

# Introduction to Technology

Section I



# Contenido



01

## Brief History

General context of the technology

03

## Software Development

Introduction to software development

02

## Basic principles

Basic computer skills

04

## Applied technologies

Programming languages and their use

# Contenido

05

## Types of roles

Explanation of roles in technology

07

## Latest developments

Most important advances in technology

06

## Basic knowledge

General development skills

08

## Conclusions

Final points of the section



---

01

# Brief History

of technology





# What does technology mean?

A set of theories and techniques that allow the practical use of scientific knowledge. - RAE

Technology is the set of knowledge and techniques that are applied in an orderly manner to achieve a certain objective or solve a problem. - economipedia



# What does information technology mean?



Information Technology - IT

Any equipment or interconnected system or subsystem of equipment used in the **automatic acquisition, storage, manipulation, management, movement, control, display, switching, exchange, transmission, or reception of data or information** - CSRC



# What does information technology mean?

## Important points

- Obtained
- Storage
- Handling / Transformation
- Management / Administration
- Movement
- Control
- Visualization
- Switching
- Automatic exchange, transmission or reception

of **data or information**



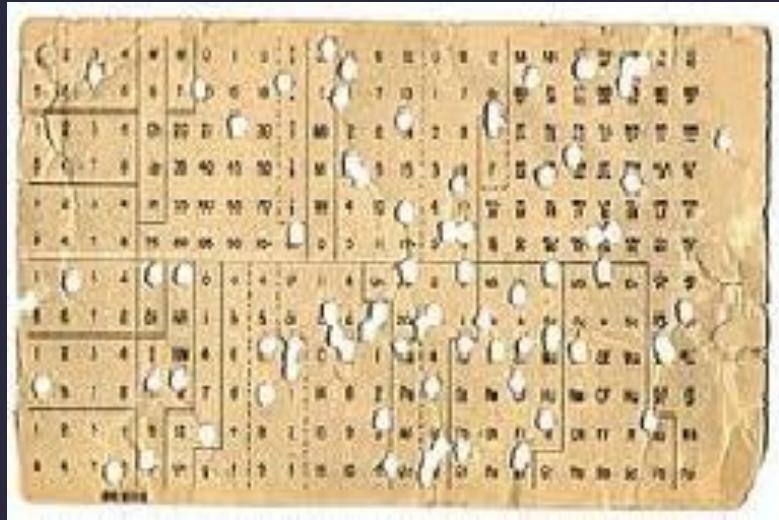
# HISTORY OF COMPUTING

(Timeline)





# Abacus



- The abacus appears, which was the first instrument used by mankind to perform calculus operations.



2,500 B.C.



# Binary System

	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	攝	賁	頤	蠱	蒙	艮	剝
地	泰	臨	明夷	復	升	師	謙	坤

- The book of mutations (I Ching) appears and in it we find the first formulation of the binary system.

