

Introducción a la Tecnología

Sección I



Contenido

01

Historia breve

Contexto general de la
tecnología

03

Desarrollo de Software

Introducción a desarrollo
de software

02

Principios básicos

Conocimientos básicos
de la informática

04

Tecnologías aplicadas

Lenguajes de programación
y su uso



Contenido

05

Tipos de roles

Explicación de roles en
tecnología

07

Principales avances

Avances más importantes
en tecnología

06

Conocimientos básicos

Conocimientos para
desarrollo en general

08

Conclusiones

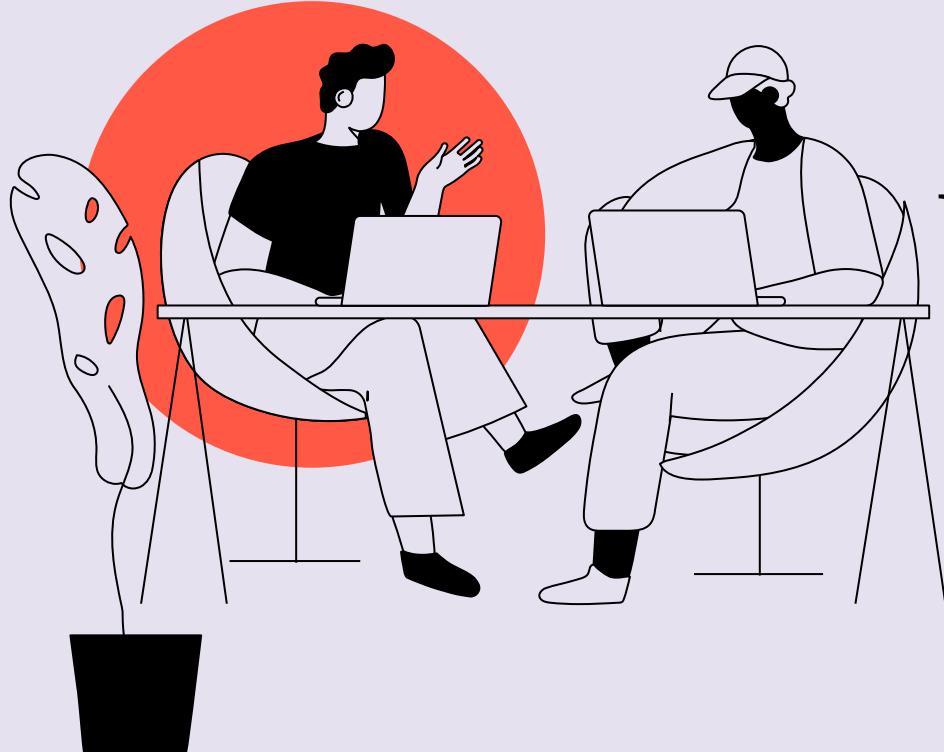
Puntos finales de la
sección



01

Historia breve

de la tecnología





¿Qué significa tecnología?

Conjunto de **teorías** y de **técnicas** que permiten el **aprovechamiento práctico** del conocimiento científico. - RAE

La tecnología es el **conjunto de conocimientos y técnicas** que se aplican de manera ordenada para alcanzar un **determinado objetivo o resolver un problema**. - economipedia



→ ¿Qué significa tecnologías de la información?

Tecnologías de la información - TI

Cualquier equipo o sistema interconectado o subsistema de equipos que se utilice en la **adquisición, almacenamiento, manipulación, gestión, movimiento, control, visualización, conmutación, intercambio, transmisión o recepción automática de datos o información** - CSRC



¿Qué significa tecnologías de la información?



Puntos importantes:

- Obtención
- Almacenamiento
- Manipulación / Transformación
- Gestión / Administración
- Movimiento
- Control
- Visualización
- Conmutación
- Intercambio, transmisión o recepción automática

de **datos o información**



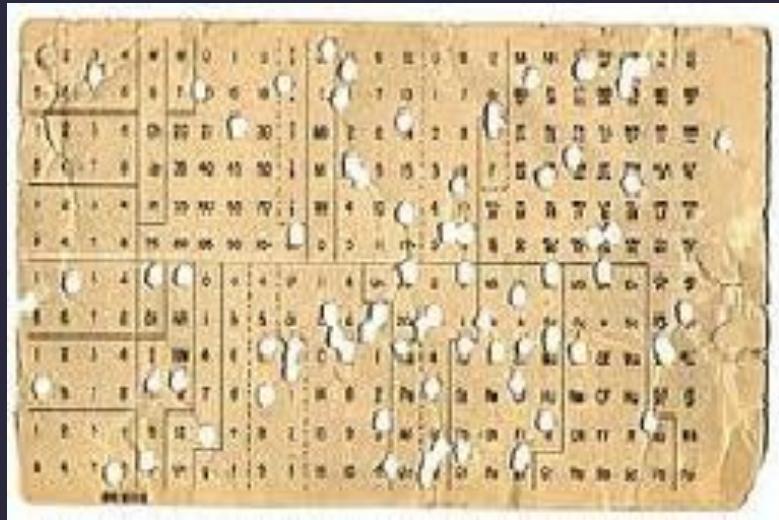
HISTORIA DE LA COMPUTACIÓN

(Línea del tiempo)





Ábaco



- > Aparece el ábaco, el cual fue el primer instrumento utilizado por la humanidad para realizar operaciones de cálculo.



2,500 A.C.

Sistema Binario

	天	澤	火	雷	風	水	山	地
天	乾	履	同人	無妄	姤	訟	遁	否
澤	夬	夬	革	隨	大過	困	咸	萃
火	大有	睽	離	噬嗑	鼎	未濟	旅	晉
雷	大壯	歸妹	豐	巽	恒	解	小過	豫
風	小畜	中孚	家人	益	巽	渙	漸	觀
水	需	師	既濟	屯	井	坎	蹇	比
山	大畜	損	賁	頤	蠱	蒙	艮	剝
地	泰	臨	明夷	復	升	師	謙	坤

- Aparece el libro de las mutaciones (I Ching) y en él encontramos la primera formulación del sistema binario.





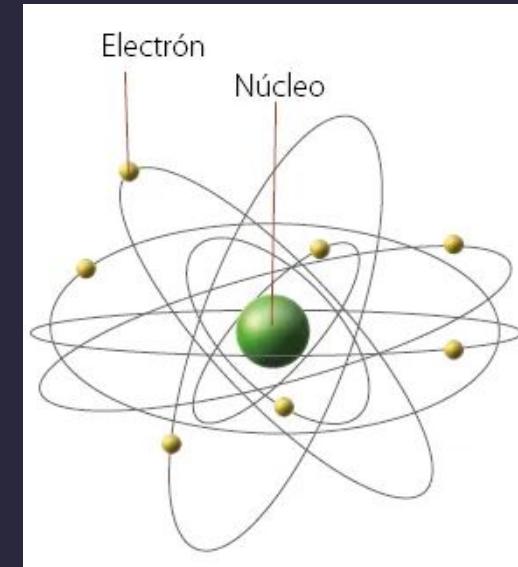
Alfabeto (Fenicio)



- Fue uno de los sistemas de escritura más utilizados al ser difundido a lo largo del mundo mediterráneo. Fue asimilado por muchas otras culturas adaptándolo a sus respectivos idiomas.



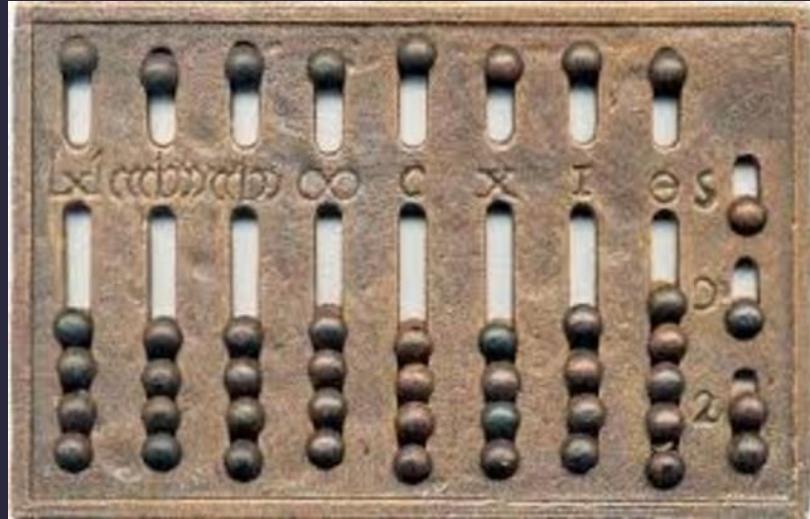
Electrón



- > Tales de Mileto describe la electricidad estática y de ahí proviene el término electrón.

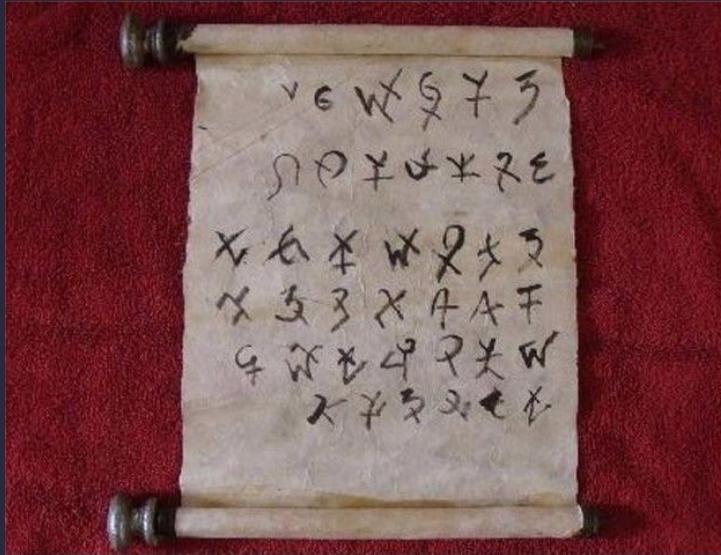
Cálculos

(Ábaco romano)



- Los romanos utilizan ábacos y para las bolas usan piedras a las que llaman cálculos.

Pergamino



- En la ciudad de Pérgamo se empieza a utilizar pergamino para la escritura.

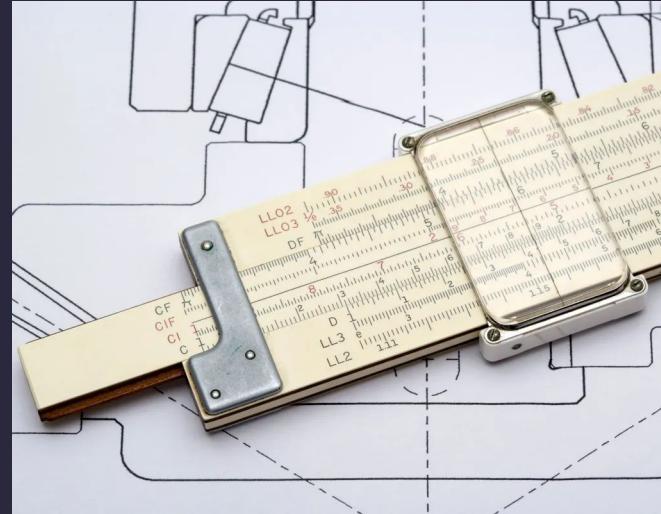
Papel



- En China se inventa el papel. Se dice que Ts'ai Lun presentó por primera vez al emperador Han Ho Ti, de la dinastía Han de China, muestras de papel

Regla de Cálculo

(William Oughtred)



- > William Oughtred creó una regla de cálculo, la cual ha sido usada por ingenieros hasta hace muy pocos años

Calculadora mecánica sumas y restas



- > Blaise Pascal construyó la primera calculadora mecánica que hacía sumas y restas

Calculadora para multiplicaciones



- Gottfried Leibnitz crea una máquina que es capaz de realizar multiplicaciones

Diseño de máquina capaz de ejecutar programas de computación



- > Charles Babbage diseña aunque no construye una máquina procesadora capaz de ejecutar programas de computación

Álgebra de Boole

REGLAS DEL ALGEBRA BOOLEANA

$$1. A + 0 = A$$

$$2. A + 1 = 1$$

$$3. A \cdot 0 = 0$$

$$4. A \cdot 1 = A$$

$$5. A + A = A$$

$$6. A + \overline{A} = 1$$

$$7. A \cdot A = A$$

$$8. A \cdot \overline{A} = 0$$

$$9. \overline{\overline{A}} = A$$

$$10. A + AB = A$$

$$11. A + \overline{A}B = A + B$$

$$12. (A + B)(A + C) = A + BC$$

A, B o C pueden representar una sola variable o una combinación de variables.

- > Nace el álgebra de Boole de la mano de George Boole. Se inician los estudios de la lógica simbólica

Perforadora mecánica



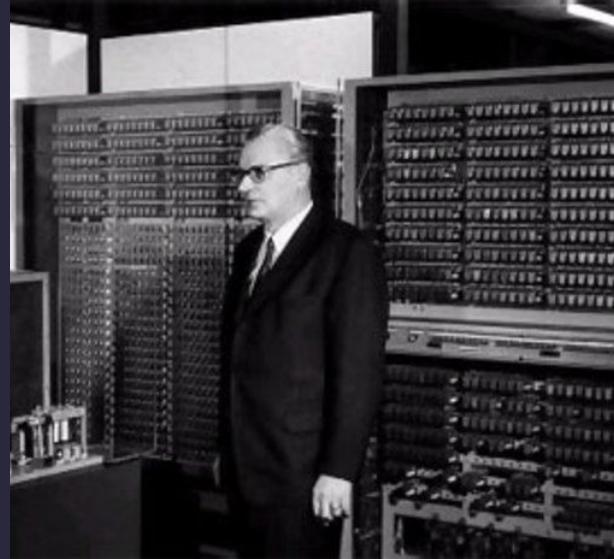
- Hermann Hollerith utiliza una perforadora mecánica como forma de representar letras y dígitos mediante tarjetas de papel.
Más adelante, en el año 1924, sería el fundador de IBM

Primera transmisión de radio



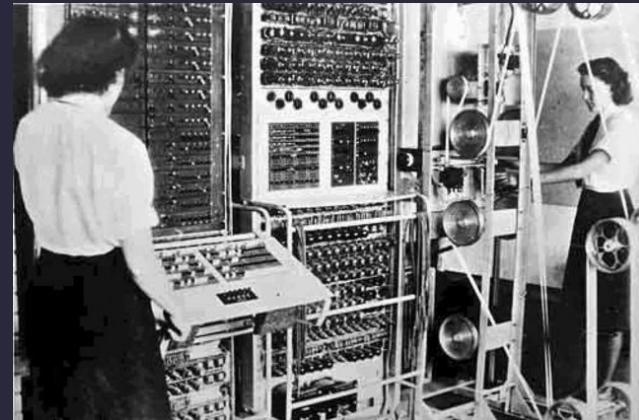
- > Durante la nochebuena de 1906 se realiza la primera transmisión de radio

Primera computadora electromagnética programable



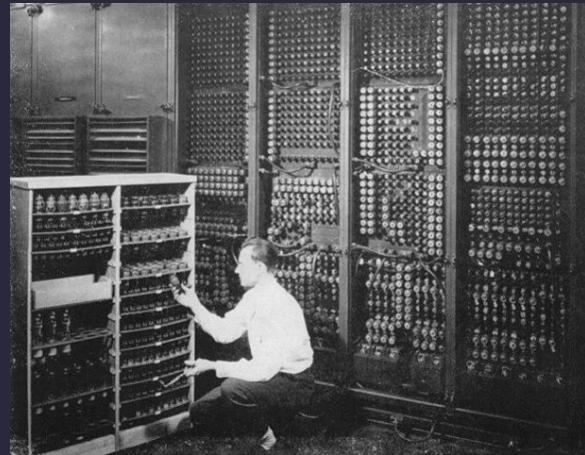
- > Konrad Suze presentó la primera computadora electromagnética programable con una cinta perforada. Tenía 2000 electroimanes, 64 palabras de 22 bits de memoria, 1000 kilos de peso y un consumo de 4000 vatios

Colossus



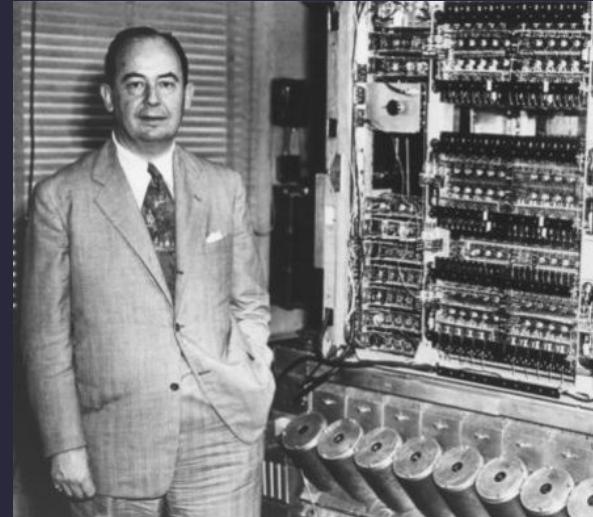
- Alan Turing construye el Colossus, un ordenador que permite descifrar en unos pocos segundos los mensajes secretos de los nazis en la Segunda Guerra Mundial (Máquina Enigma)

ENIAC



> John Presper Eckert y John W. Mauchly desarrollaron el ENIAC (Electronic Numerical Integrator And Calculator). Es considerado como el primer ordenador, ya que funcionaba de forma completamente electrónica

Arquitectura Von Neumann EDVAC



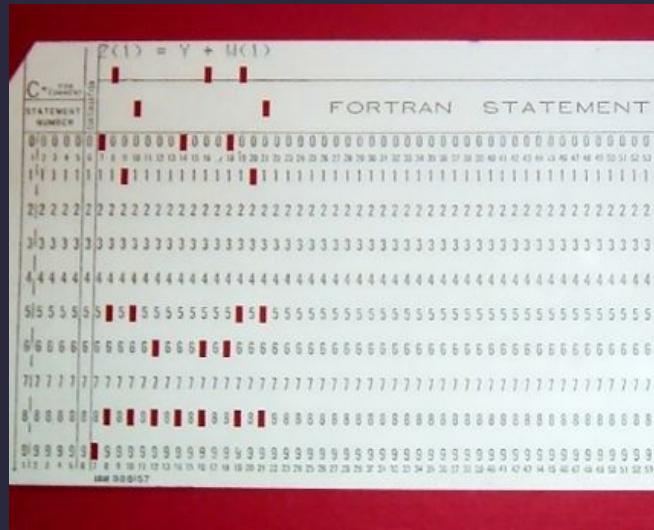
- > John Von Neumann creó la EDVAC. Tuvo la idea de poner las instrucciones en la misma memoria que los datos. Las escribía en código binario. A esto se le conoce como Arquitectura Von Neumann y es utilizado actualmente

Nace el UNIVAC 1



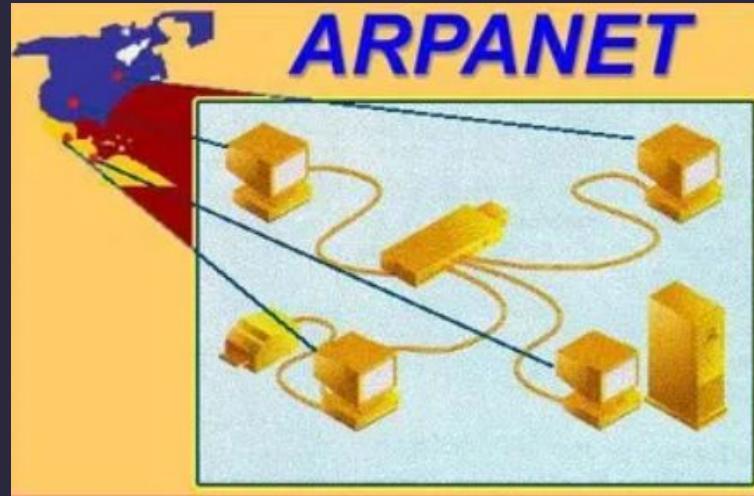
- Nació primer ordenador comercial. Es el UNIVAC 1, que fué desarrollado por la Howard Aiken Sperry-Rand Corporation y lo compró la Oficina del Censo de Estados Unidos

