige por normas muy rudimentarias que por sí mismas resultan de escasa utilidad. Sin embargo, haciendo uso de dichas normas es posible construir los protocolos denominados, que son normas de comunicación más complejas (de alto nivel) capaces de proporcionar servicios útiles.

Los protocolos son un concepto muy similar al de los idiomas de las personas. Es decir, si dos personas hablan el mismo idioma, y respetan ciertas reglas (tales como hablar y escucharse por turnos), es posible comunicarse y transmitir ideas e información. Ese es el modelo de un protocolo.

Para formar una red se requieren elementos de hardware, software y protocolos. Los elementos físicos se clasifican en dos grupos: los dispositivos de usuario final (llamados también Hosts) y los dispositivos de red. Entre los dispositivos de usuario final podemos enumerar computadoras, impresoras, escáneres, y demás elementos que brindan servicios directamente al usuario. Los segundos (dispositivos de red) son todos aquellos que conectan entre sí a los dispositivos de usuario finales posibilitando su intercomunicación. (Farías, 2013).

Internet es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos [TCP / IP](#), lo cual garantiza que las redes físicas heterogéneas que la componen, constituyan una red lógica única de alcance mundial. (Guillamón, 2009)

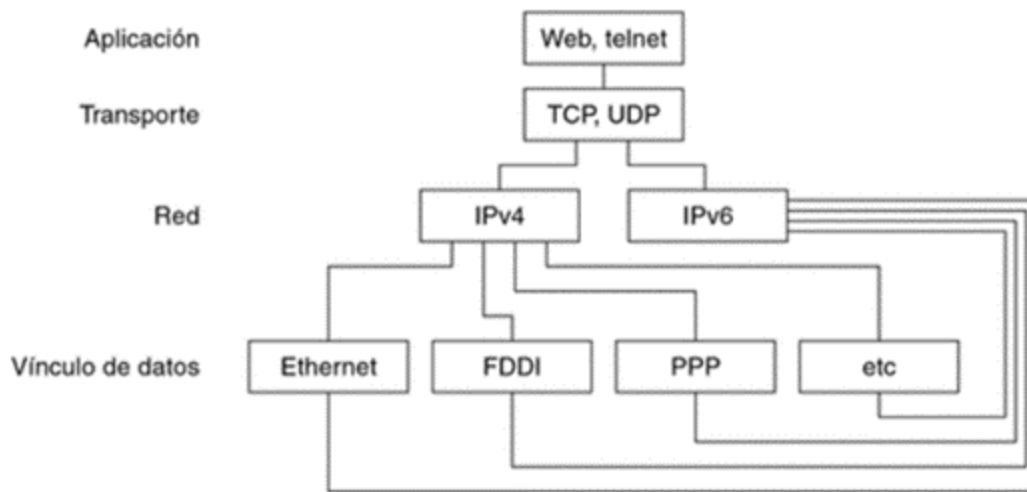
Uno de los servicios que más éxito ha tenido en Internet ha sido [world wide web](#), WWW o la web. La WWW es un conjunto de protocolos que permite de forma sencilla la consulta remota de archivos de hipertexto y utiliza Internet como medio de transmisión («Internet, n.». Oxford English Dictionary (Draft edición). Marzo de 2009).

Para hacer más notable esta distinción vamos a nombrar algunos otros servicios y protocolos en Internet aparte de la web por ejemplo para el envío de correo electrónico usamos el protocolo [SMTP](#), para la transmisión de archivos usamos el protocolo [FTP](#), para nombrar algunos.

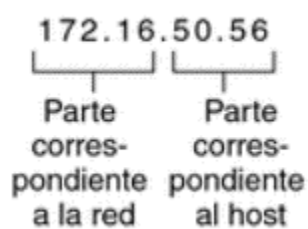
Esta familia de protocolos es un conjunto de protocolos de red en los que se basa Internet y permite la transmisión de datos entre computadoras como hemos dicho.

Entre los principales podemos nombrar el conjunto de protocolos TCP / IP que hace referencia a los dos protocolos más importantes que componen la Internet, que fueron los primeros en definirse y qué son los más utilizados.