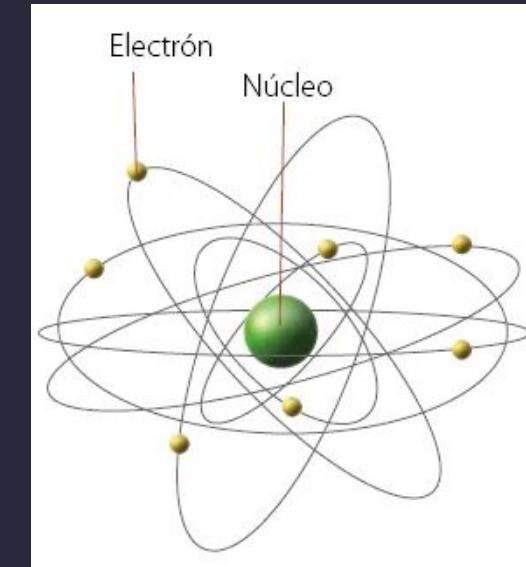


# Alphabet (Phenetius)



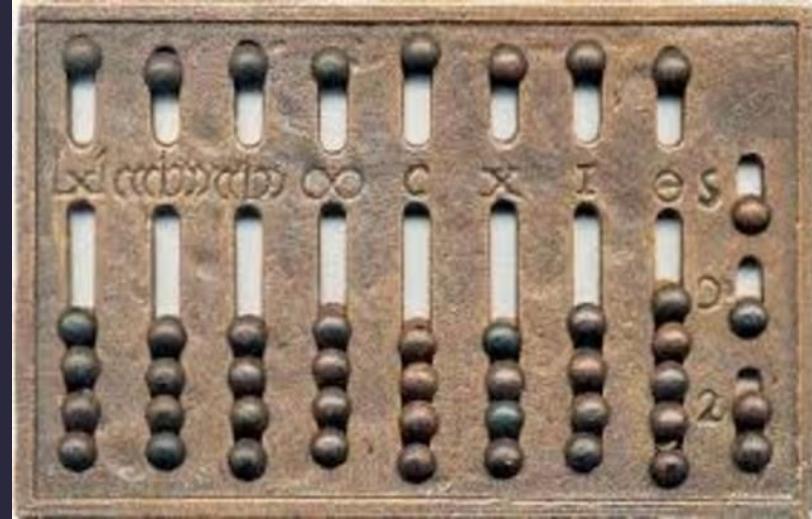
- > It was one of the most widely used writing systems as it spread throughout the Mediterranean world. It was assimilated by many other cultures adapting it to their respective languages.

# Electron



- > Thales of Miletus describes static electricity and this is where the term electron comes from.

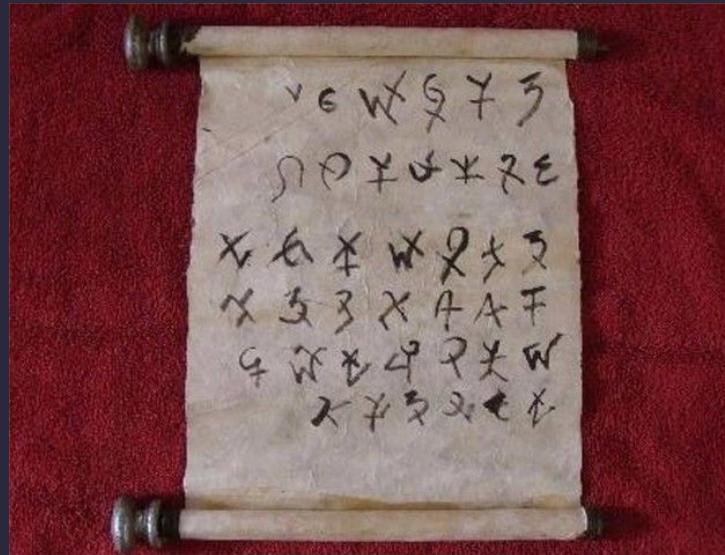
# Calculations (Roman abacus)



- > The Romans use abacuses and for the balls they use stones which they call calculus.



# Scroll



- > In the city of Pergamon, parchment began to be used for writing.



170 B.C.

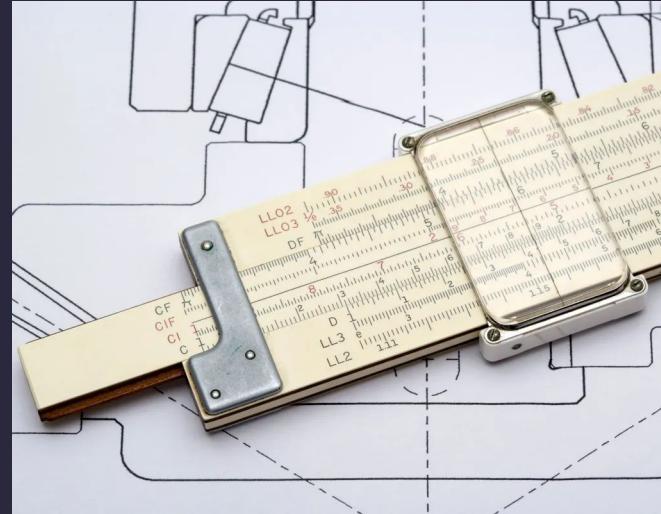


# Paper



- It is said that Ts'ai Lun first presented samples of paper to Emperor Han Ho Ti of China's Han dynasty.

# Rule of Calculation (William Oughtred)



- William Oughtred created a slide rule, which has been used by engineers until very recently.