Crean el primer lenguaje de programación



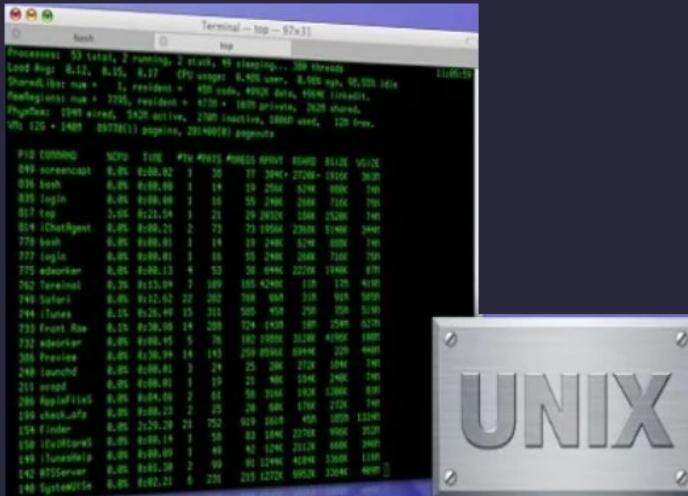
- > John Backus y sus compañeros en IBM, crearon el primer Fortran, el primer lenguaje de programación

Aparece ARPA



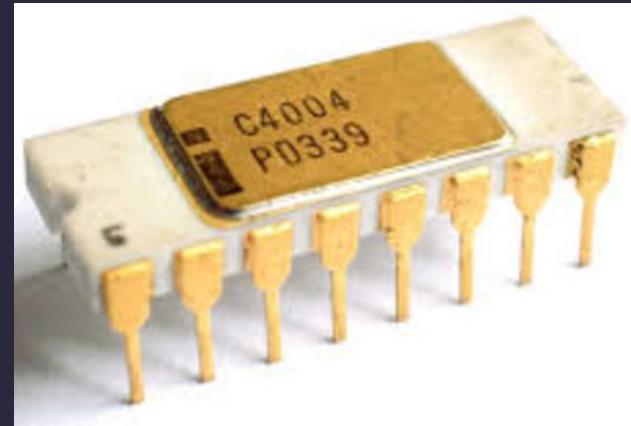
- ARPA es una agencia del Ministerio de Defensa de Estados Unidos. Allí J.C.R. Licklider defiende con éxito sus ideas acerca de una red de ordenadores global (Inicio de Internet)

UNIX



› Creación del sistema operativo UNIX

Primer microprocesador de Silicio



- > Intel fabrica y saca al mercado el primer microprocesador de silicio. Es el Intel 4004. Fué creado para realizar las operaciones básicas de Babbage con arreglo a la arquitectura de Von Neumann. Se trata de la primera CPU

Primera computadora de Apple



- Es el inicio de lo que hoy es Apple. Steve Jobs y Steven Wozniak querían fabricar y comercializar un ordenador para que fuera usado de forma masiva

Primer Ordenador de Sobremesa



- > Commodore desarrolla y comercializa el ordenador de sobremesa más vendido a nivel mundial

Nace Primer "Personal Computer" (PC)



- Nace el primero de los "Personal Computer". El primero de la plataforma de hardware compatible IBM PC. El IBM 5150. Además se define el protocolo TCP/IP y la palabra Internet

Crean Primer Servidor y primer teléfono móvil **DynaTAC 8000X**



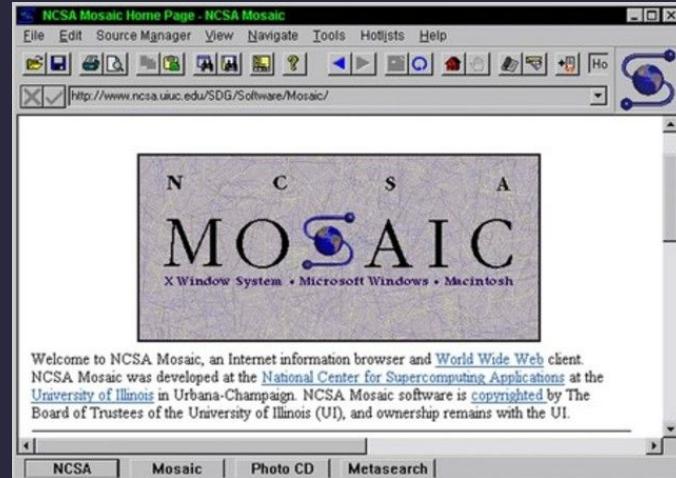
- Se crea el 1er servidor de nombres de sitios. Motorola presenta el primer teléfono móvil. El DynaTAC 8000X

Se crea un prototipo World Wide Web



- 100,000 ordenadores conectados a Internet y se crea un prototipo de la World Wide Web

Aparece el navegador para Internet Mosaic



- Aparece el navegador para Internet Mosaic, el primer buscador de Internet, se llama Wandex. El IBM Simon se convierte en el primer teléfono móvil que integra funciones de PDA

Aparecen buscadores de Internet

- En 1994 aparecen los buscadores de Internet WebCrawler, Lycos y Excite. Y en 1995 Internet AltaVista y Yahoo



10.000.000 de ordenadores conectados a Internet



- 10.000.000 de ordenadores conectados a Internet. El Nokia 9000 Communicator se convierte en el smartphone e integraba una CPU Intel 386

Google

› Hace su aparición Google



Primer libro Digital



- Aparece el 1er libro digital

Primera red social LinkedIn



- Se lanza LinkedIn siendo la primera red social profesional de la historia. Este año también aparecen MySpace y Hi5

Aparece Facebook y otras redes sociales



- > Aparece Facebook, Flickr, Vimeo, Tagged y la primera red social de Google llamada Orkut

Nace Youtube



› Se crea YouTube, DailyMotion y Reddit

Aparece el iPhone



- Se crea YouTube, DailyMotion y Reddit

Aparece Whatsapp



- Se crea WhatsApp y el buscador de Internet de Microsoft Bing

Timeline of Computer History

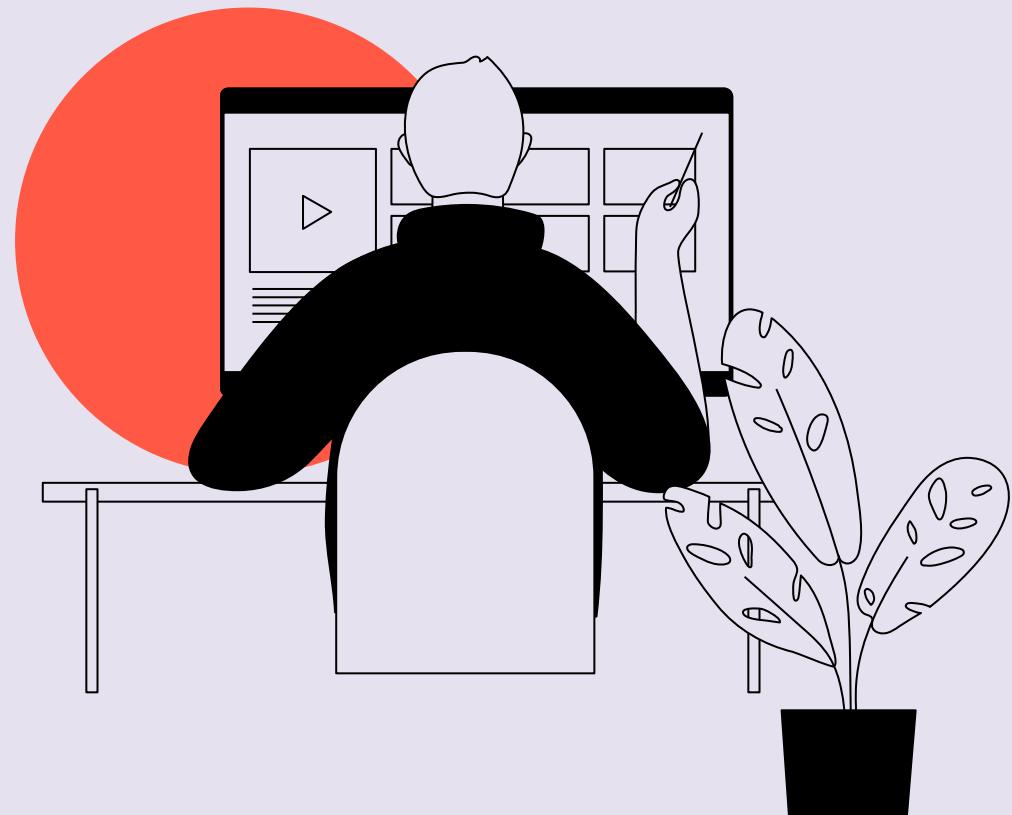


> <https://www.computerhistory.org/timeline/>

02

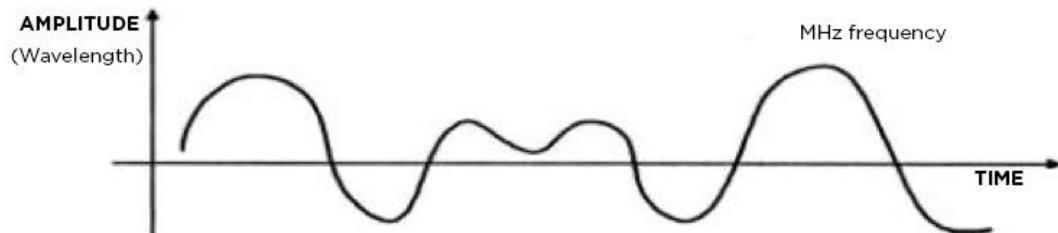
Principios

básicos

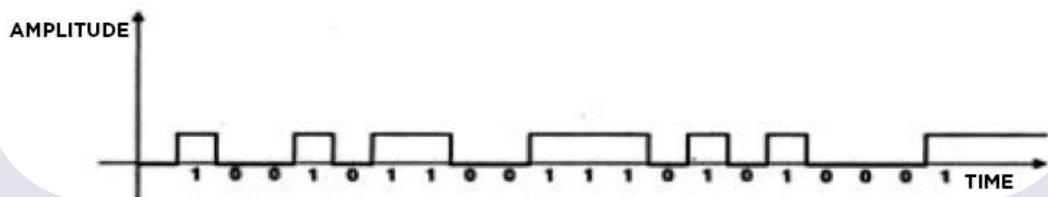


Análogo vs Digital

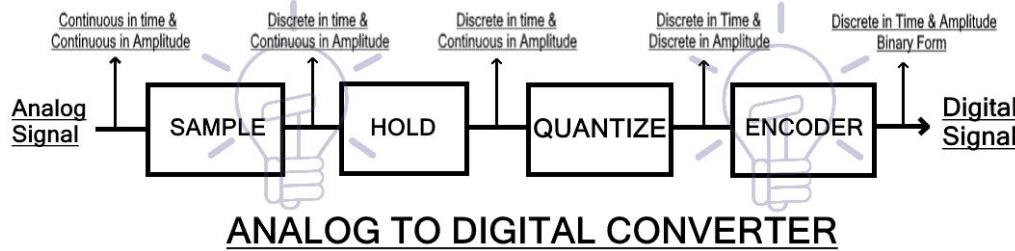
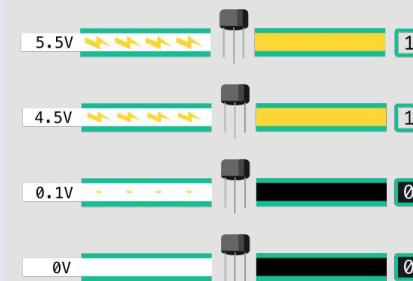
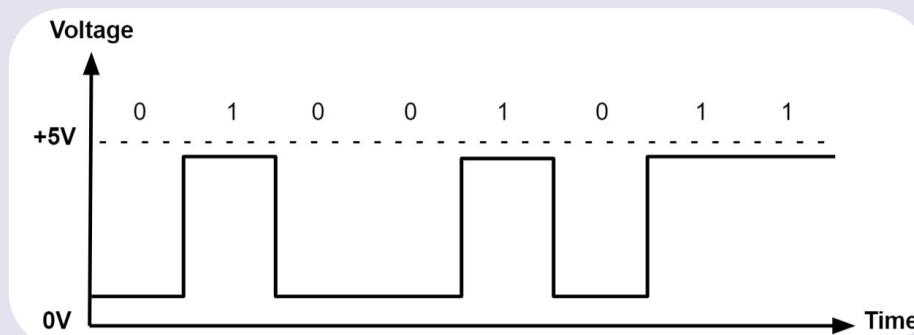
ANALOG SIGNAL



DIGITAL SIGNAL



Análogo vs Digital





Bit vs Byte

- 64 o 32 bits
- 1024 megabytes / 1024 megabits
- 1024 megabytes / 512 gigabytes

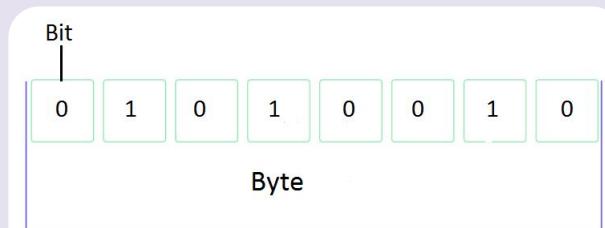




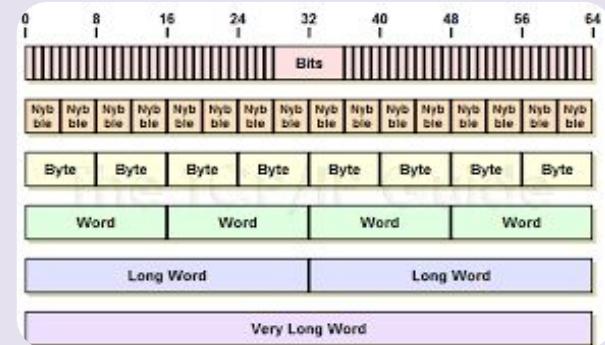
Bit vs Byte

- El **bit (b)** es la **unidad básica** en la que se mide la información en el entorno de las computadoras
- El **byte (B)** es una unidad de medida que está compuesta de **8 bits**, es una **unidad de base octadecimal**

En contraposición a lo que ocurre con el resto de unidades que más solemos manejar, como el metro, el gramo, el litro cuya base es decimal.

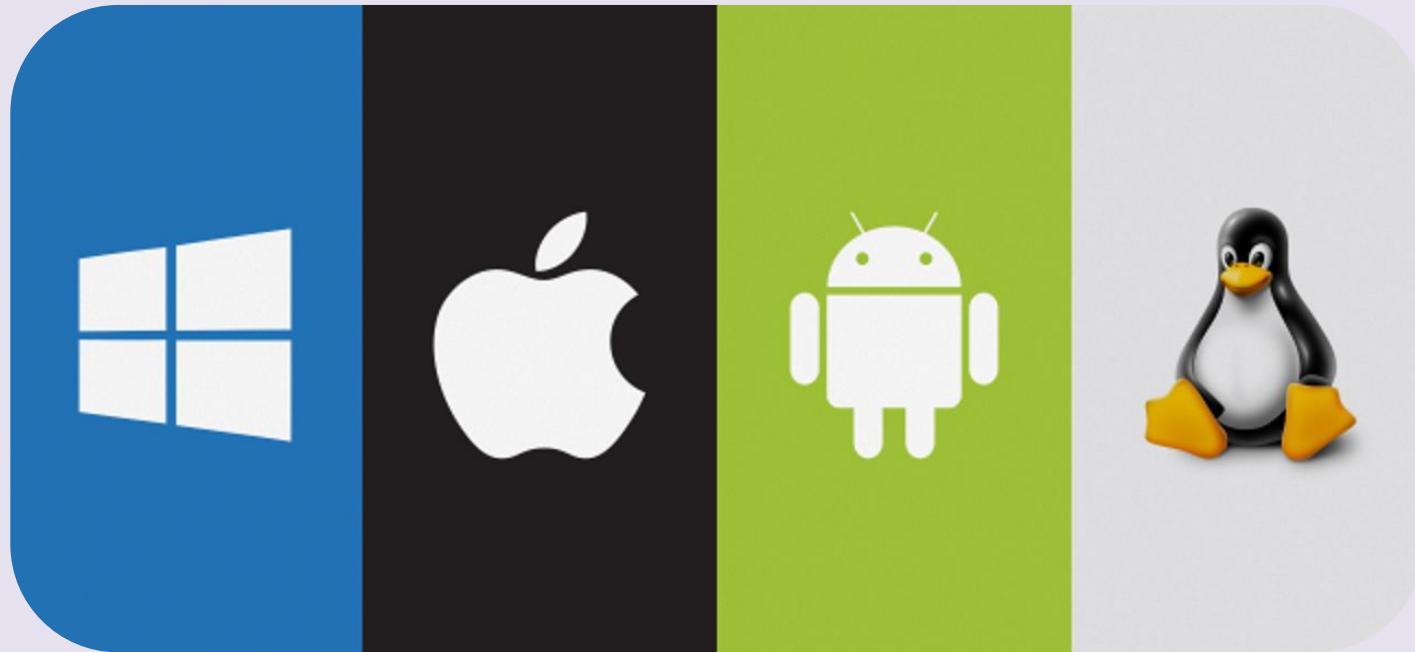


8 bits = 1 Byte





Sistemas Operativo





Sistemas Operativo

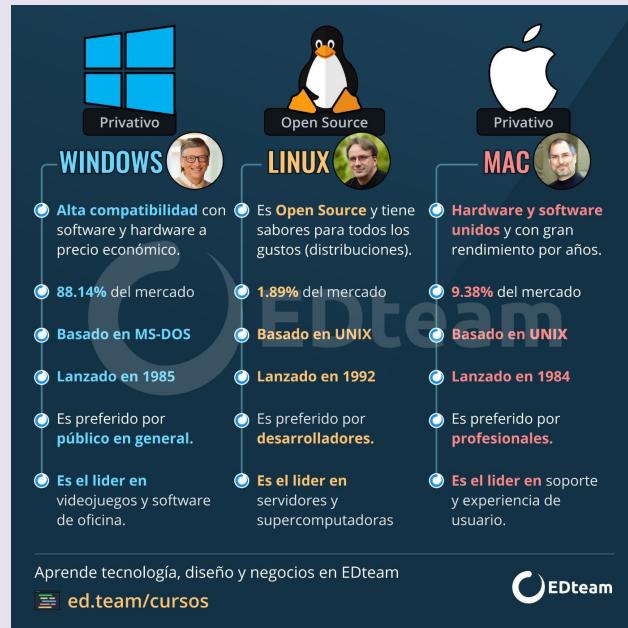


Sistemas Operativo

Existen 3 sistemas operativos principales para computadoras:

- Windows
- Mac
- Linux

Android y IOS también se basan en unix. Se puede decir que los SO de los teléfonos son variaciones del SO de las computadoras





Curiosidades

¿Existen virus para MAC?

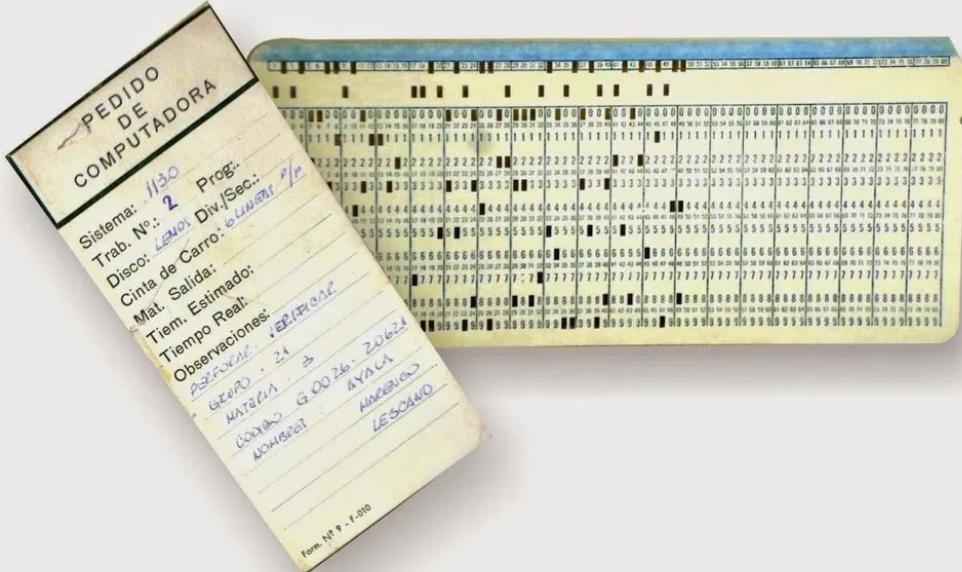
¿Tiene antivirus la MAC?