TCP (protocolo de control de transmisión) se usa para crear conexiones entre computadoras a través de las cuales pueden enviarse un flujo de datos. Por la forma en la que está implementado este protocolo los datos serán entregados en su destino sin errores y en el mismo orden en el que se transmitieron. ¿Esto qué quiere decir? que es un protocolo orientado a conexión, ya que el cliente y el servidor deben anunciarse y aceptar la conexión antes de comenzar a transmitir los datos entre ellos. Es decir que hay un intercambio de mensajes entre ellos para abrir una línea de conexión que permanece abierta durante toda la comunicación. (dsi.uclm.es 2007)



Por otro lado, el protocolo IP es un protocolo cuya función principal es el uso direccional en origen o destino de comunicación para transmitir datos mediante un protocolo no orientado a conexión que transfiere paquetes conmutados a través de distintas redes previamente enlazadas según la norma OSI.



Algo importante del diseño del protocolo IP es que se realizó suponiendo que la entrega de los paquetes de datos sería no confiable por eso se tratará de realizar del mejor modo posible mediante técnicas de enrutamiento sin garantías de alcanzar el destino final, pero tratando de buscar la mejor ruta entre las conocidas por la máquina que está usando IP.

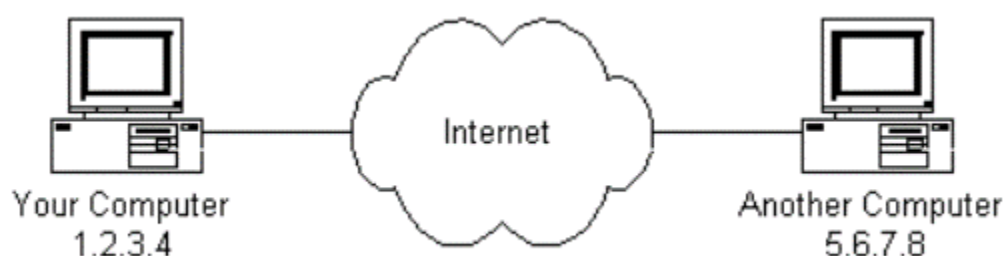
Para entender mejor esta distinción entre dos protocolos uno TCP y otro IP habría que analizar el modelo de capas OSI en éste curso no profundizaremos sobre esto, pero lo que sí vamos a decir es que hay una jerarquía entre capas y el protocolo IP pertenece a una capa denominada de red que está por encima de una capa denominada de transporte en dónde se encuentra TCP.

Entonces, en conclusión, se utiliza la combinación de estos dos protocolos para la comunicación en Internet, en donde TCP aporta la fiabilidad entre la comunicación e IP la comunicación entre distintas computadoras ya que las cabeceras de IP (cabecera por ser una parte del protocolo) contienen las direcciones de destino de las máquinas de origen y llamadas direcciones IP. Estas direcciones serán usadas por los routers para decidir el tramo de red por el que se enviarán los paquetes.

Para entender mejor el funcionamiento de la Internet vamos a decir que **dentro de la red de redes que es Internet debe existir un mecanismo para conectar dos computadoras. Este mecanismo lo proporciona el protocolo de Internet, el cual hace que un paquete de una computadora, llegue a la otra de manera segura a través del protocolo TCP y que llegue a destino a través de las direcciones IP.**

Para terminar, una dirección IP es un número que identifica de manera lógica y jerárquica una interfaz de un dispositivo dentro de una red que utiliza el protocolo de internet.