# Mechanical calculator for addition and subtraction



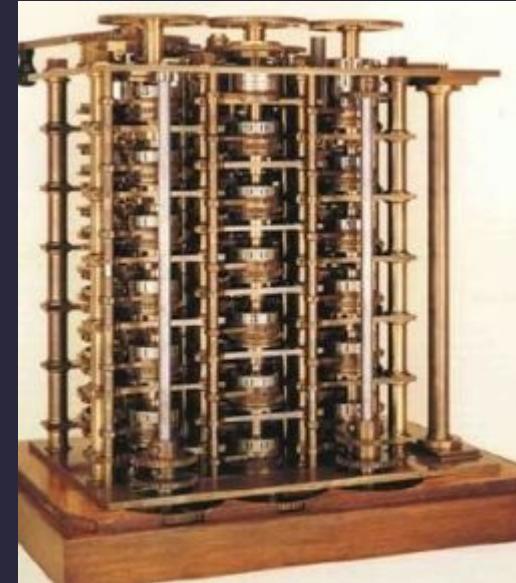
- > Blaise Pascal built the first mechanical calculator that performed addition and subtraction.

# Multiplication calculator



- Gottfried Leibnitz creates a machine capable of performing multiplications

# Design of a machine capable of executing computer programs.



- > Charles Babbage designs but does not build a processing machine capable of executing computer programs.

# Boolean Algebra

## 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.*

- > Boolean algebra is born by George Boole. The study of symbolic logic begins.



# Mechanical drilling machine



- Hermann Hollerith used a mechanical punch as a way to represent letters and digits on paper cards. Later, in 1924, he would be the founder of IBM.

# First radio transmission



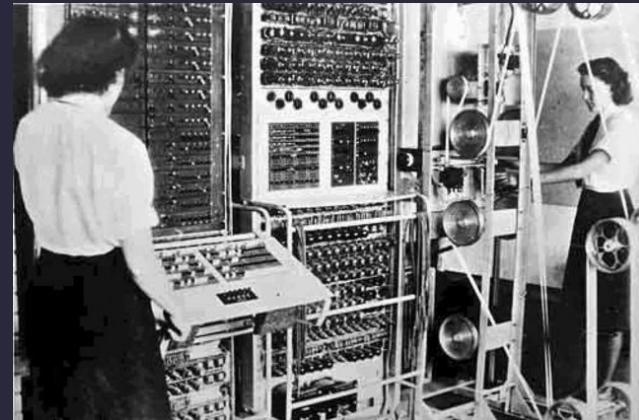
› The first radio transmission was made on Christmas Eve 1906.

# First programmable electromagnetic computer



- > Konrad Zuse presented the first programmable electromagnetic computer with a perforated tape. It had 2000 electromagnets, 64 words of 22 bits of memory, 1000 kilograms of weight and a power consumption of 4000 watts.

# Colossus

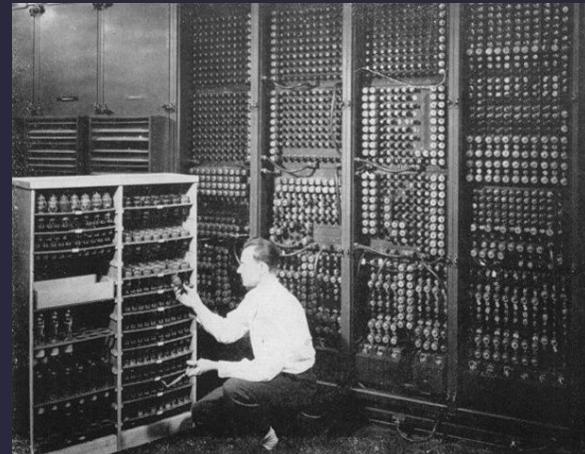


- Alan Turing builds the Colossus, a computer that allows deciphering in a few seconds the secret messages of the Nazis in World War II (Enigma Machine).