¿Existen virus para un Iphone?





Programas antiguos



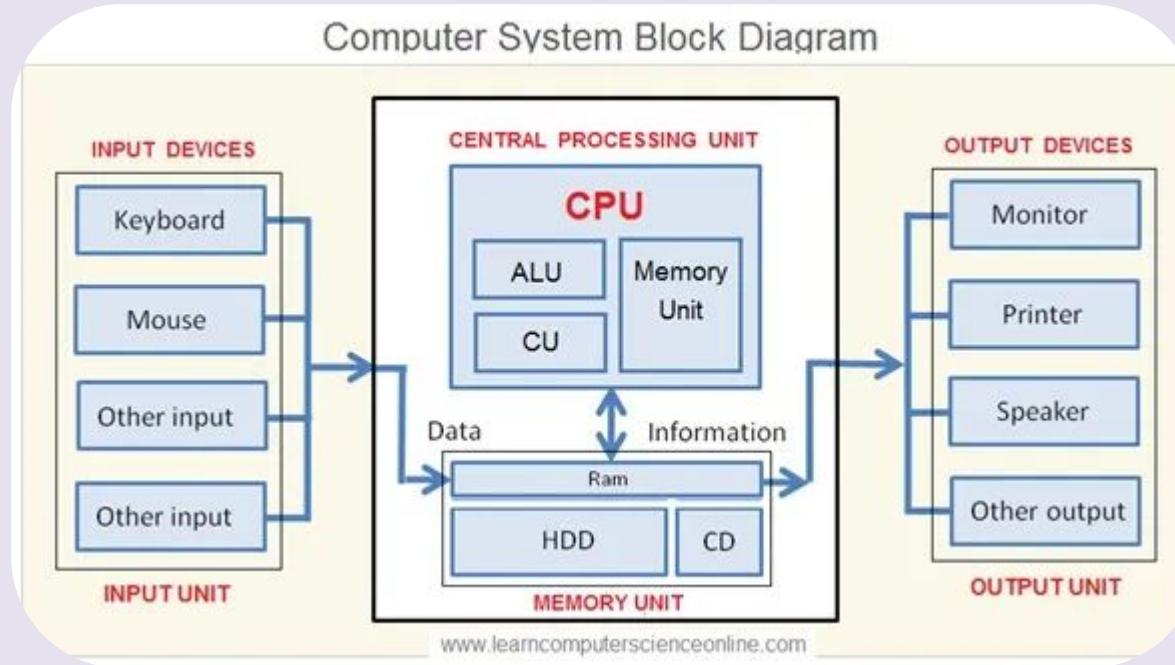


Programas antiguos

El lenguaje ensamblador es el lenguaje de programación utilizado para escribir **programas informáticos de bajo nivel**, y constituye la **representación más directa del Código máquina** específico para cada **arquitectura de computadoras** legible por un programador.

```
x000000000]> pd
    0x00000000  90      nop
    0x00000001  90      nop
    0x00000002  6800009c00 push 0x9c0000 ; 0x009c0000
    0x00000007  e8c7ace37b call  0x7be3acd3
        0x7be3acd3(unk)
    0x0000000c  bb04009c00 mov   ebx, 0x9c0004
    0x00000011  8903     mov   [ebx], eax
    0x00000013  e81903f47b call  0x7bf40331
        0x7bf40331()
    0x00000018  bb08009c00 mov   ebx, 0x9c0008
    0x0000001d  8903     mov   [ebx], eax
    0x0000001f  bb00009c00 mov   ebx, 0x9c0000
    0x00000024  c60300   mov   byte [ebx], 0x0
-> 0x00000027  68e8030000 push 0x3e8 ; 0x000003e8
    0x0000002c  e81124e37b call  0x7be32442
        0x7be32442(unk)
=< 0x00000031  ebf4     jmp   0x100000027
    0x00000033  90      nop
    0x00000034  ff      invalid
    0x00000035  ff      invalid
    0x00000036  ff      invalid
    0x00000037  ff      invalid
```

→ Programas antiguos



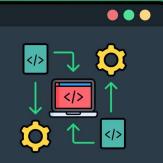
→ Paradigmas de la programación

Un paradigma de programación es una **manera** o **estilo de programación de software**. Existen diferentes formas de **diseñar un lenguaje** de programación y varios modos de trabajar para obtener los resultados que necesitan los programadores. Se trata de un conjunto de **métodos sistemáticos aplicables** en todos los niveles del diseño de programas para resolver problemas computacionales.

Python o JavaScript, que son **multiparadigmas**.

¿QUÉ SON LOS PARADIGMAS DE PROGRAMACIÓN?

Los paradigmas son los diferentes estilos de usar la programación para resolver un problema.



PROGRAMACIÓN ESTRUCTURADA

Programación secuencial con la que todos aprendemos a programar. Usa ciclos y condicionales.



PROGRAMACIÓN REACTIVA

Observa flujos de datos asíncronos y reacciona frente a sus cambios.



PROGRAMACIÓN ORIENTADA A OBJETOS

Divide los componentes del programa en objetos que tienen datos y comportamiento y se comunican entre sí.



PROGRAMACIÓN FUNCIONAL

Divide el programa en tareas pequeñas que son ejecutadas por funciones.



Aprende a programar en cualquier lenguaje (primer curso gratis) en:

ed.team/cursos/paradigmas



→ Paradigmas de la programación

Permite **separar los diferentes componentes** de un programa, simplificando así su creación, depuración y posteriores mejoras. La programación orientada a objetos **disminuye los errores** y promociona la **reutilización del código**. Es una manera especial de programar, que se acerca de alguna manera a cómo expresaríamos las cosas en la vida real.

Ejemplos de lenguajes de programación orientados a objetos serían Java, Python o C#.





Internet



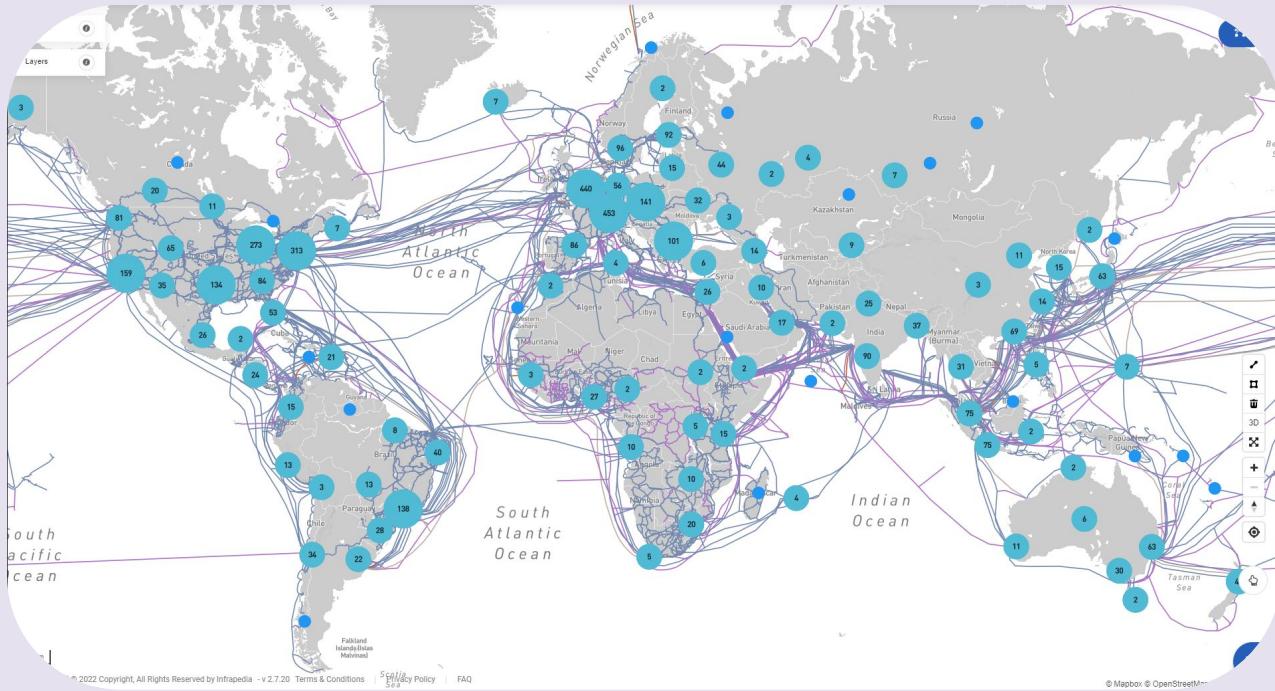
Internet

Red informática mundial, **descentralizada**, formada por la **conexión directa entre computadoras** mediante un protocolo especial de comunicación. -
RAE





Vista de la internet



<https://www.infrapedia.com/>



Origen de la Internet

Existen 2 versiones:

1. La más popular señala su creación como una respuesta del Departamento de Defensa estadounidense, quienes en los años 60 buscaban la forma en la que todos los ordenadores que se utilizaban dentro de la organización funcionarán en red, aún y cuando una de las computadoras sufriera una falla debido a un ataque enemigo.
2. La Oficina para Tecnologías de Procesado de la Información (IPTO), un hombre llamado Robert Taylor (quien se estrenaba como director de la oficina) tuvo la idea de generar un sistema que permitiera que los investigadores pudieran compartir recursos a través de la utilización de enlaces.

Internet

En 1969 el departamento de defensa de los Estados Unidos comenzó a buscar alternativas de comunicación ante una eventual guerra atómica. Tres años después se realizó la primera demostración entre 4 universidades (3 en California y 1 en Utah) al establecer una conexión conocida como ARPANET (Advanced Research Projects Agency Network).

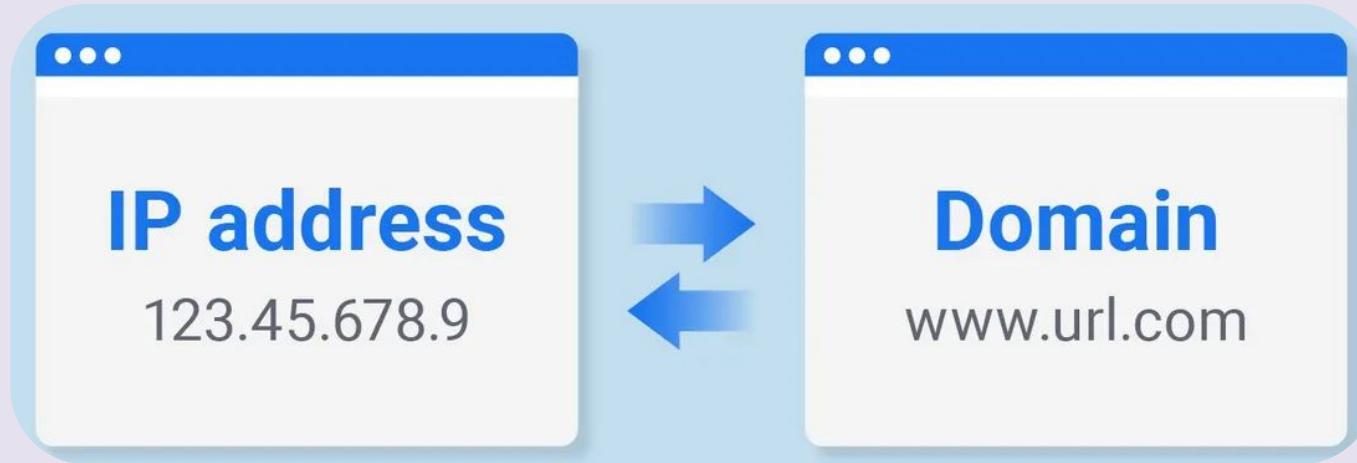




IP - Internet Protocol



IP - Internet Protocol



```
C:\Users\ANDREI>ping google.com
```

```
Pinging google.com [142.250.65.110] with 32 bytes of data:  
Reply from 142.250.65.110: bytes=32 time=11ms TTL=118
```

```
142.250.65.110 = Google
```

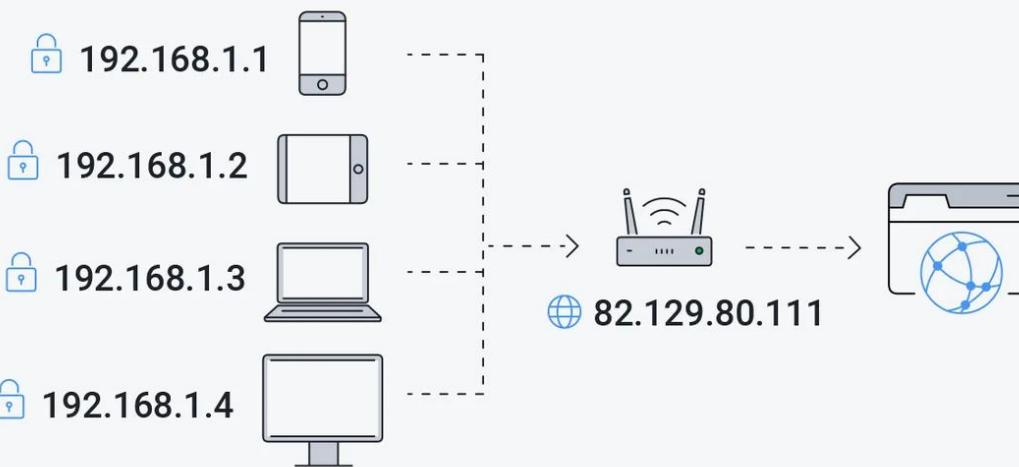


IP - Internet Protocol





IP - Internet Protocol

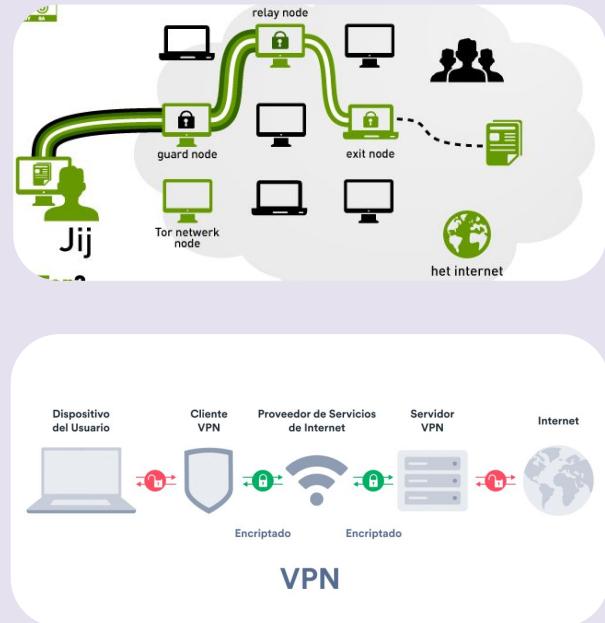


Public IP vs Private IP

Public IP	Private IP
Used over the Public Network ex. WAN	Used with in the private network ex. LAN
Recognized over the Internet	Not recognized over the Internet
Public IP are unique over the Globe.	Private IP are unique with in the network or LAN.
Public IP are paid.	Private IP are free of cost.
Assigned by Network Administrator.	Assigned by Internet Service Provider /IANA
Class A- 10.0.0.0 to 10.255.255.255 Class B- 172.16.0.0 to 172.31.255.255 Class C- 192.168.0.0 to 192.168.255.255	RANGE-- Class A- 10.0.0 to 9.255.255.255 11.0.0.0 to 126.255.255.255 Class B- 128.0.0.0 to 172.15.255.255 172.32.0.0 to 191.255.255.255 Class C- 192.0.0.0 to 192.167.255.355 192.169.0.0 to 223.255.255.255



Curiosidad



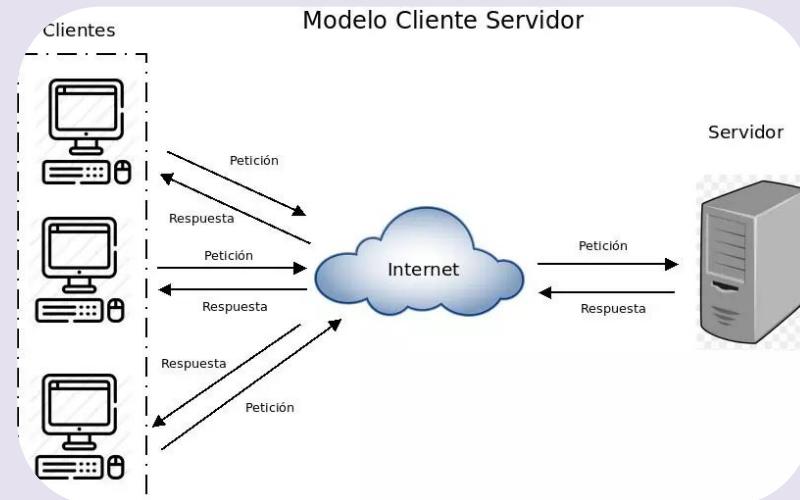


Cliente Servidor

→ Cliente Servidor

El modelo cliente-servidor es una **estructura de aplicación** distribuida que **divide tareas o cargas de trabajo** entre los **proveedores de un recurso** o servicio, denominados servidores, y los **solicitantes del servicio**, denominados clientes.

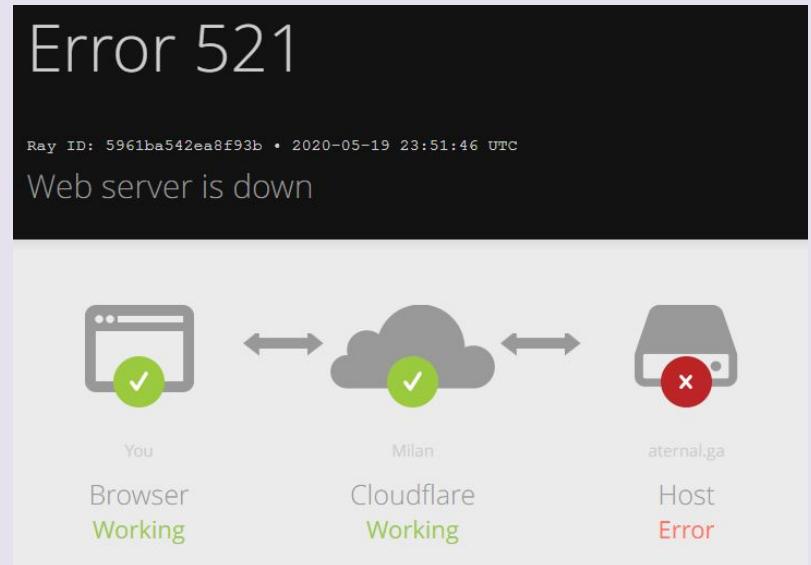
Es decir, el cliente **realiza peticiones** de servicios al servidor, que se encarga de **satisfacer las peticiones**.