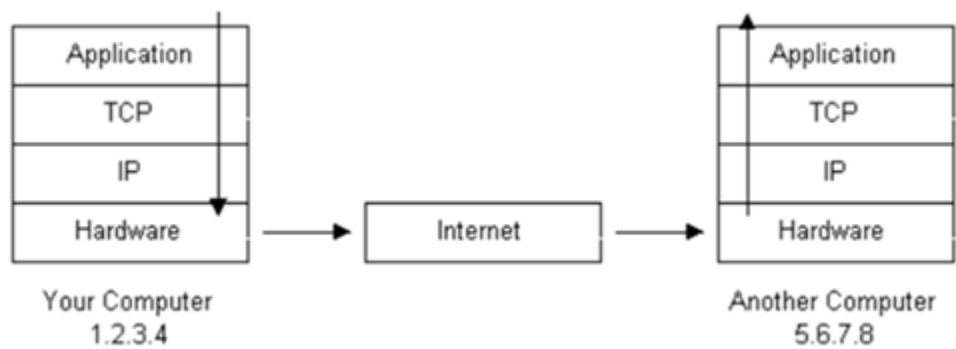
Todas las computadoras en internet tienen una dirección IP. A modo informativo vamos a decir que existe otro sistema que se denomina sistema de nombres de dominio o DNS que asocian nombres comunes a direcciones IP por ejemplo la dirección www.cba.gov.ar tiene asociado un número de IP correspondiente, pero este mecanismo existe para que sea más fácil llegar a esa página web sin tener que recordar el número de IP.



Con esto concluimos una introducción de lo que es la comunicación entre computadoras en internet. Si bien parece algo complejo es de mucha importancia conocer estos mecanismos con los que trabajamos.

Por ejemplo, si queremos comunicarnos con una base de datos en Internet y nos pide un número de IP ya sabemos de qué se trata. También sabemos que podemos comunicarnos a cualquier computadora conectada en red a través de mecanismos y que nos abstraemos de cómo se hace esto. Es decir, no nos tenemos que preocupar de manejar errores o interferencias en la comunicación.

Ya vimos que la estructura de red se maneja en capas. También mencionamos que hay una capa de red en dónde está el protocolo IP, una capa Superior de transporte en dónde está el protocolo TCP y ahora vemos una nueva capa que es la de aplicación en dónde se usa el protocolo HTTP.



Por otro lado, antes de continuar con el desarrollo web tradicional tenemos que aprender un nuevo protocolo que es imprescindible para la comunicación en la web.

HyperText Transfer Protocol (HTTP)

Hypertext Transfer Protocol ([HTTP](#)) (**Protocolo de Transferencia de Hipertexto** en español) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también.

HTTP define un conjunto de [métodos de petición](#) para indicar la acción que desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados *verbos HTTP* .

Es muy importante saber que HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. Luego veremos las implicancias de esto.

Una descripción importante del protocolo es que **está orientado a transacciones y sigue el esquema de petición / respuesta entre un cliente y un servidor**. El cliente realiza una petición enviando un mensaje con cierto formato al servidor. El servidor le envía un mensaje de respuesta. Para hacerlo más concreto, un cliente podría ser un navegador web y un servidor podría ser una aplicación en un servidor web corriendo en Internet.

Para que entendamos la forma de cómo se programa una aplicación web, necesitamos en alguna medida entender los mensajes HTTP. Los mensajes HTTP son en texto plano lo que hace más legible y fácil de depurar. Estos tienen la siguiente estructura:

- Primero, hay una línea inicial en donde se diferencian dependiendo de si son peticiones y respuestas. Para las solicitudes la línea comienza con una acción requerida por el servidor, a esto se le denomina método de petición seguido de la url del recurso y la versión http que soporte al cliente. Lo importante es el método de petición y la URL (Uniform Resource Locator o localizador de recursos uniforme).
- Para las respuestas, la línea comienza con la versión de HTTP seguido por un código de respuesta y con una frase asociada a dicho retorno.
- También los mensajes tienen una cabecera que son metadatos con información diversa y el cuerpo de mensaje que es opcional. Típicamente este cuerpo tiene los datos que se intercambian entre el cliente y el servidor.

Los métodos de petición (o también llamados verbos) son varios. Cada método indica la acción que desea que se efectúe sobre el recurso identificado lo que este recurso representa dependiente de la aplicación del servidor.

Los métodos que vamos a usar son los de **get** y **post** . Cabe destacar que hay convenciones en donde se utilizan otros métodos. En este punto entran en juego el estándar REST y las API's.

El método get solicita una representación del recurso especificado. Las solicitudes que usan sólo deben recuperar datos y no deben tener ningún otro efecto.