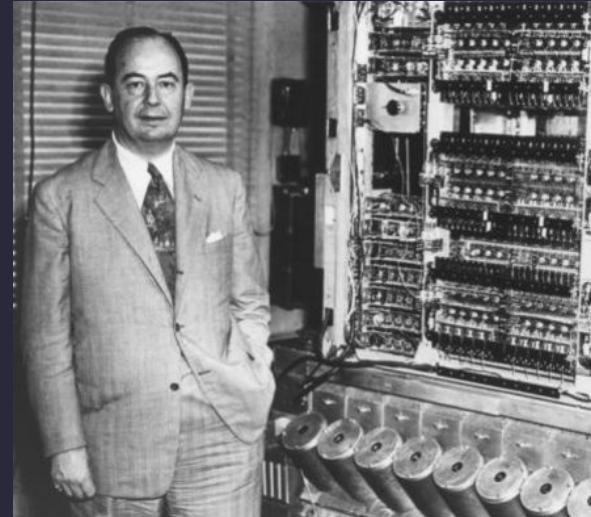


# ENIAC

- > John Presper-Eckert and John W. Mauchly developed the ENIAC (Electronic Numerical Integrator And Calculator). It is considered to be the first computer, since it worked completely electronically.



# Arquitectura Von Neumann EDVAC



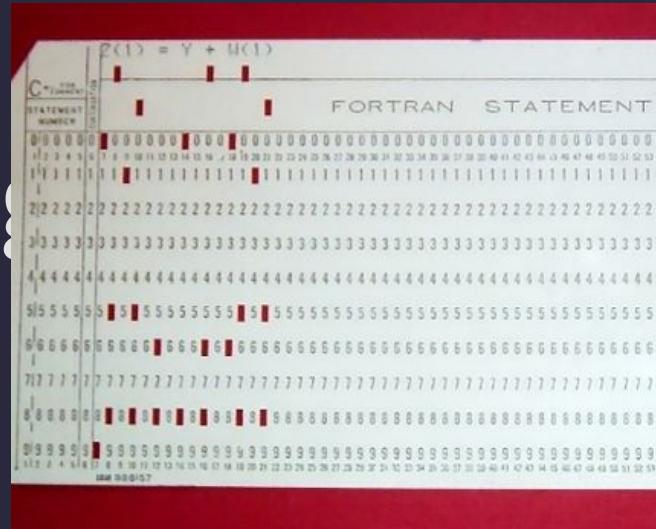
- > John Von Neumann created EDVAC. He had the idea of putting the instructions in the same memory as the data. He wrote them in binary code. This is known as the Von Neumann Architecture and is used today.

# UNIVAC 1 is born



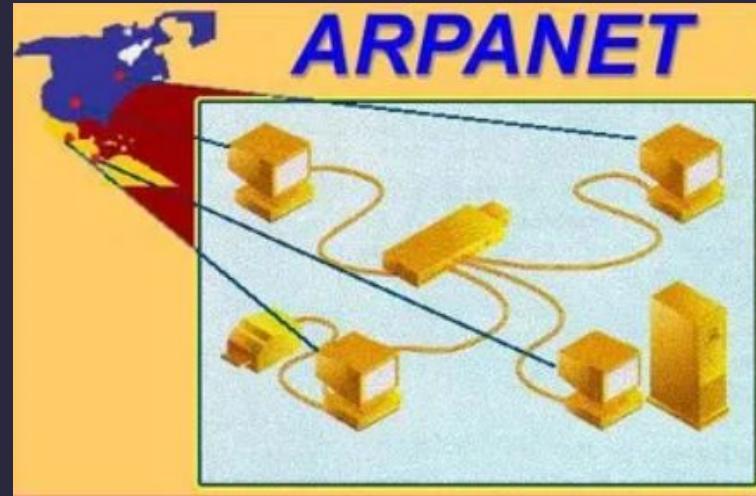
- > The first commercial computer was born. It is the UNIVAC 1, which was developed by the Howard Aiken Sperry-Rand Corporation and purchased by the U.S. Census Bureau.

# First programming language created



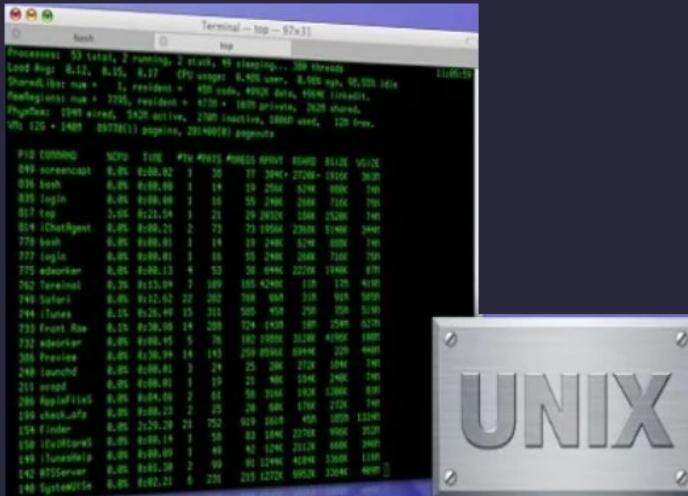
- > John Backus and his colleagues at IBM created the first Fortran, the first programming language.