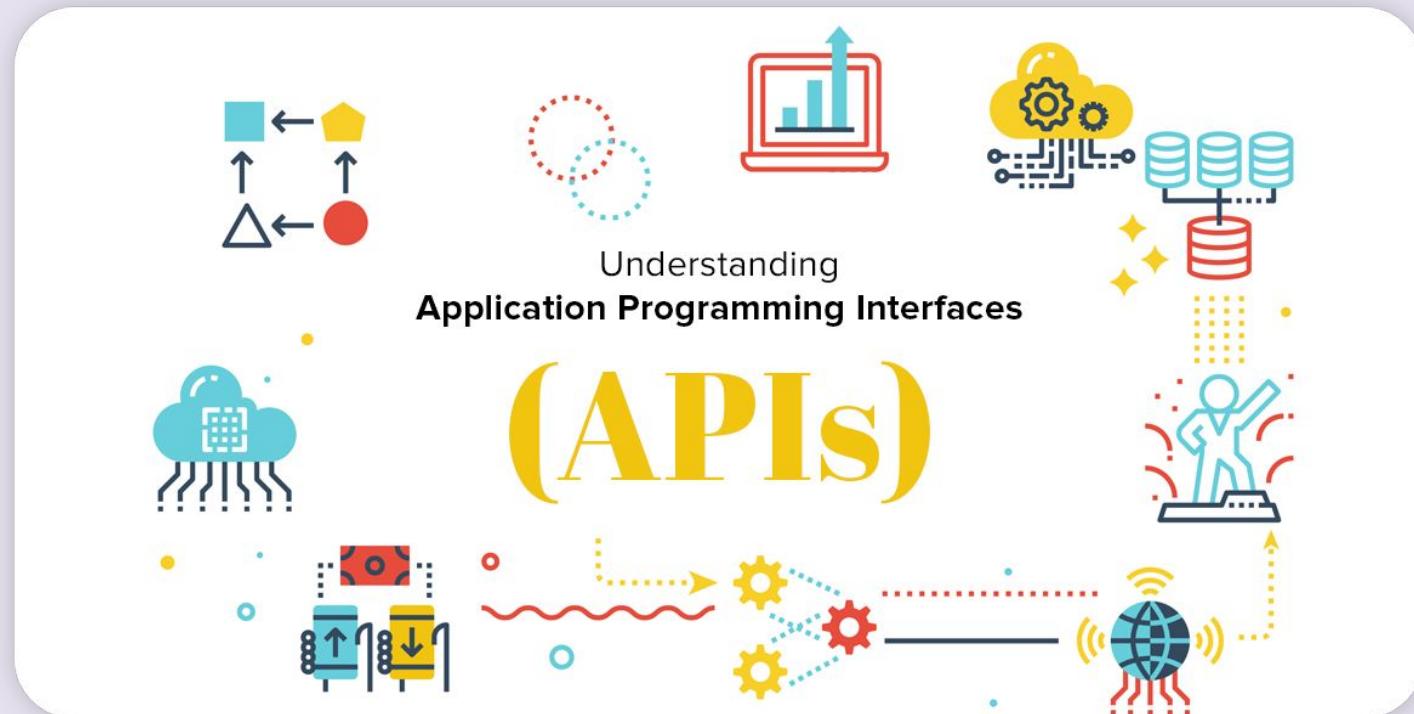
Curiosidades

- ¿Qué pasa si el servidor se cae?
- ¿Cómo se soluciona?
- ¿Existe algo más?





API - Application Program Interface



→ API - Application Program Interface



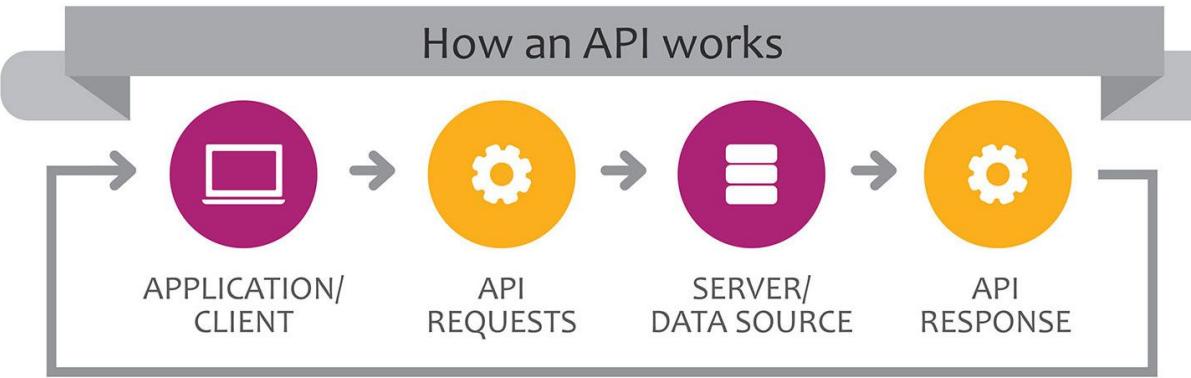
The API

Application Programming Interface

 API Definition

An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data.

How an API works



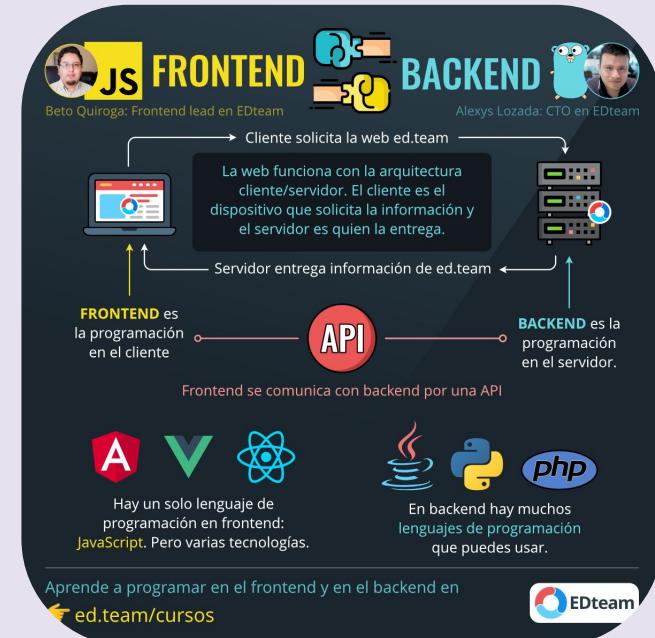
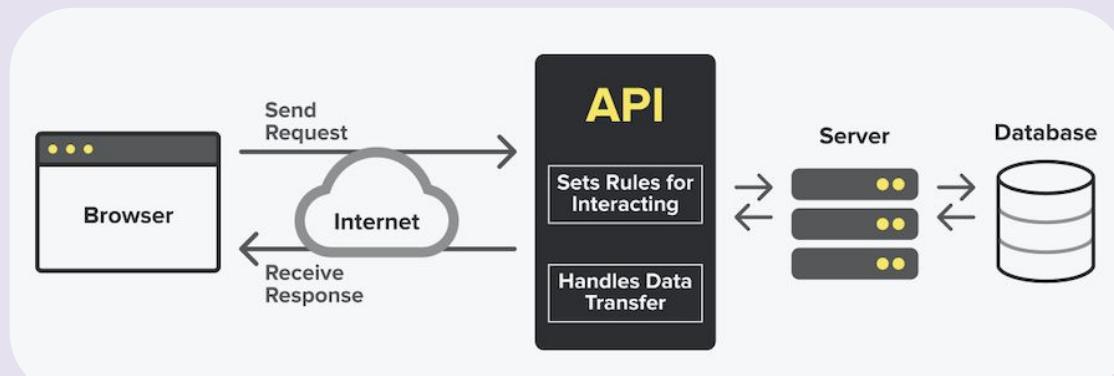
APPLICATION/
CLIENT

API
REQUESTS

SERVER/
DATA SOURCE

API
RESPONSE

→ API - Application Program Interface





Linux





Linux

Es un **sistema operativo** (o una familia de sistemas operativos) tipo Unix compuesto por **software libre y de código abierto**. GNU/Linux surge de las contribuciones de varios proyectos de software, entre los cuales destacan **GNU** (iniciado por Richard Stallman en 1983) y el **kernel «Linux»** (iniciado por Linus Torvalds en 1991).

Su capacidad para su **personalización** y poder de ser realmente **modificable** para las exigencias únicas de software y aplicaciones no tiene par. Tiene como beneficios **estabilidad, manejo súper eficiente de recursos de memoria, procesadores y almacenamiento**; sus diferentes tipos de **distribuciones** para cada escenario lo hacen un elemento imprescindible para albergar las más exigentes y peculiares aplicaciones o plataformas de software.

DISTRIBUCIONES LINUX MÁS USADAS

EMPRESAS



redhat

Red Hat Enterprise Linux (RHEL) es la distro empresarial más importante (no es gratis).



CentOS

Community ENTerprise Operating System. Es un fork gratuito de Red Hat.



SUSE

Suse ofrece soluciones de servidores, desktop y sistemas embebidos para empresas.



ubuntu

Basada en Debian, es la distro más famosa de Linux.

USUARIO FINAL



KALI

Basada en Debian, trae herramientas para auditoría y seguridad informática.



fedora

Distro para usuario final basada en Red Hat, pero desarrollada por la comunidad.



Linux Mint

Basada en Ubuntu, intenta dar una experiencia cercana a Windows.

Domina la administración de servidores Linux en EDteam:



ed.team/cursos/linux





Linux VS Windows Server

	LINUX SERVER	WINDOWS SERVER
ARCHITECTURE	centered around the Linux kernel	based on the Windows NT architecture
COST	free, open-source software	owned by Microsoft, includes a licensing fee per user
SECURITY	highly secure against malware and cyber threats	more prone to hacking attempts and cyber threats
SUPPORT	large community supports that can answer commonly asked questions	community and long-term customer support, along with great documentation
MODE OF OPERATION	command line	graphical user interface
USER EXPERIENCE	requires an relatively experienced Linux administrator	more beginner-friendly
DATABASE SUPPORT	MySQL, PostgreSQL	Microsoft SQL, Microsoft Access
SCRIPT SUPPORT	Python, PHP, Perl , and other Unix languages	ASP and ASP.NET



Componentes de computadora



→ Componentes de computadora

CPU: Unidad de procesamiento principal

RAM: Memoria de acceso aleatorio

Disco duro

Fuente de poder: Fuente de alimentación

Motherboard: Tarjeta madre

Gabinete

Es decir,

CPU: Hace todo el procesamiento

RAM: Mantiene los programas abiertos

Disco duro: Guarda los archivos

Fuente de poder: Proporciona la electricidad

Motherboard: Conecta todos los componentes

Gabinete: Contiene todos los componentes





Tipos de uso

- Oftálmico

Uso muy básico como para oficinas, consultorios, etc. Sólo requiere abrir pocas páginas de internet y uno que otro programa

- Estudiantes (Primaria - Preparatoria)

Uso básico como para ejecutar office, abrir varias páginas web, realizar tareas y abrir varios programas

- Trabajo

Uso moderado como para abrir varios programas, guardar muchos documentos y usar múltiples plataformas

- Juegos

Uso intensivo para correr juegos de video que requieren de procesamiento en general y de gráfico

- Cómputo intensivo

Uso intensivo para realizar Aprendizaje Profundo, operaciones complejas matemáticas o que requiere mucho procesamiento o procesamiento de video, parecido a los de juegos



Tipos de uso

Procesadores	RAM	Almacenamiento
<ul style="list-style-type: none">• Intel<ul style="list-style-type: none">◦ i3◦ i5◦ i7◦ i9• AMD<ul style="list-style-type: none">◦ Ryzen 3◦ Ryzen 5◦ Ryzen 7◦ Ryzen 9	<ul style="list-style-type: none">• 2GB• 4GB• 8GB• 16GB• 32GB• 64GB• 128GB• 256GB• 512GB• 1024GB – 1TB• ...	<ul style="list-style-type: none">• 32GB• 64GB• 128GB• 256GB• 512GB• 1024GB – 1TB• 2TB• ...



Recomendaciones (2023)

- Oftálmico

Intel i3 - Ryzen 3, 4GB a 8GB de RAM, 128GB

- Estudiantes (Primaria - Preparatoria)

Intel i3 a i5 - Ryzen 3 a Ryzen 5, 8GB a 16GB de RAM, 256GB a 512GB

- Trabajo

Intel i5 a i7 - Ryzen 5 a Ryzen 7, 8GB a 32GB de RAM, 512GB a 1TB

- Juegos

Intel i5 a i9 - Ryzen 5 a Ryzen 9, 16GB a 64GB de RAM, 512GB a 2TB, Tarjeta de video

- Cómputo intensivo

Intel i7 a i9 - Ryzen 7 a Ryzen 9, 16GB+ de RAM, 1TB+, Tarjeta de video (Opcional)



Curiosidades

- Intel - Actual: 13 Generaciones | Comprar mínimo gen 10ma
- AMD - Actual: Serie 7000 | Comprar mínimo series 5

Ejemplos

- i7 13700K → iX = #Procesador | XX = Generación | XXX - Grado de procesador
- R7 7800X → RX = #Procesador | XX = Generación | XXX - Grado de procesador

NOTA:

Siempre el grado de procesador son de 3 dígitos y termina con una o dos letras



03

Desarrollo de

Software



¿Qué es?



¿Qué es?

El desarrollo de software se refiere a un conjunto de actividades informáticas dedicadas al **proceso de creación, diseño, despliegue y compatibilidad de software**.

Hay tres tipos básicos:

- **Software del sistema** para proporcionar funciones básicas como sistemas operativos, administración de discos, servicios, administración de hardware y otras necesidades operacionales.
- **Software de programación** para brindar a los programadores herramientas como editores de texto, compiladores, enlazadores, depuradores y otras herramientas para crear código.
- **Software de aplicación (aplicaciones o apps)** para ayudar a los usuarios a realizar tareas. Las suites de productividad de Office, el software de gestión de datos, los reproductores multimedia y los programas de seguridad son algunos ejemplos. Aplicaciones también se refiere a aplicaciones web y móviles





¿Qué es?

- **Los programadores, o codificadores**, escriben el código fuente para programar computadoras para realizar tareas específicas como fusionar bases de datos, procesar pedidos en línea, enrutar comunicaciones, realizar búsquedas o mostrar texto y gráficos. Los programadores suelen **interpretar las instrucciones de los desarrolladores e ingenieros de software**.
- **Los ingenieros de software** aplican principios de **ingeniería para crear software y sistemas para resolver problemas**. Utilizan lenguaje de modelado y otras herramientas para idear soluciones que a menudo se pueden aplicar a problemas de manera general, en lugar de simplemente resolver solo una instancia o un cliente específico. Las soluciones de ingeniería de software se adhieren al **método científico** y deben funcionar en el mundo real. Su **responsabilidad ha aumentado** a medida que los **productos** se han vuelto cada vez **más inteligentes** con la adición de microprocesadores, sensores y software.
- **Los desarrolladores de software** tienen un rol menos formal que los ingenieros y pueden participar de cerca en áreas específicas del proyecto, incluida la escritura de código. Al mismo tiempo, **impulsan el ciclo de vida general del desarrollo de software** mediante el trabajo en equipos funcionales para **transformar los requisitos en funciones, la gestión de equipos y procesos de desarrollo y la realización de pruebas y mantenimiento de software**.



¿Por qué surge?

El Vicepresidente de IBM y bloguero Dibbe Edwards señala: "El software ha surgido como un **diferenciador fundamental en muchos productos**, desde automóviles hasta lavadoras y termostatos, con un creciente Internet de las cosas que los conecta".

Las empresas que usan el desarrollo de software, tienen **procesos más limpios, rápidos y efectivos** que las empresas que usan procesos totalmente manuales, ya que se **eliminan tiempos de respuesta, tiempos muertos y el error humano**.





Pasos del proceso de desarrollo de software

1. Seleccionar una metodología
2. Recopilar requisitos
3. Elegir o crear una arquitectura
4. Desarrollar un diseño
5. Crear un modelo
6. Crear código
7. Realizar pruebas
8. Gestionar la configuración y los defectos
9. Desplegar
10. Migrar datos
11. Gestionar y medir el proyecto



Resumen de pasos:

1. Análisis y especificación de requisitos
2. Diseño y desarrollo
3. Pruebas
4. Despliegue
5. Mantenimiento y soporte

Metodologías desarrollo de software



04

Tecnologías aplicadas





Lenguaje de programación

Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un **lenguaje con reglas gramaticales bien definidas**) que le proporciona a una persona, en este caso el programador, la **capacidad de escribir (o programar)** una **serie de instrucciones o secuencias de órdenes en forma de algoritmos** con el fin de controlar el **comportamiento físico o lógico de un sistema informático**, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa informático.

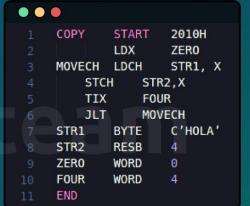
LENGUAJES DE PROGRAMACIÓN ALTO NIVEL VS BAJO NIVEL

 Es un lenguaje que entienden los humanos.

 Son instrucciones para el procesador.



```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     cout << "Hola EDteam" << endl;
7     return 0;
8 }
```



```
1 COPY    START    2010H
2        LDX      ZERO
3 MOVECH LDCH    STR1, X
4 STCH    STR2, X
5 TIX     FOUR
6 JLT    MOVECH
7 STR1   BYTE    C'HOLA'
8 STR2   RESB    4
9 ZERO   WORD    0
10 FOUR   WORD    4
11 END
```

- Está orientado al **software**.
- Utilizan **menos instrucciones** para realizar una acción.
- Te permite programar **aplicaciones y videojuegos**.
- Nos ayuda a entender **cómo funcionan las instrucciones** en la computadora.
- Puedes construir **sistemas operativos y núcleos**.

ed.team/programacion





Lenguaje de programación



TYPES OF LANGUAGES





Lenguaje de programación

LOS PRIMEROS LENGUAJES DE PROGRAMACIÓN



FORTRAN (1955)



Desarrollada para la **computación científica de alto nivel, matemáticas y estadísticas**. Se sigue usando en la industria automovilística, aeroespacial y gubernamental.



LISP (1959)

Diseñado para su uso en el campo de la **IA** y hasta hoy se puede usar con Python y Ruby.

BASIC (1964)



Desarrollado como una **herramienta de enseñanza** para aquellos que no tenían fuertes conocimientos técnicos y matemáticos.



PASCAL (1970)



Se desarrolló como una **herramienta de aprendizaje para la programación de ordenadores**. Era fácil de aprender y el favorito de Apple en sus inicios.



COBOL (1959)

Podía ejecutarse en todo tipo de ordenadores y **actualmente se usa en procesadores de tarjetas de crédito, cajeros automáticos, telefonía y señales de tráfico**.



C (1969)

Se le llamó "C" porque derivaba del lenguaje Basic. Es tan poderoso que **Linux todavía está basado en este lenguaje**.

10 LENGUAJES DE PROGRAMACIÓN CON MAYOR DEMANDA EN 2020

FUENTE: HIRED
hired.com



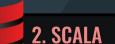
10. JAVA

ed.team/cursos/java



1. GO

ed.team/cursos/go



2. SCALA



9. PHP

ed.team/cursos/php



3. RUBY



8. SWIFT

ed.team/cursos/swift



4. TYPESCRIPT



5. KOTLIN



7. JAVASCRIPT

ed.team/cursos/javascript



6. OBJECTIVE-C

ed.team/cursos/kotlin

Tu próximo empleo te está esperando. Estudia en EDteam

ed.team/cursos

Lenguaje de programación



Tipos de tipado

Un lenguaje de programación tipados y no tipados:

- Los **lenguajes tipados** son aquellos que **forzosamente** se tiene que **definir el tipo de dato o variable**. C#, C, C++ y Java son algunos ejemplos.
- Los **lenguajes no tipados** son aquellos **no necesitan alguna definición** y toman el tipo del valor que se les es asignado. JavaScript, Python y PHP son algunos ejemplos.

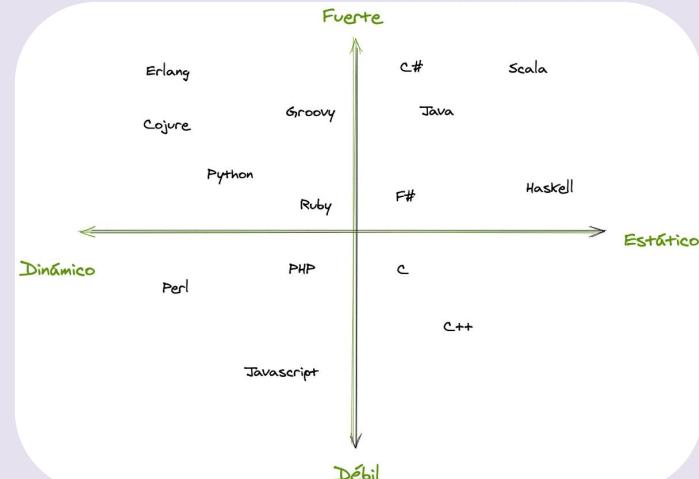
Tipado = Estático | No Tipado = Dinámico

Tipado = Fuerte | No Tipado = Débil



Tipos de tipado

- El **tipado débil** te permite trabajar en **menos tiempo** y comprobar cuáles son los resultados del programa en tiempo real. Para la **fase de ideación** del proyecto esta opción es ideal. Se adapta muy bien a programas de menor capacidad.
- El **tipado fuerte** admite realizar operaciones con distintos tipos de variable. La consecuencia directa es que **no vas a cometer tantos errores** como con el lenguaje del punto anterior. Ahora bien, necesitas **escribir más código**. Este lenguaje es ideal para **proyectos de mayor alcance** y con un **número mayor de especificaciones**.





Paradigmas

Un paradigma es como un mapa en el que existen muchos caminos para llegar a un mismo fin. Hay lenguajes que adoptan un paradigma específico y hay otros que adoptan más de un paradigma como JavaScript, Python y PHP.