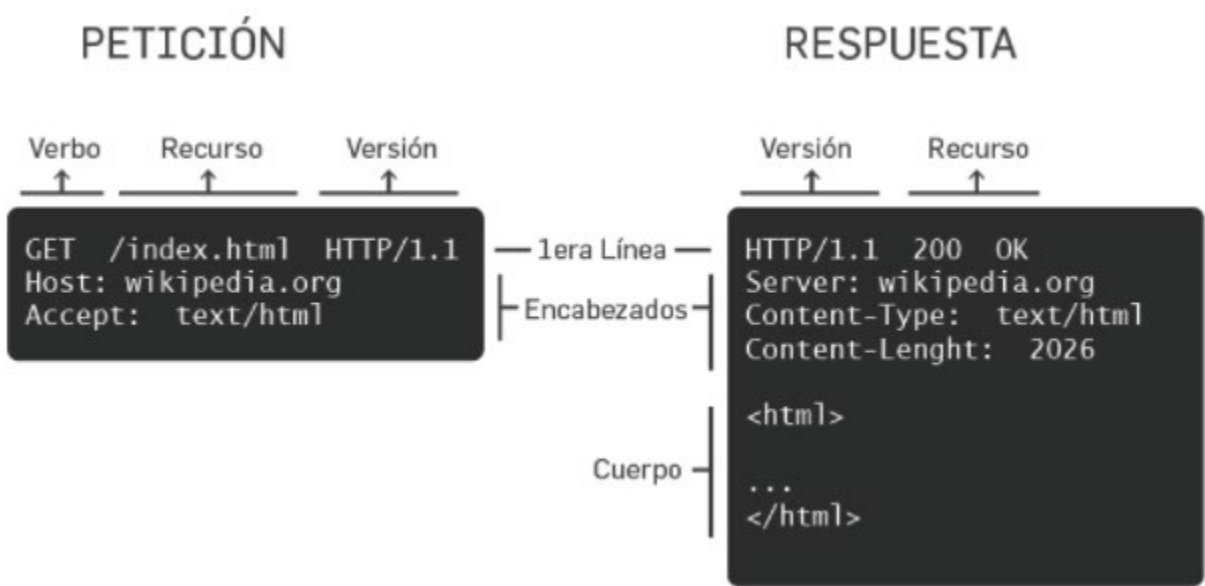
El método post envía los datos para que sean procesados por el recurso identificado. Los datos enviados se incluirán en el de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes.

Para finalizar con la explicación de HTTP vamos a nombrar códigos de respuesta:

- 1. Por ejemplo, el 200 representa una respuesta correcta es decir que indica que la petición ha sido procesada correctamente.
- 2. Otro ejemplo es la respuesta 404 que significa errores causados por el cliente, por ejemplo que el recurso no se encuentra.
- 3. Finalmente, las respuestas que comienzan con 5 por ejemplo el 500 son errores causados por el servidor. Esto indica que ha habido un error en el proceso de la petición a causa de un fallo en el servidor.

Todo lo que hablamos de HTTP lo vamos a usar al momento de programar una aplicación web. Vamos a ver que al presionar por ejemplo un enlace se hace una petición “get” al servidor para buscar otra página lo que resultará en una respuesta 200 si se encuentra la página o en un 404 si no se encuentra. También veremos que al llenar un formulario y enviarlo al servidor lo haremos a través de una petición post.

Los otros protocolos de TCP / IP son necesarios para configuraciones de conexiones, pero a nivel de aplicación no los utilizaremos. Eso no significa que no haya que entenderlos pues nos facilitarán la solución de problemas, por ejemplo, al conectarnos a una base de datos desde la aplicación web del servidor.



Method	Description	Sec.
GET	Transfer a current representation of the target resource.	4.3.1
HEAD	Same as GET, but only transfer the status line and header section.	4.3.2
POST	Perform resource-specific processing on the request payload.	4.3.3
PUT	Replace all current representations of the target resource with the request payload.	4.3.4
DELETE	Remove all current representations of the target resource.	4.3.5
CONNECT	Establish a tunnel to the server identified by the target resource.	4.3.6
OPTIONS	Describe the communication options for the target resource.	4.3.7
TRACE	Perform a message loop-back test along the path to the target resource.	4.3.8