# ARPA appears



- ARPA is an agency of the U.S. Department of Defense. There
- J.C.R. Licklider successfully defends his ideas about a global computer network (Beginning of the Internet)

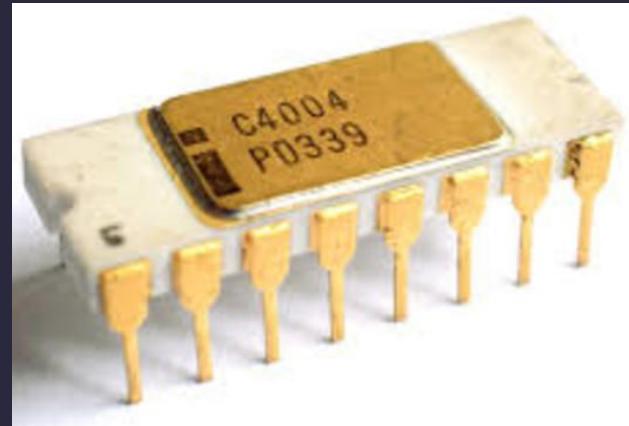
# UNIX



› Creation of the UNIX operating system



# First silicon microprocessor



- > Intel manufactures and brings to market the first silicon microprocessor. It is the Intel 4004. It was created to perform Babbage's basic operations according to the Von Neumann architecture. It is the first CPU

# Apple's first computer



- It is the beginning of what today is Apple. Steve Jobs and Steven Wozniak wanted to manufacture and market a computer for mass use.

# First Desktop Computer



- Commodore develops and markets the world's best-selling desktop computer

# First "Personal Computer" (PC) is born



- > The first of the "Personal Computer" is born. The first of the IBM PC compatible hardware platform. The IBM 5150. In addition, the TCP/IP protocol and the word Internet are defined.

# First Server and first DynaTAC 8000X cell phone created



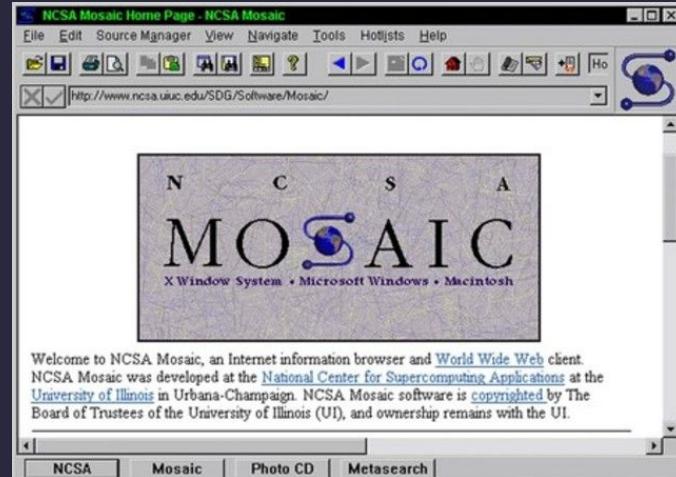
- The 1st site name server is created. Motorola introduces the first cell phone. The DynaTAC 8000X

# A World Wide Web prototype is created



- 100,000 computers connected to the Internet and the World Wide Web is prototyped

# The Mosaic Internet browser appears



- The Internet browser Mosaic, the first Internet search engine, appears, called Wandex. The IBM Simon becomes the first cell phone to integrate PDA functions.

# Internet search engines appear

- In 1994 the Internet search engines WebCrawler, Lycos and Excite appeared. And in 1995 Internet AltaVista and Yahoo



# 10,000,000 computers connected to the Internet



- 10,000,000 computers connected to the Internet. The Nokia 9000 Communicator becomes the smartphone and integrates an Intel 386 CPU.