POO - Programación Orientada a Objetos

Permite **separar los diferentes componentes** de un programa, simplificando así su creación, depuración y posteriores mejoras. La programación orientada a objetos **disminuye los errores** y promociona la **reutilización del código**. Es una manera especial de programar, que se acerca de alguna manera a cómo expresaríamos las cosas en la vida real.

Ejemplos de lenguajes de programación orientados a objetos serían Java, Python o C#.





POO - Programación Orientada a Objetos

OOP

Más fácil de mantener

No te repitas (DRY)

Pequeños trozos de código reutilizados en muchos lugares

Enfoque por objetos

Más fácil de depurar

Gran curva de aprendizaje

Utilizado en grandes proyectos

Programación Estructurada

Difícil de mantener

Código repetido en muchos lugares

Una gran cantidad de código en pocos lugares

Enfoque de código de bloques

Más difícil de depurar

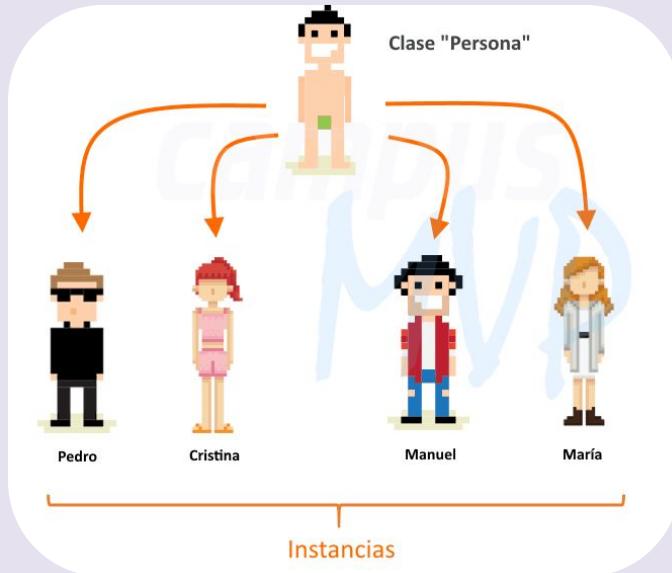
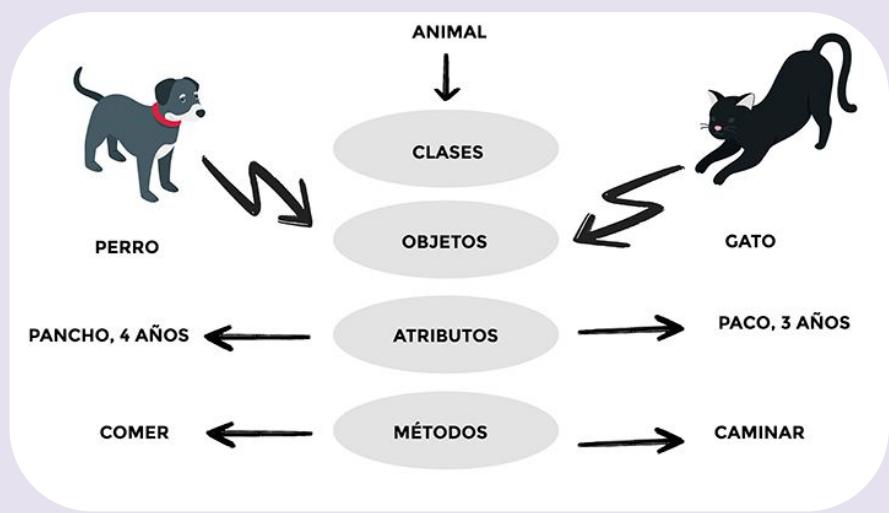
Una curva de aprendizaje más sencilla

Optimizado para programas sencillos





POO - Programación Orientada a Objetos



Tipos de desarrollos

RAMAS DE LA PROGRAMACIÓN

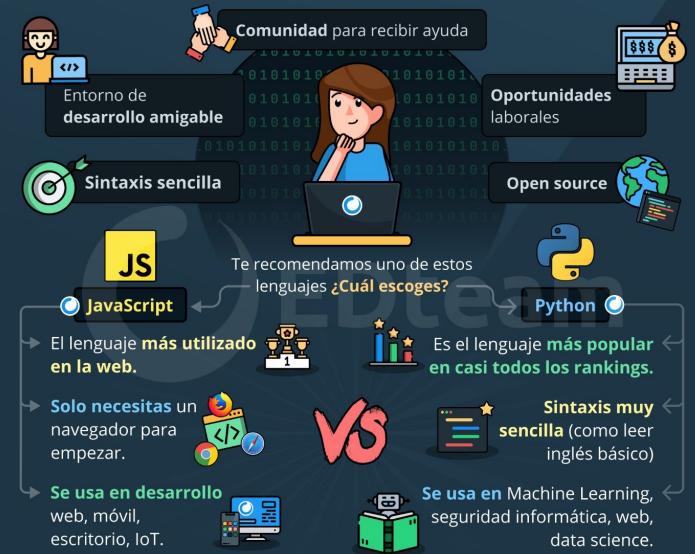


Aprende a programar en cualquier lenguaje (primer curso gratis) en:

ed.team/programacion



¿CON QUÉ LENGUAJE APRENDER A PROGRAMAR?



Elije uno de estos lenguajes y apréndelo en EDteam

ed.team/cursos

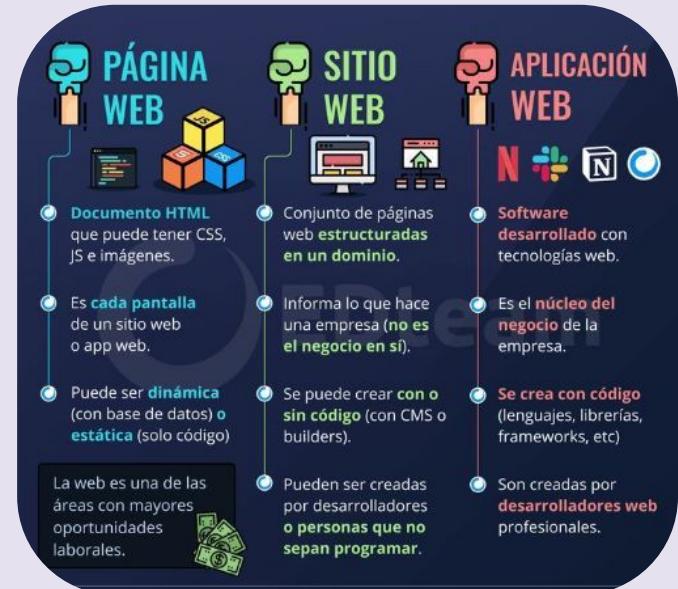




Desarrollo Web

Desarrollo web es un término que define la **creación de sitios web** para **Internet o una intranet**. Para conseguirlo se hace uso de tecnologías de software del **lado del servidor** y del cliente que involucran una combinación de procesos de base de datos con el uso de un **navegador web** a fin de realizar determinadas tareas o mostrar información.

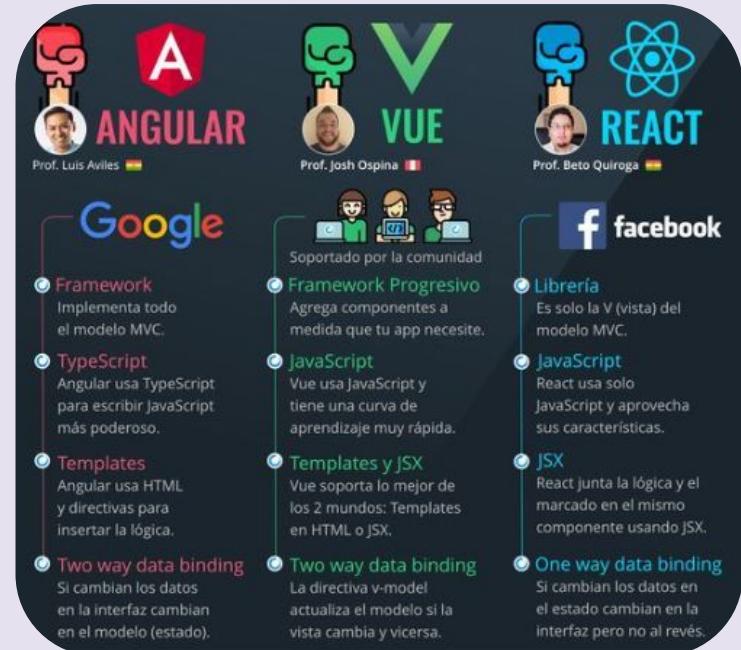
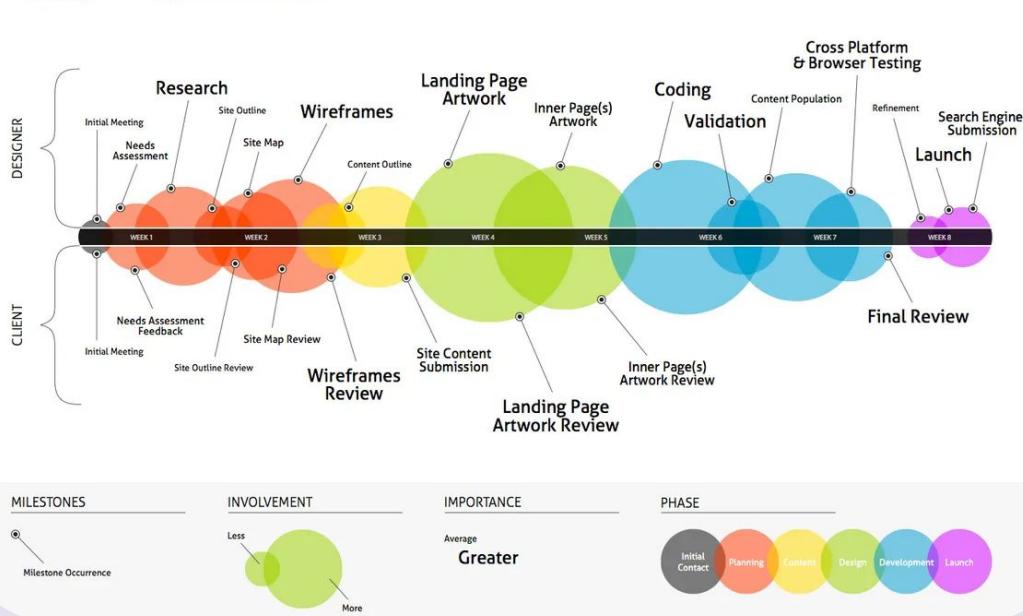
Funcionalmente, el **desarrollador web**, que es quien realiza esta labor, normalmente sólo se preocupa por el **funcionamiento del software**, es tarea del diseñador web preocuparse del aspecto final(layout) de la página y del webmaster el integrar ambas partes.



Desarrollo Web

A Web Site Designed

MILESTONES, INVOLVEMENT, IMPORTANCE & TIMELINE

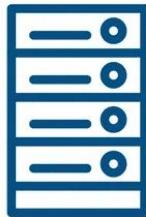


Desarrollo Web



Front End

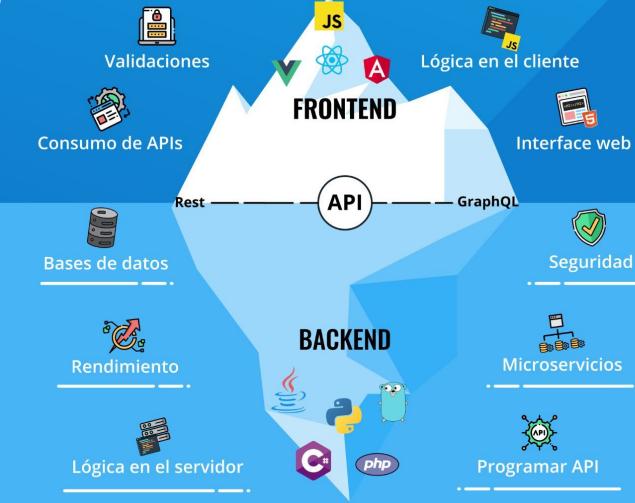
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



Back End

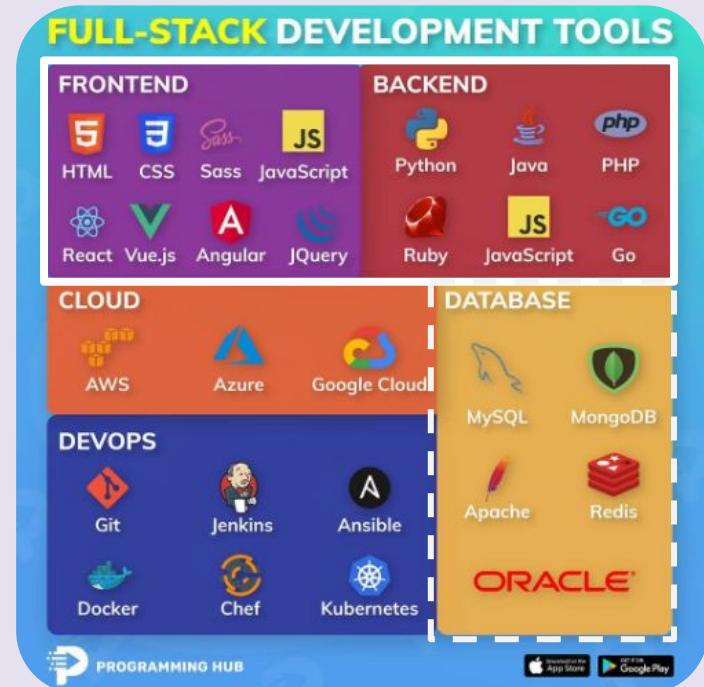
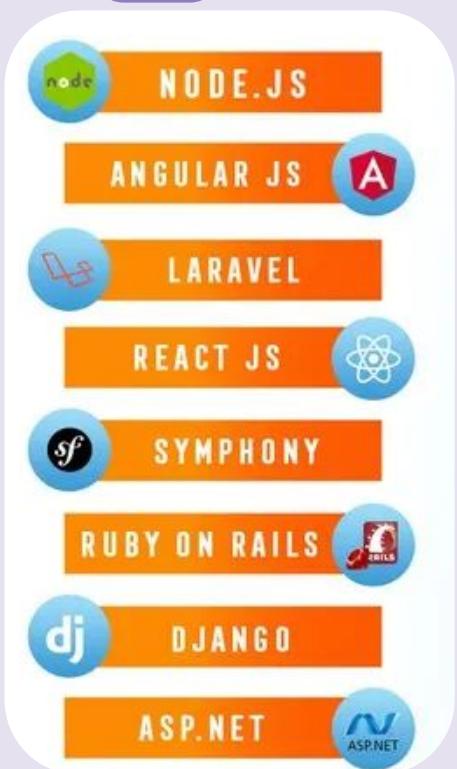
- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

¿QUÉ ES BACKEND Y FRONTEND?



Domina las tecnologías Backend y Frontend en:
ed.team/cursos

Desarrollo Web





Desarrollo de Escritorio

Las aplicaciones de escritorio son el **conjunto de programas o herramientas que tenemos instalados en nuestros ordenadores** de sobremesa o portátiles y que **únicamente podemos usar en dichos dispositivos**. No los podemos trasladar a otro distinto, por lo que posee esta limitación.

Al contrario sucede con las aplicaciones web, que al tener su presencia en la nube, no tienen ese problema de portabilidad.

Ejemplos de aplicaciones de escritorio podrían ser OpenOffice, Excel o Photoshop.





Desarrollo de Escritorio

Entre las **ventajas** más destacables de las aplicaciones de escritorio se encuentran:

- Los datos están centralizados.
- Son programas mucho más estables y robustos.
- La carga de datos se produce con una mayor rapidez.
- Si se te cae Internet, no te preocupes porque vas a poder usarla igual.
- Suelen ser más económicas.
- La seguridad es mayor en estas aplicaciones.
- Puedes hacer copias de seguridad en todo momento.



→ Desarrollo de Escritorio

Empezando por las **desventajas** o defectos que se le pueden poner a las aplicaciones de escritorio se encuentran:

- Requiere de una instalación en cada cliente o dispositivo.
- Al fin y al cabo se desarrollan para un sistema operativo específico. Si tienes otro sistema diferente, tienes que esperar a que desarrollen la versión para el tuyo.
- Se requiere actualizar en cada dispositivo. No puedes meter una actualización para todo el mundo que disponga de esa aplicación.





Desarrollo de Escritorio

Los lenguajes más conocidos y usados son Java y Python, pero como vamos a ver existen muchos más.

- En Windows tenemos Visual C++ y Visual Basic.
- C/C++ co Qt o GTK.
- Java con AWT o Swing.
- En Mac tenemos Objetive-C/Swift con Cocoa.
- En Linux tenemos C/C++ con Qt o GTK.





Desarrollo para móviles

El desarrollo de aplicaciones móviles es el conjunto de procesos y procedimientos involucrados en la escritura de software para dispositivos informáticos **pequeños e inalámbricos**, como **teléfonos inteligentes** y otros dispositivos portátiles.

Las aplicaciones móviles a menudo **se desarrollan específicamente para aprovechar las características únicas de un dispositivo móvil en particular**. Por ejemplo, se podría escribir una aplicación de juegos para aprovechar el acelerómetro del iPhone o se podría escribir una aplicación de salud móvil para aprovechar el sensor de temperatura de un reloj inteligente.





Desarrollo para móviles

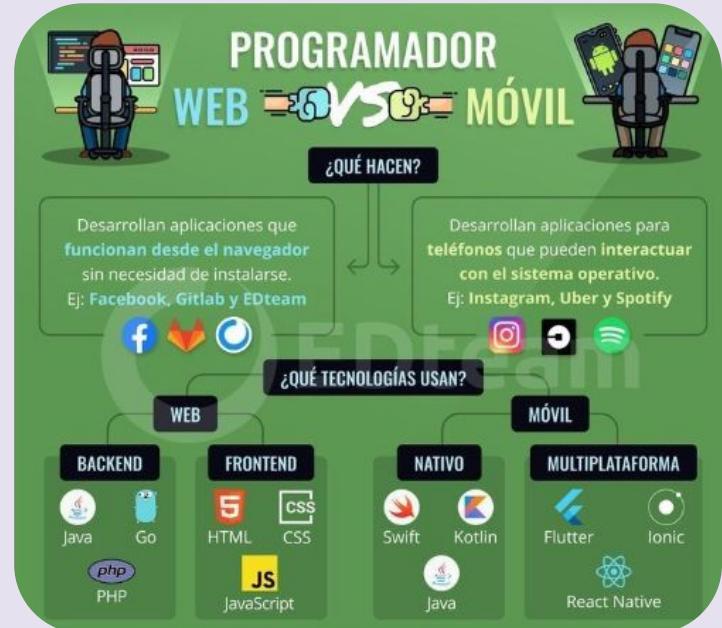
- **Aplicaciones nativas.** Estas aplicaciones se crean utilizando entornos de desarrollo integrados (IDE) e idiomas para sistemas operativos móviles como **Apple iOS o Google Android**. Las aplicaciones nativas le permiten personalizar las funciones necesarias, pero pueden ser más costosas que otras tecnologías.
- **Aplicaciones híbridas.** Estas son **aplicaciones web** que actúan como aplicaciones nativas. Se desarrollan utilizando tecnologías como **HTML, JavaScript** y Hojas de estilo en cascada (**CSS**). Las aplicaciones híbridas son **más rentables de desarrollar que las aplicaciones nativas** y se pueden **crear más rápido**, pero **no son tan ricas en funciones** como las aplicaciones nativas y no se instalan en el dispositivo.
- **Aplicaciones web progresivas.** Una PWA es un sitio web que se ve y se **comporta como si fuera una aplicación móvil**. Estas aplicaciones se desarrollan con tecnologías web como React.
- **Aplicaciones encapsuladas.** Una aplicación encapsulada se ejecuta dentro de una aplicación de contenedor. Productos como la herramienta de creación de aplicaciones de arrastrar y soltar **Microsoft Power App** permiten a los desarrolladores menos experimentados **crear una aplicación móvil rápidamente**. Pero la falta de aislamiento del sistema operativo central, el bloqueo del sistema operativo y la novedad relativa podrían plantear problemas.



Desarrollo para móviles

Una **app móvil** se refiere a una **aplicación nativa**, creada para determinados smartphones, que se puede descargar desde la correspondiente store (cada marca de smartphone tiene la suya), y se **instala en el disco duro** del dispositivo.