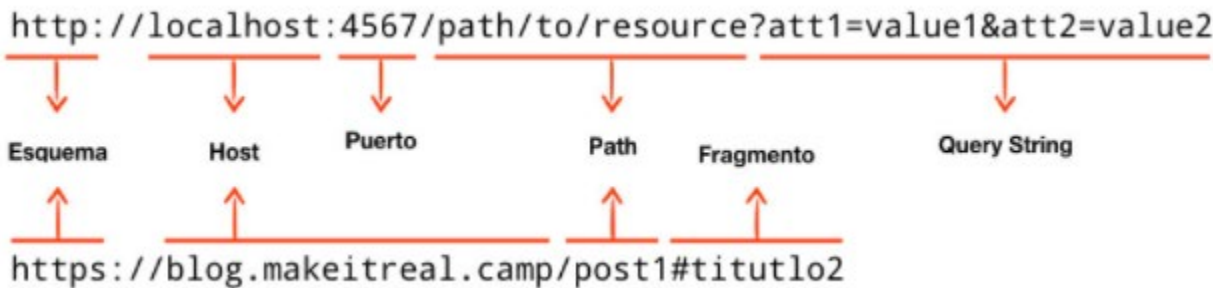
Verbos y cómo se relacionan con operaciones

Código de estados

Cuando el servidor devuelve una respuesta se indica un código de estado:

- **1xx:** Mensaje informativo.
 - **2xx:** Éxito
 - 200 OK
 - 201 Created
 - 202 Accepted
 - 204 No Content
 - **3xx:** Redirección
 - 300 Multiple Choice
 - 301 Moved Permanently
 - 302 Found
 - 304 Not Modified
- **4xx:** Error del cliente
 - 400 Bad Request
 - 401 Unauthorized
 - 403 Forbidden
 - 404 Not Found
 - **5xx:** Error del servidor
 - 500 Internal Server Error
 - 501 Not Implemented
 - 502 Bad Gateway
 - 503 Service Unavailable

Código de respuesta HTTP:



URL: Composición de la URL

Glosario

- Bases de datos:** Una base de datos es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora (<https://www.oracle.com/mx/database/what-is-database>)
- DOM:** El *DOM* (**Modelo de Objetos de Documento** en español **Modelo de Objetos del Documento**) es una API definida para representar e interactuar con cualquier documento HTML o XML. El DOM es un modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos, en donde cada nodo representa una parte del documento (puede tratarse de un elemento, una cadena de texto o un comentario). (<https://developer.mozilla.org/es/docs/Glossary/DOM>)
- Frameworks:** La traducción literal de *framework* es ' **marco de referencia** ', y explica muy bien lo que significa. Un *framework* es un patrón o esquema que ayuda a la programación a estructurar el código ya ahorrar tiempo y esfuerzos a los programadores. (<https://fp.uoc.fje.edu/blog/que-es-un-framework-en-programacion/>)

LAN: LAN (red de área local) describe una red cuya área geográfica no se extiende más de una milla
(<http://www.frlp.utn.edu.ar/materias/info2/Redes%20Lan.html>)

Python: Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad
(<https://desarrolloweb.com/articulos/1325.php>)

Rails: Ruby on Rails (o simplemente Rails) es un framework Open Source, multiplataforma y basado en el lenguaje de scripting Ruby que facilita sobremanera el diseño y desarrollo de aplicaciones web que acceden a bases de datos. Rails separa automáticamente en tres capas todos los componentes de una aplicación web (Model, View y Controller: MVC), lo que hace más sencillo y rápido el mantenimiento de aplicaciones que en otros entornos (<https://www.spri.eus/euskadinnova/es/enpresa-digitala/agenda/introduccion-ruby-rails-completo/709.aspx#:~:text=Ruby%20on%20Rails%20>)

Swift: Swift es un lenguaje intuitivo de programación de código abierto creado por Apple que permite diseñar aplicaciones para iOS, Mac, el Apple TV y el Apple Watch. (<https://www.apple.com/es/swift/>).

WAN: (Wide Area Network) describe una red cuya área geográfica excede a la de una ciudad. A menudo, varias LAN o varias MAN son enlazadas para crear una WAN
(<http://www.frlp.utn.edu.ar/materias/info2/Redes%20Lan.html>)