# Google

› Google makes its appearance



# First Digital book



› The 1st digital book appears

# First social network LinkedIn



- LinkedIn is launched as the first professional social network in history. This year also saw the appearance of MySpace and Hi



# Facebook and other social networks appear



- > Facebook, Flickr, Vimeo, Tagged and Google's first social network, Orkut, appear on the scene

# Youtube is born



- › YouTube, DailyMotion and Reddit are created

2005

# Iphone appears



- › YouTube, DailyMotion and Reddit are created

Whatsapp appears



- › WhatsApp and Microsoft Bin Internet browser created

# Timeline of Computer History



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

# 02

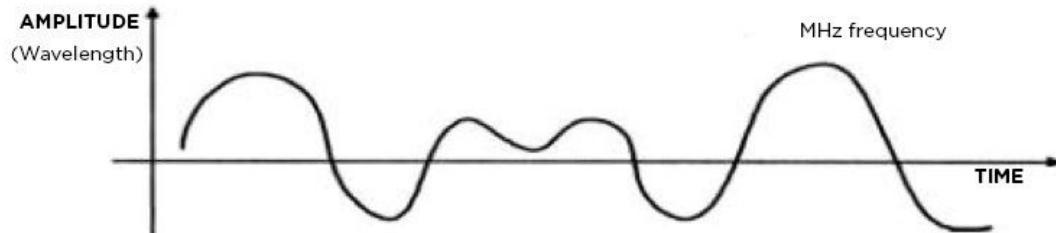
# Principles

basics

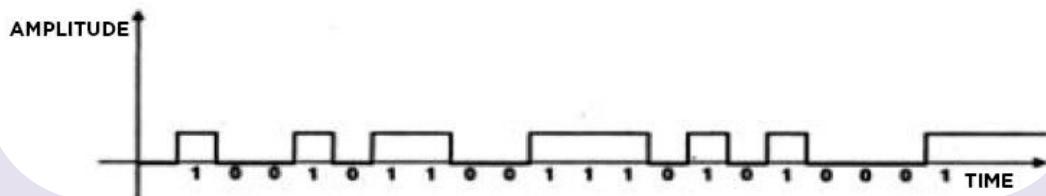


# Analog vs Digital

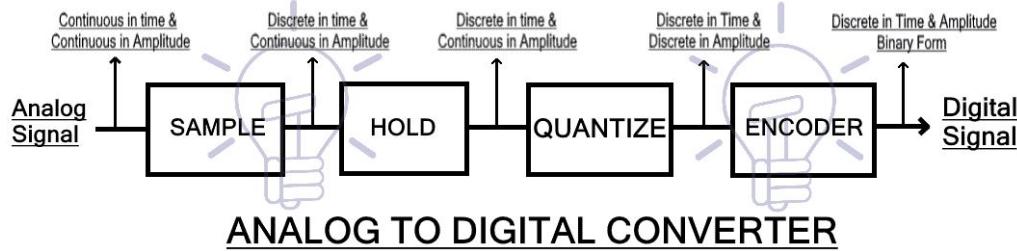
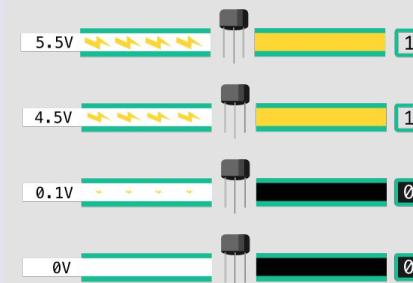
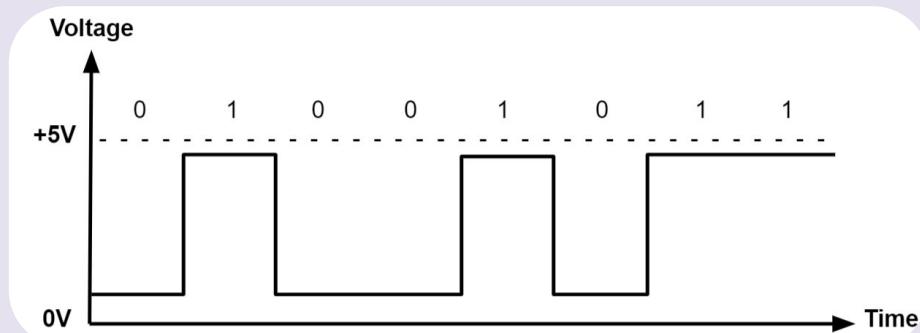
## ANALOG SIGNAL