Una **web móvil** es un **desarrollo web diseñado para imitar a una app móvil**, pero se accede mediante un navegador web. Es un **acceso directo a la página web** pero el aspecto que ofrece está adaptado a nuestro dispositivo móvil.



Desarrollo para móviles





Desarrollo para multiplataforma

El desarrollo multiplataforma consiste en manera general en poder programar para múltiples dispositivos, puede ser mediante un script, aplicativo web o un API con el objetivo de ser compatible con varios dispositivos.

Es la manera más general de un desarrollador cuando no está realizando un programa a un dispositivo en específico.



05

Roles

Y sus actividades



→ Roles en Desarrollo

Se necesitan más que desarrolladores e ingenieros para formar un equipo de desarrollo de software eficaz. Se necesitan muchos roles en el ciclo de vida del desarrollo de software, y el equipo ideal está formado por los siguientes roles:

- Dueño del producto (Product owner)
- Project manager
- Diseñadores UX y UI (UX/UI)
- Analista comercial
- Desarrolladores de software
 - Programadores
- Líder de equipo y de tecnología (team lead o tech lead)
- Scrum master
- Desarrolladores Full-stack
- Desarrolladores frontales (Front End)
- Desarrolladores de back-end



→ Roles en Desarrollo

10 CARGOS TI MÁS SOLICITADOS

JEFE DE PROYECTO

Renta promedio: \$1.500.000



DESAROLLADORES

Renta promedio: \$1.300.000



INGENIERO EN INFORMÁTICA

Renta promedio: \$1.100.000



INGENIERO DE SOFTWARE

Renta promedio: \$1.000.000



CIBERSEGURIDAD

Renta promedio: \$1.300.000

ANALISTA PROGRAMADOR

Renta promedio: \$1.000.000

ANALISTA DE SISTEMAS

Renta promedio: \$1.200.000

ETHICAL HACKER

Renta promedio: \$1.500.000

JEFE DE ÁREA TI

Renta promedio: \$1.500.000

SOPORTE TÉCNICO TI

Renta promedio: \$800.000



ATCOM
WWW.ATCOM.CL

AGILE IT PROJECT MANAGEMENT



ARTIFICIAL INTELLIGENCE



BUSINESS ANALYSIS



CLOUD COMPUTING



CYBERSECURITY



DATA ANALYTICS



SOFTWARE ENGINEERING



TECHNOLOGY, INFORMATION & CYBERSECURITY RISK





¿Cómo elegir un rol?

Para elegir un rol se deben considerar los siguientes:

- Conocimiento
- Herramientas
- Gustos
- Experiencia
- Habilidades





Rangos de roles

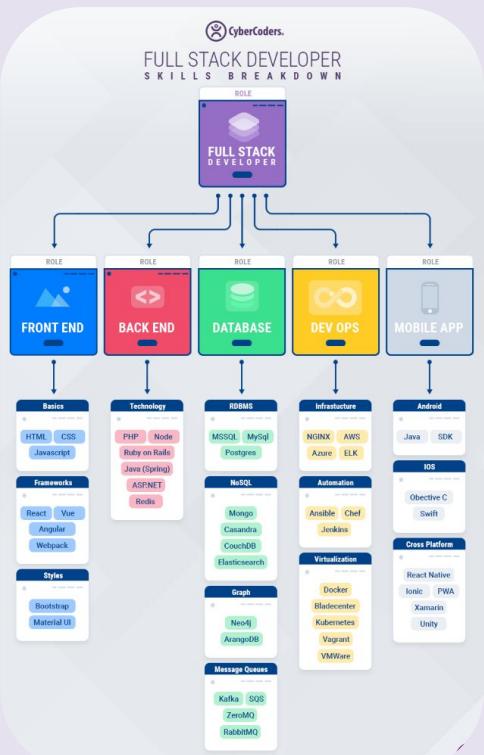




Rangos de roles



→ Roles en la industria



- 1 COMPUTER CODING**
Computer coding is vital for the increasing use of digital technology. As technology progresses rapidly, many programming languages are being simplified, making them easier for humans to understand. A good grasp of computer coding skill will be sought after for a long time to come.
- 2 BUSINESS INTELLIGENCE**
A good percent of IT hiring decision-makers need Business Intelligence (BI) and data analytics. Using specific BI tools and data-related programming languages to sift data, this skill requires deep, broad experience across database technologies with an emphasis on analytical and reporting tools.
- 3 ARTIFICIAL INTELLIGENCE**
To support the new and innovative Internet of Things (IoT) products and services, Artificial Intelligence (AI) skill is in serious demand and the field is understood. Due to the significant growth in this field, companies are investing heavily in the skill.
- 4 DATA SCIENCE**
Related to BI skill, the understanding of data science also helps businesses see a return on investment in big data analysis. Data scientists are responsible for the database technology and design used to store any kind of data. These tech professionals typically have a PhD or a master's degree in Math.
- 5 CYBERSECURITY**
As cyber threats continue to upsurge in scope and sophistication, the need for Cybersecurity will continue to be a priority across all industries. Consequently, many companies are increasing spending on cybersecurity skill and technologies to prevent the exploitation of hackers.
- 6 CLOUD COMPUTING**
Due to the noticeable growth and the adoption of Cloud services in Amazon Web Services (AWS), Azure and Google Cloud, there is a demand for professionals with a working knowledge of these platforms.
- 7 Data Architect**
Data architects are responsible for overseeing the maintenance and conception of a range of databases and networks.
Median annual salary: £80,000
- 8 Solutions Architect**
A solutions architect is responsible for developing and maintaining technical architecture. They work on a client's behalf to identify their technological goals and requirements, and to create a plan that will allow them to meet those requirements.
Median annual salary: £77,500
- 9 Machine Learning/AI Engineer**
Image recognition, natural language processing and economic forecasting are just a few of the sub-fields of machine learning that involve writing codes in training data. The demand for Artificial Intelligence/Machine Learning Engineers is growing as the emerging field of automation becomes the focus of the tech industry.
Median annual salary: £76,250
- 10 Data Scientist**
"The sexiest job of the 21st century," data scientists are in high demand from companies that need help making business decisions.
Median annual salary: £62,500
- 11 Cybersecurity**
"Never fail in such incidents and outside that organizations must be maintained in order to protect enterprise IT initiatives and these IT professionals are the ones to do it. As the case becomes increasingly important, those with experience in data, information, network, systems and cloud security are in high demand this year."
Median annual salary: £57,500
- 12 Network Administrator**
A network administrator is responsible for keeping communications and information flowing securely by overseeing communication systems and networks.
Median annual salary: £40,000
- 13 Software Developer**
Software development positions account for the largest number of job openings in the tech world.
Median annual salary: £45,000
- 14 BI Analyst**
A BI Analyst needs to be able to communicate to stakeholders a company's unique data with complete understanding and clarity. Acquiring Business Intelligence (BI) Analysts need experience with reporting tools, SQL programming and analytics.
Median annual salary: £45,000
- 15 Hardware Engineer**
Any computer hardware you've ever encountered has been designed by a hardware engineer. They are responsible for creating, developing and maintaining hardware or improve existing hardware by working in a team with other focusing on hardware and software.
Median annual salary: £32,500

Desarrollador

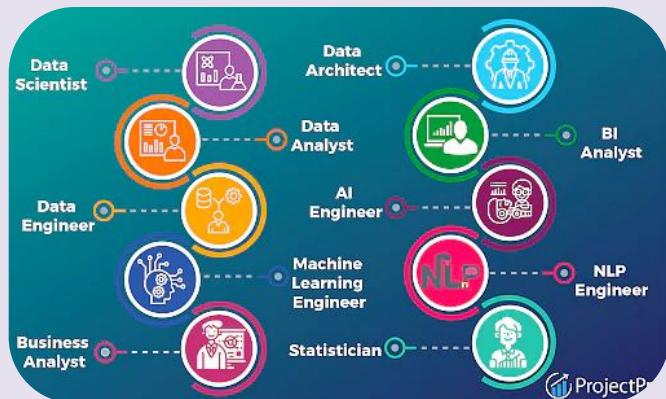
El rol principal es la creación y adaptación de programas informáticos, aunque obviamente se trata de una descripción algo simplista. Su ámbito de aplicación abarca una serie de aplicaciones, como programas, procesos, redes, actualizaciones de versión, parches, migraciones, DevOps y pruebas.

- **Creación de un código** específico y pruebas posteriores.
- Colaboración con los clientes sobre los **informes** necesarios y **supervisión** del proceso que los convierte en realidad.
- Uso de una serie de herramientas de desarrollo para **facilitar el uso de procesos y sistemas**.
- Cartografiar el **diseño de una aplicación** de software y utilizar diagramas de **flujo para resaltar** cada etapa del proceso.
- Organización de **actualizaciones y reparaciones** de aplicaciones de **software existentes**.
- Comunicación del progreso con la dirección a través de **informes, reuniones y presentaciones**.



→ Desarrollo en datos

Un **analista de datos** necesita **procesar e interpretar** los datos. Un **científico de datos** debe poder **crear y desarrollar** herramientas que procesen la información. Echemos un vistazo a cada rol con un poco más de profundidad. Además, un **ingeniero de datos** necesita poder **crear programas o sistemas** que puedan **tomar datos y convertirlos en información** significativa que se pueda estudiar.

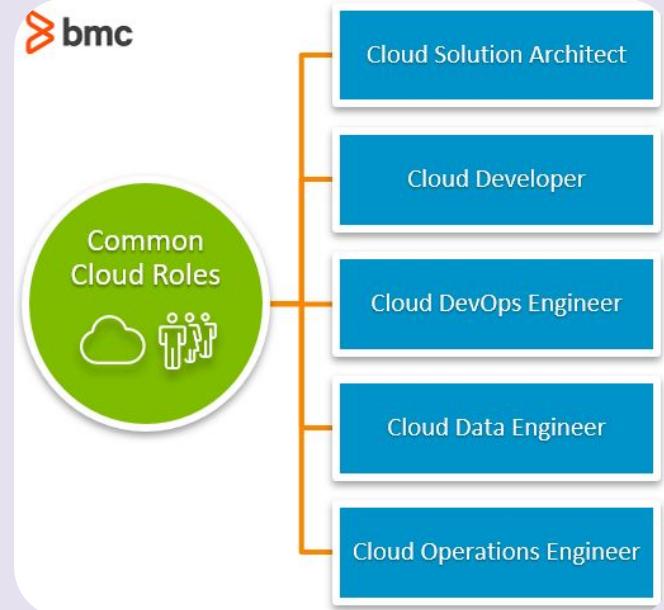




Desarrollo en la nube

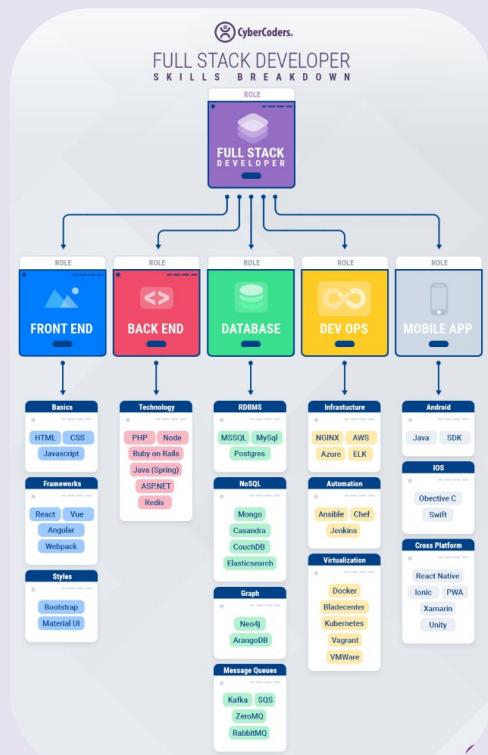
Una **desarrolladora o desarrollador cloud** se centra en **desarrollar componentes de la nube y funciones sin servidor** para organizaciones. La función principal de un desarrollador cloud es el desarrollo de scripts y procesos empresariales, siempre intentando aprovechar las funcionalidades cloud para automatizar el proceso de configuración e implementación.

Como Cloud Developer, es **crucial que conozcas las capacidades de la nube**, así como elementos de red y **conectividad** (por ejemplo, cortafuegos, WAF, routing y seguridad, y segmentación). El desarrollador cloud es responsable de **monitorizar procesos y de evaluar los recursos consumidos** en el proceso de **autoescala e implementación** de componentes dinámicos.





Tecnología que ocupan





Tecnología que ocupan

COMMON TECHNOLOGIES

that you would need to develop expertise in

FRONT END

HTML

HTML5

Java Script

J Query

CSS3

BACKEND

Ruby on Rails

PHP

Angular2

Node.js

.Net

DATABASE

MySQL

MongoDB

CouchDB

MS SQL

VERSION CONTROL

GIT

Grunt

Xdebug

Subversion

Teamwork

PROJECT MGMT TOOLS

Basecamp

Jira

Trello

Redmine



Tecnología que ocupan

CAMPOS DE APLICACIÓN DE PYTHON

Python es el lenguaje más usado, es fácil de aprender y tiene muchos campos de aplicación. ¿Qué esperas para aprenderlo?

SEGURIDAD INFORMÁTICA

Programa scripts que ejecuten pruebas automáticas para detectar vulnerabilidades.

TESTING Y QA

Automatiza tests de código y de funcionalidades.

VIDEOJUEGOS

Crea videojuegos con los frameworks: PyGame, PyOpenGL, Blender, etc.



DESARROLLO WEB

Crea apps web con frameworks como Django, Flask, Pyramid, etc.



BIG DATA Y DATA SCIENCE

Extrae, procesa, almacena (ETL) y analiza grandes cantidades de datos.



MACHINE LEARNING

Escribe modelos de machine learning con librerías como SciKit, SciPy, etc.

Comienza a estudiar Python en EDteam y #NoTeDetengas

ed.team/cursos/python



¿EN QUÉ PUEDES TRABAJAR SI SABES JAVASCRIPT?

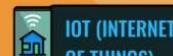
JS es el lenguaje con mayor demanda laboral de toda la web. Y en muchos rankings **ocupa el puesto #1** en popularidad y uso.

DESARROLLO WEB



FRONTEND

Desarrolla aplicaciones **del lado del cliente** (lo que se ejecuta en el navegador).



IOT (INTERNET OF THINGS)

Conecta objetos cotidianos (neveras, televisores, etc.) a Internet.

BACKEND



Programa la lógica del servidor, conexión a base de datos y el intercambio de datos con el frontend.



MÓVIL

Crea apps con frameworks como Ionic, React Native, Native Script, etc.

ESCRITORIO



Crea apps compatibles para Mac, Windows y Linux con frameworks como Electron.

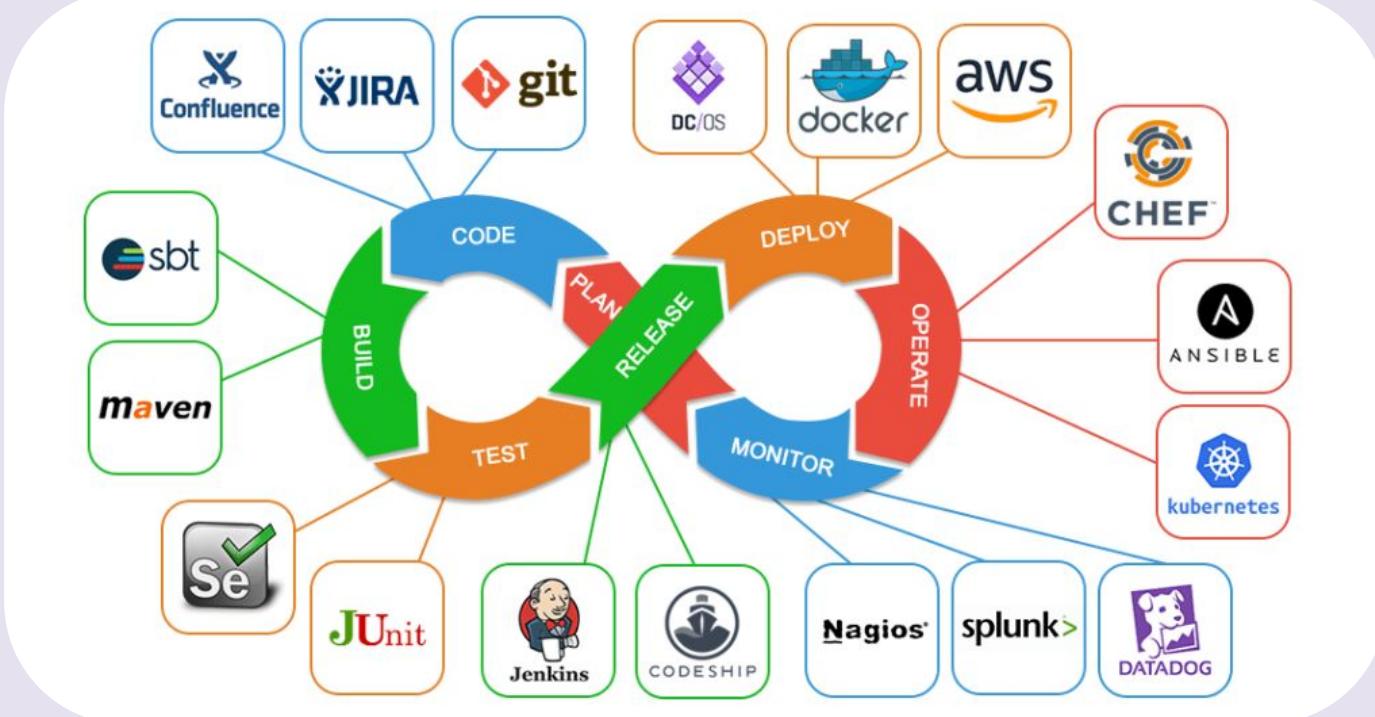


Aprende JavaScript desde cero a avanzado en:

ed.team/javascript

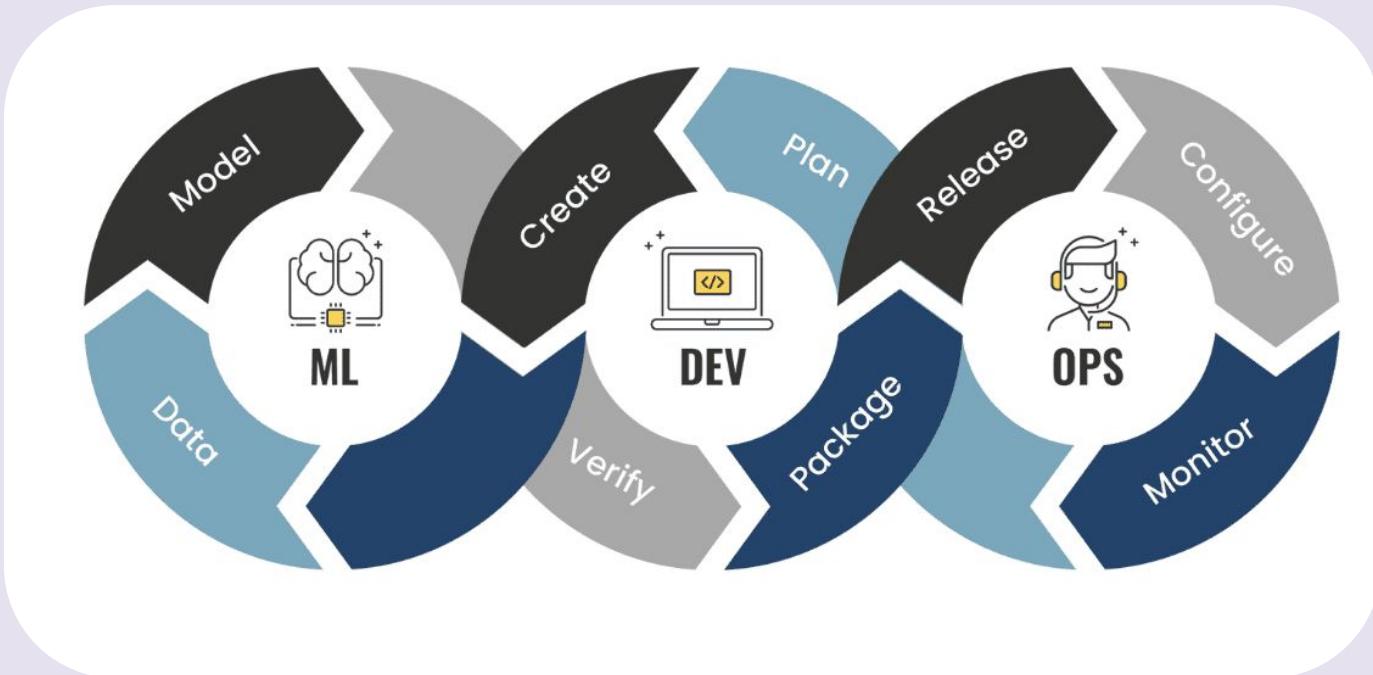


Tareas a realizar





Tareas a realizar



Salarios en México (2021)

Rol	Promedio	Mediana	Máximo	Mínimo	Muestras	Desviación
Artificial Intelligence Developer	\$34,133.33	\$30,000.00	\$80,000.00	\$8,000.00	9	23751.00
Programador Salesforce	\$29,555.56	\$30,000.00	\$59,000.00	\$9,000.00	9	16583.96
Programador Back End	\$26,784.86	\$25,000.00	\$76,000.00	\$4,800.00	297	13499.27
Programador Full Stack	\$24,534.83	\$21,000.00	\$100,000.00	\$3,500.00	646	14075.20
Programador Móvil	\$24,167.03	\$20,000.00	\$70,000.00	\$5,000.00	88	14960.99
Programador Front End	\$20,069.81	\$17,500.00	\$60,000.00	\$4,000.00	167	11713.56
Programador de Aplicaciones de Escritorio	\$18,985.68	\$18,300.00	\$35,000.00	\$6,000.00	69	8745.40

Años de experiencia	Promedio	Mediana	Maximo	Mínimo	Muestras	Desviación
1 a 3 años	\$15,965.00	\$14,800.00	\$70,000.00	\$3,500.00	429	8855.57
4 a 5 años	\$24,282.53	\$21,800.00	\$84,000.00	\$5,000.00	271	12276.13
6 a 10 años	\$30,240.39	\$29,000.00	\$93,400.00	\$5,000.00	335	15180.71
11 a 15 años	\$29,934.00	\$30,000.00	\$60,000.00	\$5,000.00	120	12219.08
16 a 20 años	\$31,358.32	\$32,500.00	\$60,000.00	\$6,000.00	72	13051.38
21 a 25 años	\$30,973.68	\$28,000.00	\$55,000.00	\$16,900.00	19	10461.99
Más de 25 años	\$29,941.44	\$28,000.00	\$100,000.00	\$8,500.00	39	15932.78

Salarios en México (2021)

Lenguaje de programación	Promedio	Mediana	Max	Min	Muestras	Desviación
C++	\$38,654.55	\$45,000.00	\$54,200.00	\$15,000.00	11	16070.87
Swift	\$33,721.71	\$32,000.00	\$70,000.00	\$8,000.00	17	19167.46
Go	\$33,666.67	\$30,000.00	\$55,000.00	\$20,000.00	6	13002.56
Apex	\$31,750.00	\$30,000.00	\$59,000.00	\$9,000.00	8	16272.24
Ruby	\$30,590.43	\$29,000.00	\$85,000.00	\$10,000.00	14	18658.85
PL-SQL	\$27,914.93	\$28,300.00	\$80,000.00	\$6,600.00	28	13645.62
Kotlin	\$27,838.10	\$24,100.00	\$57,000.00	\$12,000.00	21	13257.43
TypeScript	\$26,583.33	\$24,000.00	\$70,000.00	\$8,000.00	12	16940.85
Dart	\$26,285.71	\$22,000.00	\$40,000.00	\$7,000.00	7	12037.64
Java	\$26,027.33	\$25,000.00	\$76,000.00	\$4,000.00	283	12987.03
C#	\$25,472.89	\$24,000.00	\$100,000.00	\$3,500.00	268	13111.73
Python	\$25,096.00	\$20,000.00	\$65,000.00	\$6,000.00	75	15437.55
Cobol	\$24,061.54	\$22,000.00	\$42,000.00	\$10,000.00	13	10314.52
JavaScript	\$23,401.52	\$20,000.00	\$93,400.00	\$4,245.00	256	14708.45
Delphi	\$23,285.71	\$20,000.00	\$34,000.00	\$18,000.00	7	6264.03
PHP	\$17,200.33	\$15,000.00	\$80,000.00	\$4,000.00	173	10614.44
Visual Basic	\$16,466.14	\$15,250.00	\$35,000.00	\$6,000.00	28	6706.99
C	\$13,120.00	\$14,000.00	\$20,000.00	\$6,000.00	5	5046.98

Salarios en México (2021)

Framework	Promedio	Mediana	Máximo	Mínimo	Muestras	Desviación
Express	\$35,117.47	\$30,000.00	\$84,000.00	\$13,920.00	17	17973.41
Ruby on Rails	\$32,365.18	\$30,000.00	\$85,000.00	\$10,000.00	11	18998.30
.Net Core	\$31,700.00	\$30,500.00	\$70,000.00	\$6,500.00	30	14763.47
Nodejs	\$30,246.88	\$23,500.00	\$92,000.00	\$2,500.00	32	23958.95
Spring	\$28,331.04	\$27,000.00	\$76,000.00	\$6,000.00	187	12512.28
Flask	\$28,173.33	\$20,000.00	\$54,000.00	\$6,000.00	15	16600.42
React	\$26,536.97	\$22,000.00	\$93,400.00	\$4,500.00	117	16590.36
Django	\$26,212.90	\$20,000.00	\$65,000.00	\$9,000.00	31	16029.16
ASP	\$25,782.11	\$21,000.00	\$55,000.00	\$9,500.00	15	12795.19
MVC	\$25,463.64	\$28,000.00	\$41,000.00	\$9,500.00	11	8473.73
.NET	\$24,891.08	\$23,000.00	\$100,000.00	\$5,000.00	132	13101.77
Android	\$24,404.71	\$20,000.00	\$57,000.00	\$6,000.00	17	14505.24
Entity Framework	\$23,477.27	\$23,500.00	\$40,000.00	\$10,000.00	22	9073.53
Vue	\$23,172.71	\$19,000.00	\$84,000.00	\$7,300.00	48	13886.24
Angular	\$23,143.12	\$22,000.00	\$93,400.00	\$4,000.00	91	14077.26
Codeigniter	\$17,133.33	\$16,000.00	\$28,000.00	\$8,000.00	27	5449.77
Laravel	\$16,279.02	\$15,000.00	\$50,000.00	\$4,000.00	97	9468.76

US companies provide the highest pay increase for remote devs in Latin America
Companies based in these countries offer the biggest pay raises:



3.3x
United States



3.0x
United Kingdom



2.1x
Argentina



2.0x
Uruguay



1.8x
France



1.5x
Canada



1.5x
Colombia



1.4x
Mexico

*Global remote salary X times of local remote salary

arc.dev



Back-End

El desarrollador backend trabaja con **lenguajes de programación y herramientas específicas para su área**. Aunque es **recomendable** que sepa **aspectos básicos de Frontend** para poder comunicarse con esa área, por lo general se buscan perfiles especializados que puedan realizar su trabajo con mayor calidad.

El trabajo de un desarrollador backend **no es visible para el usuario** que utilizará la aplicación o el sitio web, ya que su trabajo consiste en el **soporte del sitio y su mantenimiento** en cuanto a servidores y lógica del producto digital.

El desarrollador especializado en **backend es uno de los más importantes al momento de crear un producto digital** ya que estará **presente desde su construcción hasta el final**, e incluso después para dar el **mantenimiento y el soporte al mismo**.

¿CONOCES LOS ROLES EN BACKEND?

Aunque una sola persona podría hacer estos tres roles **es recomendable separarlos si quieres que tu proyecto crezca**.



BACKEND DEVELOPER

Se encarga de la lógica del negocio (el código del lado del servidor) y crea las APIs para que el Frontend pueda consumirlas.



DATABASE ADMINISTRATOR

Diseña, implementa, mejora y mantiene el sistema de base de datos.



ADMINISTRADOR DEL SERVIDOR

Gestiona la instalación, soporte y mantiene el servidor en donde se aloja la web o app.

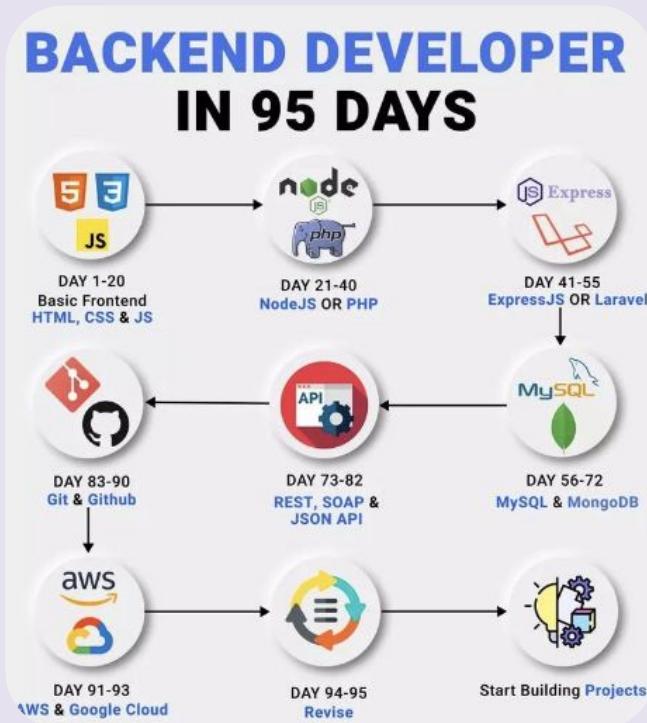
Domina las tecnologías para ser un desarrollador backend en:

ed.team/cursos





Back-End



Front-End

El desarrollador frontend se **encarga de crear la interfaz de un sitio web o aplicación a través de código**.

En el proceso de desarrollo de un producto digital, el **desarrollador frontend recibe los diseños creados por el diseñador UI**, y los **traduce a lenguajes de programación**.

El desarrollador frontend es uno de los pilares que sostienen la creación de un sitio web o aplicación. Este **perfil tiende a ser creativo**, ya que antes de que el perfil de diseñador UI se hiciera más común y necesario, eran los desarrolladores frontend quienes proponían el diseño de las interfaces.

El desarrollo de la interfaz de **sitios web o aplicaciones está siendo cada vez más demandado** ya que, con la aceleración del mundo digital, hay cada vez más productos que solucionan **problemas de la vida diaria** de las personas.

¿CONOCES LOS ROLES EN FRONTEND?

¿VERDAD O MITO?

Puesto que el frontend ocurre en el lado del cliente y es lo que el usuario ve, un programador frontend debe saber diseñar además de programar.



DISEÑADOR UI

Diseña los flujos de usuario, las pantallas e interacciones en un programa de diseño (no escribe código).



MAQUETADOR

Lleva el diseño a código HTML y CSS. No se preocupa de la lógica, solo de la presentación.



PROGRAMADOR FRONTEND

Agrega datos reales desde una API y lógica a la presentación creada por el maquetador.

Domina las tecnologías para ser un desarrollador frontend en:

ed.team/frontend



Front-End



FullStack

Full-Stack DEVELOPER

Front-End

Back-end

 HTML5 CSS3

 JavaScript

 Bootstrap

 React

 JQuery

 Angular

 Python

 PHP

 Java

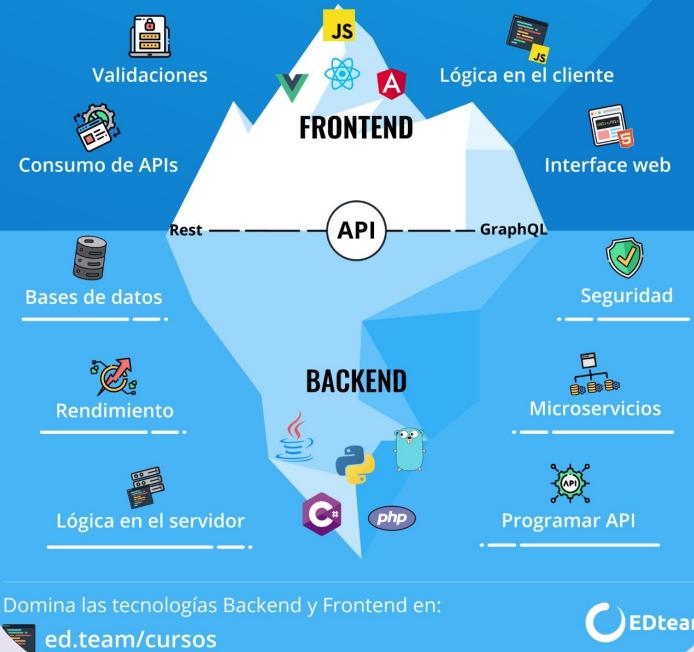
 SQL

 Django

 MY-SQL

 MongoDB

¿QUÉ ES BACKEND Y FRONTEND?



06

Conocimientos básicos

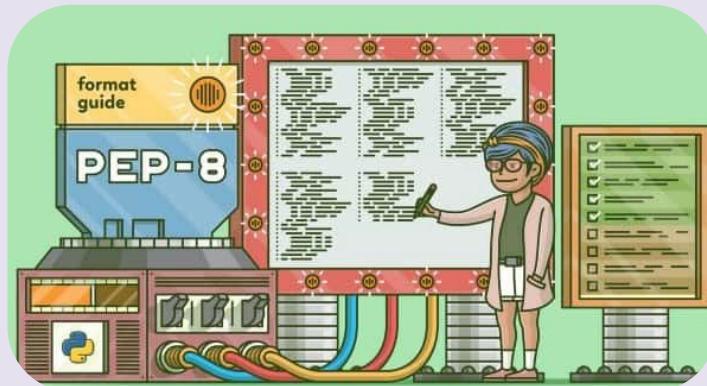




Estándares

PEPs (Python Enhancement Proposals) que regulan distintos aspectos de la **producción de código con Python**. Los PEPs cubren desde principios de diseño generales, como el Zen de Python hasta convenciones a la hora de escribir documentación, como Docstring Convention.

El **estándar para JavaScript es ECMAScript (ECMA-262)** y la especificación de la API para la Internacionalización de ECMAScript (ECMA-402). La documentación en MDN está basada enteramente en las últimas versiones preliminares de ECMA-262 y ECMA-402. Y en algunos casos donde algunas propuestas para nuevas funciones para ECMAScript ya hayan sido implementadas en los navegadores, la documentación y algunos artículos de MDN pueden hacer uso de algunas de estas funciones.





Estándares

¿Qué define o recomiendan los estándares? Algunos ejemplos son:

- Declaración de variables
 - Nombres de variables
 - Valores de variables
 - Declaración de multiples variables
 - Variables de tipo colección
- Declaración de funciones
 - Nombres de funciones
 - Llaves y espaciado de funciones
 - Tipos de funciones
 - Argumentos de una función
- Construcción de cadenas
- Operaciones
 - Operaciones lógicas de igualdad y desigualdad

The image shows three tweets from a user named 'One Developer Army' (@OneDevlo...). Each tweet features a profile picture of a man with glasses and a beard.

- 1st rule of programming:** - You're always in control
9 replies, 40 retweets, 198 likes
- 2nd rule of programming:** - Programmers are lazy
9 replies, 102 retweets, 513 likes
- 3rd rule of programming:** - If it works don't touch it
40 replies, 560 retweets, 1,861 likes



Funciones

Una función es un bloque de código que realiza alguna operación. Una función puede definir **opcionalmente parámetros de entrada** que permiten a los llamadores pasar argumentos a la función. Una función **también puede devolver un valor como salida.** Las funciones son útiles para **encapsular las operaciones comunes** en un solo bloque reutilizable, idealmente con un nombre que describa claramente lo que hace la función.

No hay ningún límite práctico para la longitud de la función, pero **un buen diseño tiene como objetivo funciones que realizan una sola tarea bien definida.** Los algoritmos complejos deben dividirse en funciones más sencillas y fáciles de comprender siempre que sea posible.

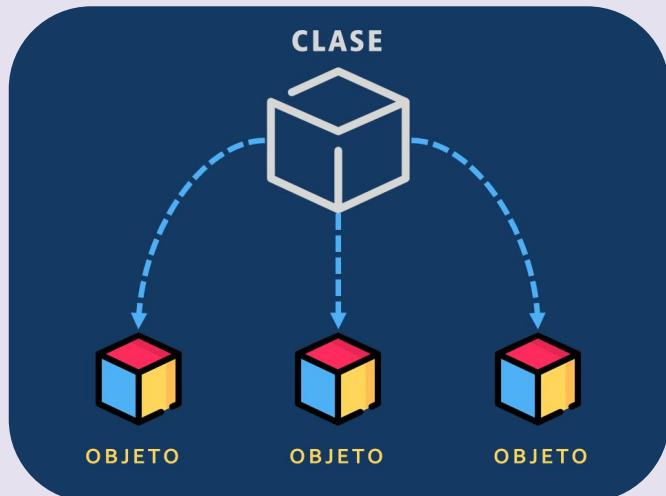
```
function cuadrado(n){  
    return n * n  
}
```

```
let num = cuadrado(2);
```



Clases

Una **clase** es un elemento de la programación orientada a objetos que actúa **como una plantilla y va a definir las características y comportamientos de una entidad**. La clase va a ser como un molde a partir del cual vamos a poder definir entidades. Una clase va a definir las características y los comportamientos de una entidad. Si yo defino, por ejemplo, la clase persona, sus características o atributos podrían ser género, es decir, si es hombre o mujer, edad y nombre. Los comportamientos o métodos que tendríamos en el caso de la clase persona podrían ser respirar, moverse, caminar, pensar, entre otras.





Decoradores

Un **decorador** es un **patrón de software** que se utiliza para **alterar el funcionamiento de una determinada pieza de código**; ya sea una función, o una clase, sin la necesidad de emplear otros mecanismos como la herencia.

En concreto, nos referimos a **funciones** u objetos con un **comportamiento similar que nos permiten alterar cómo funcionan** otras entidades sin tener que modificar su código explícitamente.

Un ejemplo del uso de patrones de diseño

```
class MyClass:  
    @property  
        def my_property(self):  
            return self._my_property
```



Operaciones sync-async

Un código síncrono es aquel código donde cada instrucción espera a la anterior para ejecutarse mientras que un código asíncrono no espera a las instrucciones diferidas y continúa con su ejecución. Por lo general la asincronía permite tener una mejor respuesta en las aplicaciones y reduce el tiempo de espera del cliente.

Síncrono

Cada instrucción se ejecutará en secuencia hasta terminar.

Asíncrono

En el caso asíncrono, algunas de las instrucciones se ejecutarán a destiempo.



IDE vs Editores

Un editor de código es un **programa ligero** que no exige mucha RAM o procesador, en dónde puedes **abrir y crear un archivo a la vez y guardarlos en una carpeta**. A un editor puedes **agregarle plugins para realizar muchas más funciones** (por ejemplo que pueda soportar múltiples lenguajes) y hacerlo más potente.