## DIGITAL SIGNAL



# Analog vs Digital





# Bit vs Byte

- **64 or 32 bits**
- **1024 megabytes / 1024 megabits**
- **1024 megabytes / 512 gigabytes**

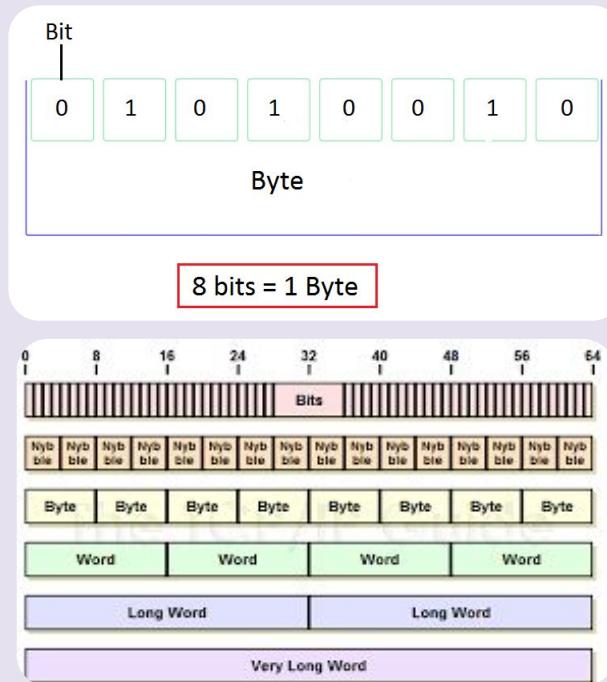




# Bit vs Byte

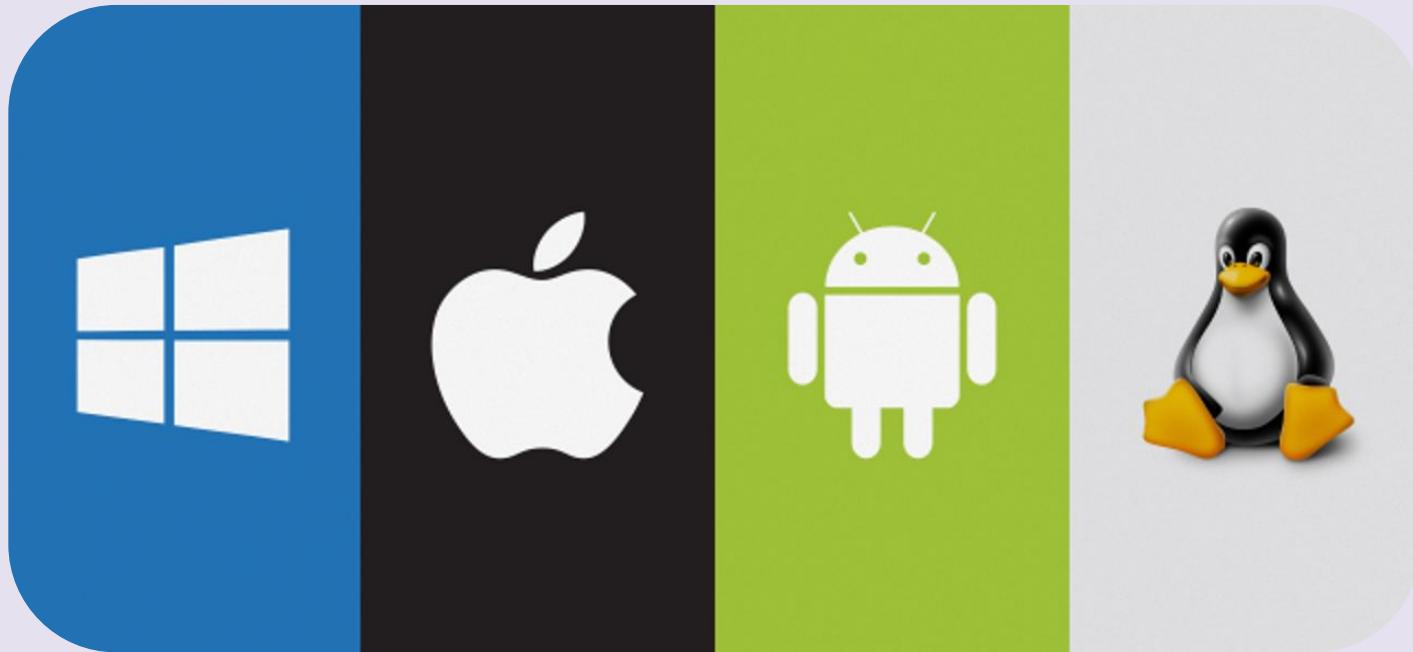
- The **bit (b)** is the **basic unit** in which information is measured in the computer environment.
- The **byte (B)** is a unit of measurement that is composed of **8 bits**, it is an **octadecimal base unit**.