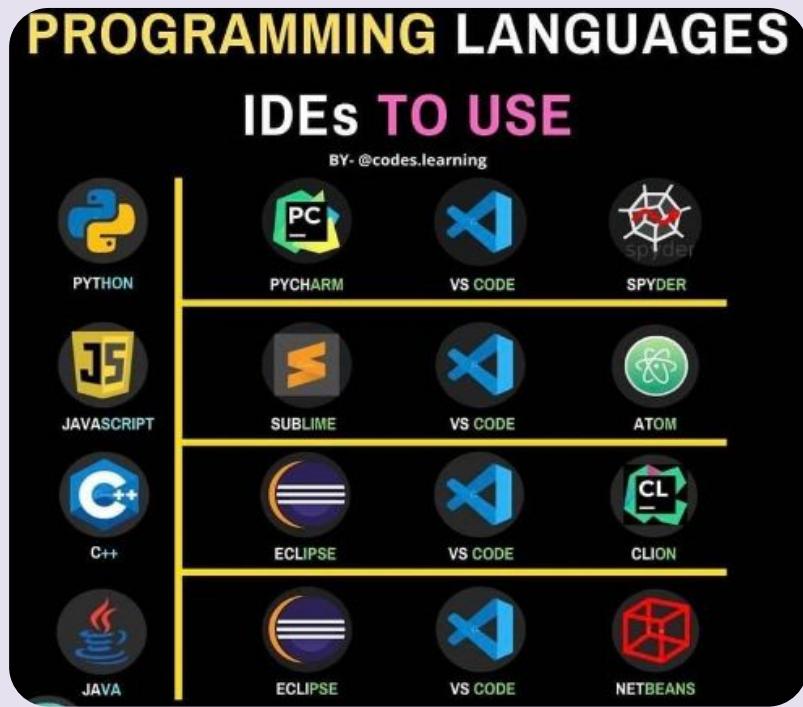
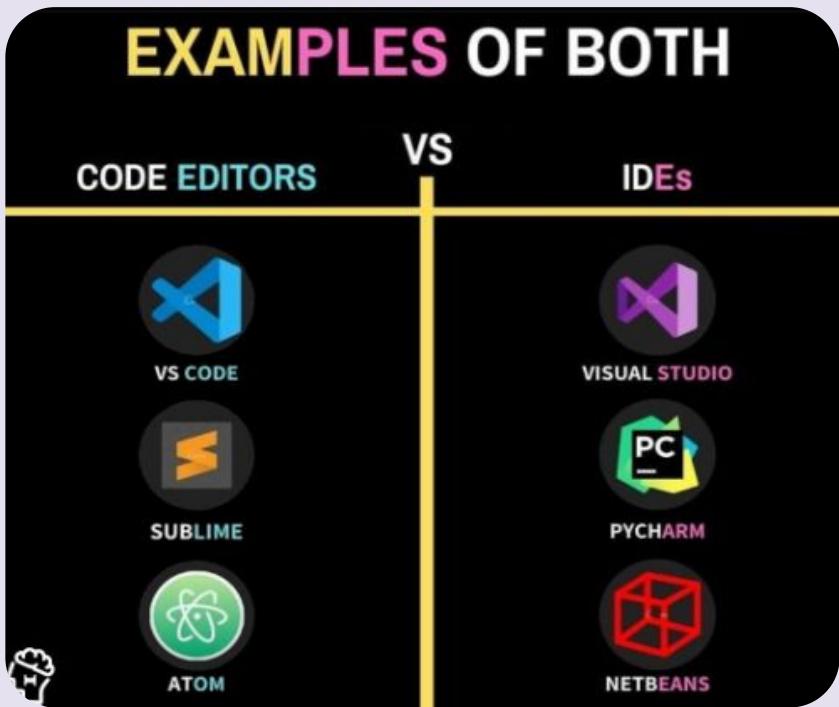
Un **ambiente de desarrollo integrado** (Integrated Development Environment), a diferencia de un editor, es un programa más pesado que pide mucha más memoria RAM y un procesador más poderoso, además de que es un **espacio para trabajar proyectos completos no solo en archivos**. Contienen **herramientas integradas**, es decir, ahora ya no crearás carpetas por tu cuenta, pueden tener un **compilador** (para los lenguajes compilados), un **emulador, control de versiones y terminales**.

The infographic is titled "EDITOR VS IDE" and features a central question: "Con ambos puedes escribir código, pero ¿en qué se diferencian?". It compares the two based on several key points:

- Software ligero con ayudas para escribir código** (vs. **Integra un editor con las herramientas que necesita un desarrollador (debugger, compilador, etc.)**)
- Soporta múltiples lenguajes y tecnologías** (vs. **Se especializa en un lenguaje o tecnología (Java, Python, Go, Android, etc.)**)
- Enfocado en archivos (no tienen el concepto de proyecto)** (vs. **Enfocado en proyectos completos. Desde la primera línea hasta la salida a producción.**)
- Puedes agregar plugins para darle el poder de un IDE pero te toca configurar cada uno a mano.** (vs. **Trae herramientas integradas y configuradas (ej. Android Studio trae un emulador de Android).**)

At the bottom, there are sections for "EJEMPLOS DE EDITORES" (VSCode, Sublime Text, Atom) and "EJEMPLOS DE IDES" (Android Studio, Eclipse, IntelliJ IDEA). The infographic also includes the EDteam logo and the tagline "Domina la tecnología con EDteam y #NuncaTeDetengas" with the URL "ed.team/cursos".

IDE vs Editores

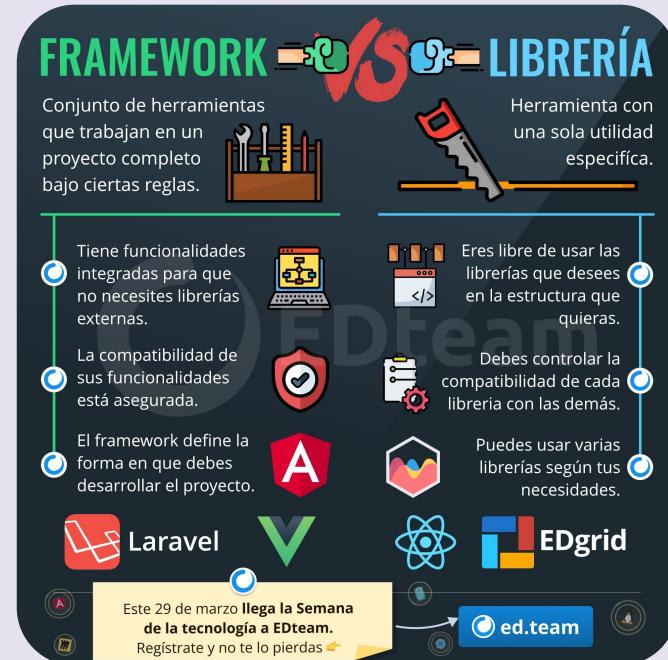




Framework

Un **framework** es un **entorno o marco de trabajo, un conjunto de prácticas, conceptos y criterios a seguir estandarizados**. Siguiendo unas reglas, el framework nos obliga a utilizar **buenas prácticas** para nuestro código.

Por otro lado, los frameworks también nos **proporcionan una serie de herramientas ya desarrolladas**. Suelen ser **funciones comunes** en todos los proyectos. Por ejemplo, si tenemos un proyecto web seguramente tendremos usuarios y si tenemos usuarios lo más normal es que estos tengan que hacer «login». Esta funcionalidad de hacer «login» con un correo electrónico y una contraseña ya estaría hecha si utilizamos un framework.





Scripts vs Programas

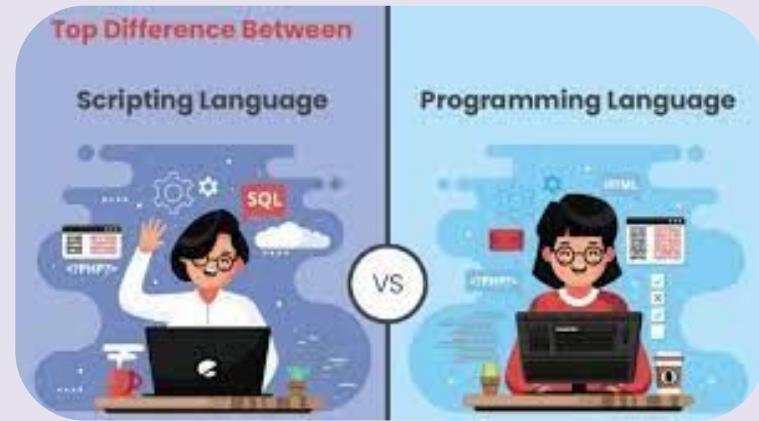
Un script es un archivo de código que puede ejecutar comandos para realizar una tarea en específico. Es un solo archivo.

Un programa al igual que el script ejecuta comandos pero puede tener la diferencia de no ser únicamente un solo archivo sino toda una solución.

Todos los scripts son programas pero no todos los programas son scripts.

Un ejemplos de programas y script son los siguientes:

- Script: Código que manda mensaje todos los días
- Programa: Sistema de facturación





Control de versiones

El **control de versiones**, también conocido como "**control de código fuente**", es la práctica de **rastrear y gestionar los cambios en el código de software**. Los sistemas de control de versiones son herramientas de software que **ayudan** a los equipos de software a **gestionar los cambios en el código fuente a lo largo del tiempo**.

El software de control de versiones realiza un **seguimiento de todas las modificaciones en el código** en un tipo especial de base de datos. Si se **comete un error**, los desarrolladores **pueden ir hacia atrás en el tiempo y comparar las versiones anteriores** del código para ayudar a resolver el error, al tiempo que se **minimizan las interrupciones** para todos los miembros **del equipo**.



→ Control de versiones





API

The API

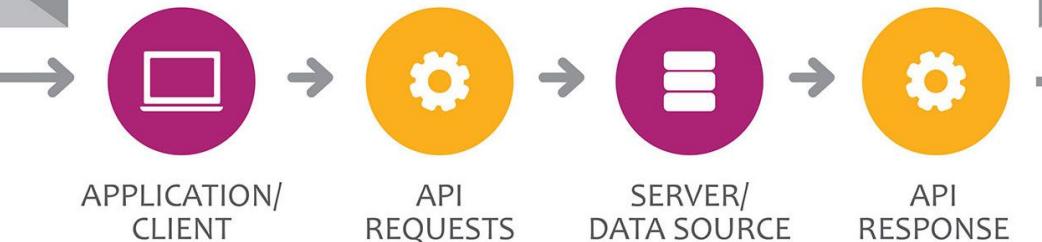
Application Programming Interface



API Definition

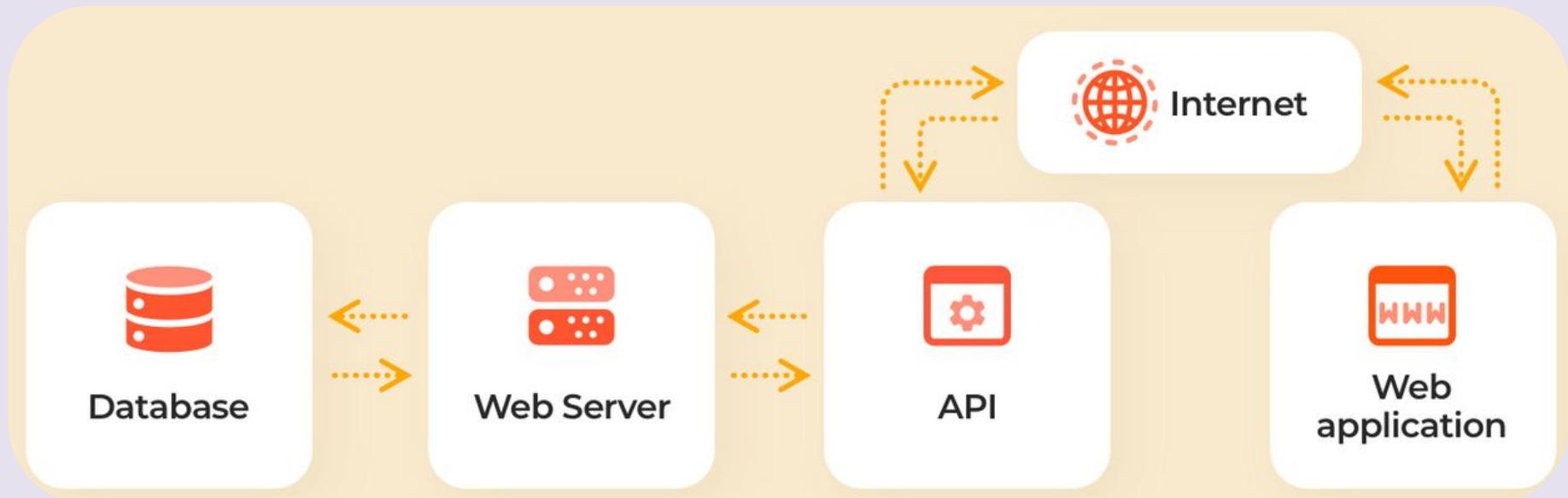
An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data.

How an API works





API



Swagger Petstore 1.0.8

[Base url: petstore.swagger.io/v2]
<http://petstore.swagger.io/v2/swagger.json>

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net, #swagger](#). For this sample, you can use the api key `special-key` to test the authorization filters.

[Terms of service](#)

[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

Schemes

HTTP

Authorize 

pet Everything about your Pets

POST /pet Add a new pet to the store 

PUT /pet Update an existing pet 

GET /pet/findByStatus Finds Pets by status 

GET /pet/findByTags Finds Pets by tags 

GET /pet/{petId} Find pet by ID 

POST /pet/{petId} Updates a pet in the store with form data 

DELETE /pet/{petId} Deletes a pet 



Patrones de Diseño

Los **patrones de diseño** o **design patterns**, son una **solución general, reutilizable y aplicable a diferentes problemas** de diseño de software. Se trata de **plantillas que identifican problemas** en el sistema y proporcionan **soluciones apropiadas a problemas generales** a los que se han **enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error.**

En definitiva, los patrones de diseño **te ayudan a estar seguro de la validez de tu código**, ya que son soluciones que funcionan y **han sido probados por muchísimos desarrolladores** siendo menos propensos a errores.

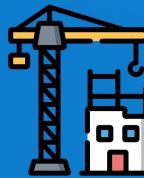
¿QUÉ SON LOS PATRONES DE DISEÑO?

Son técnicas para resolver problemas recurrentes de diseño de software.



CREACIONALES

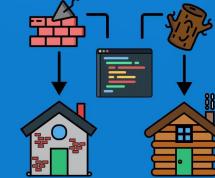
Solucionan la creación de objetos, hace un sistema independiente de cómo sus objetos son creados.



Ej. Singleton, Factory.

ESTRUCTURALES

Describe cómo los objetos se componen para formar estructuras complejas.



Ej. Adapter, Decorator.

COMPORTAMENTALES

Establece soluciones relacionados con el comportamiento de la aplicación respecto a la interacción entre objetos y clases.



Ej. Observer, State.

Si tienes un problema de diseño es muy probable que ya exista un patrón que lo solucione, no reinventes la rueda.



Alejandro Rodriguez
Backend Developer en EDteam

Domina los patrones de diseño de software en:
ed.team/programacion





Microservicios

Los **microservicios** son tanto un **estilo de arquitectura** como un modo de programar software. Con los microservicios, **las aplicaciones se dividen en sus elementos más pequeños e independientes entre sí.**

Cada uno de esos elementos o procesos es un microservicio. Este enfoque de desarrollo de software **valora el nivel de detalle, la sencillez y la capacidad para compartir un proceso similar en varias aplicaciones.** Es un elemento fundamental de la **optimización del desarrollo** de aplicaciones hacia un **modelo nativo de la nube.**

¿QUÉ ES LA ARQUITECTURA DE MICROSERVICIOS?

Divide un software en varias aplicaciones pequeñas (microservicios) que se comunican entre sí.



Podemos actualizar un servicio sin volver a implementar toda la app.



Podemos usar distintos lenguajes de programación en el mismo proyecto.



Si un servicio deja de funcionar, no se detiene toda la app, solo el microservicio que falla.



Los microservicios pueden escalarse de forma independiente.



Domina la tecnología con EDteam. Empieza gratis en:
ed.team/cursos

→ Contenedores

Los **contenedores**, en esencia, son procesos que **corren dentro del sistema operativo** los cuales cumplen con algunas características que los hacen "especiales". **Comparten el ecosistema junto con procesos "comunes"** y es gracias al **KERNEL** de Linux que, mediante algunas características de este, brinda a estos procesos, capacidades notables.

Un contenedor es un **proceso** que corre **aislado** de otros contenedores, incluso también se ejecuta aislado de otros procesos "comunes". **Los contenedores son una forma de virtualización del sistema operativo. Un solo contenedor** se puede usar para **ejecutar cualquier cosa**, desde un microservicio o un proceso de software a una aplicación de mayor tamaño.

¿QUÉ ES docker?

Es un proyecto open source que facilita la creación y ejecución de aplicaciones mediante el uso de contenedores. Eliminando el clásico: "En mi computadora si funciona".
Prof. Mauricio Ballesteros Valladares

¿CÓMO FUNCIONA?

- Una imagen es una copia de una aplicación o S.O. y se crean usando el comando `build`
- Las imágenes contienen todo el **código necesario** para ejecutar el proyecto.
- Utilizando las imágenes, **cualquier programador podrá correr el código** para crear los contenedores.
- Los contenedores permiten al desarrollador **empaquetar una aplicación con todas las partes que necesita**.

¿QUIÉNES LO USAN?

PayPal, Spotify, redhat, Uber

docker hub

Cuéntanos tu experiencia trabajando con Docker.
ed.team/cursos/docker

EDteam



Nube

La nube no es una entidad física, sino una red enorme de servidores remotos de todo el mundo que están conectados para funcionar como un único ecosistema. Estos servidores están **diseñados para almacenar y administrar datos, ejecutar aplicaciones o entregar contenido o servicios**, como streaming de videos, correo web, software de ofimática o medios sociales. En lugar de acceder a archivos y datos desde un equipo personal o local, accedes a ellos online desde cualquier dispositivo conectado a Internet, es decir, la información **está disponible dondequieras que vayas y siempre que la necesites.**





Sistema operativo y variables de entorno

Las **variables de entorno** son variables disponibles para tu programa/aplicación de forma **dinámica durante el tiempo de ejecución**. El valor de estas variables puede proceder de **diversas fuentes**: archivos de texto, gestores secretos de terceros, scripts de llamada, etc.

Lo importante aquí es el hecho de que el valor de estas variables de entorno **no está codificado en tu programa**. Son realmente **dinámicas y pueden cambiarse en función del entorno en el que se ejecuta tu programa**.





Arquitectura

La arquitectura de software son patrones o lineamientos que ayudan a la construcción de un programa (aplicación).

Estos patrones permiten tener una guía para los desarrolladores, analistas y todos los cargos relacionados para lograr cumplir con los requerimientos de la aplicación.

El definir las tecnologías es uno de los puntos más importantes de la arquitectura de software pero no quiere decir que si se toma una decisión sea algo definitivo que no se pueda modificar en el futuro.

Uno de los objetivos de la arquitectura de software es **crear una estructura** de la aplicación que sea **fácilmente escalable**, que **no esté fuertemente acoplada**.

Arquitecto de software



Funciones

- Diseñar arquitectura y cada componente del sistema
- Seguimiento tras implementación

Formación

- Grado de ingeniería informática o de sistemas

Habilidades

- Conocimientos de Lenguaje de Lenguaje de Modelado Universal (UML)
- Habilidades de analíticas ,organización y de liderazgo

Salario

Junior: \$ 80.000
Medio: \$ 117.000
Senior: \$ 153.000

07

Principales

avances

en la tecnología



Últimos avances



Realidad virtual

- Educación
- Cultura
- Sector de eventos
- Experiencia de cliente



Impresión en 3D

- Prótesis
- Industria automotriz
- Construcción
- Comida
- Espacial
- Etc



IoT

- Medicina
- Granjas
- Escuelas
- Espacial
- Domótica
- Etc

Últimos avances



Machine learning

El ‘*machine learning*’ –aprendizaje automático– es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello.



Big data

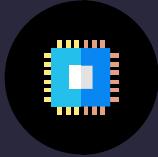
Big Data es un término que describe el gran volumen de datos, tanto estructurados como no estructurados, que inundan los negocios cada día.



Criptomonedas

Utilizan un código complejo para crear una moneda única y verificable que se puede comercializar online y utilizar para comprar. Son descentralizadas.

Últimos avances



Computo cuántico

Se basa en el uso de cubits, una especial combinación de unos y ceros. Es un paradigma de computación distinto al de la informática clásica



5G

La tecnología 5G ofrece una velocidad máxima teórica de 20 Gbps, mientras que la velocidad máxima de la tecnología 4G es solo de 1 Gbps.



Redes neuronales

Un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una.



Principales avances en AI

DALL-E 2





Principales avances en AI

Hugging Face

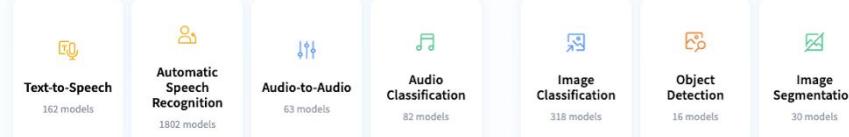
Tasks

Hugging Face is the home for all Machine Learning tasks. Here you can find what you need to get started with a task: demos, use cases, models, datasets, and more!

Natural Language Processing



Audio



Computer Vision



Inicio de 2023

We tried the AI-powered version of Microsoft Bing. Its huge, user-friendly search box and detailed responses make it so much better than Google.

Google vs. Microsoft: guerra abierta por la inteligencia artificial que va a cambiar tu vida

Microsoft desafía a Google: Bing es solo el principio