In contrast to what happens with the other units that we usually handle, such as the meter, the gram, the liter whose base is decimal.





# Operating Systems





# Operating Systems

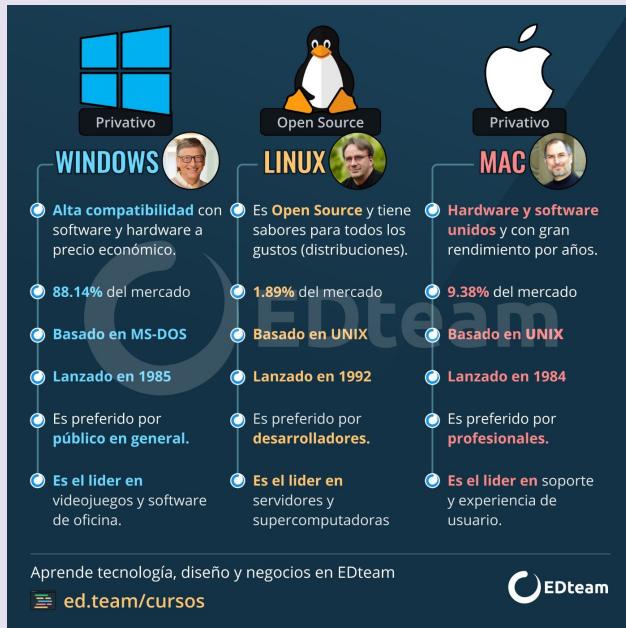


# Operating Systems

There are 3 main operating systems for computers:

- Windows
- Mac
- Linux

Android and IOS are also based on unix. You can say that phone OS are variations of computer OS.





# Curiosities

Are there viruses for MAC?

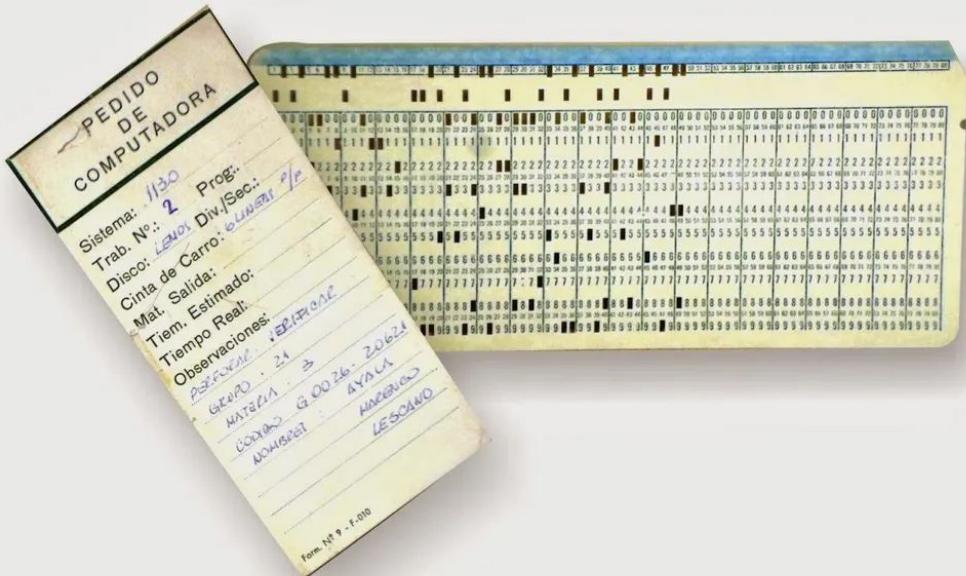
Does the MAC have antivirus?

Are there viruses for an Iphone?





# Older programs





# Old programs

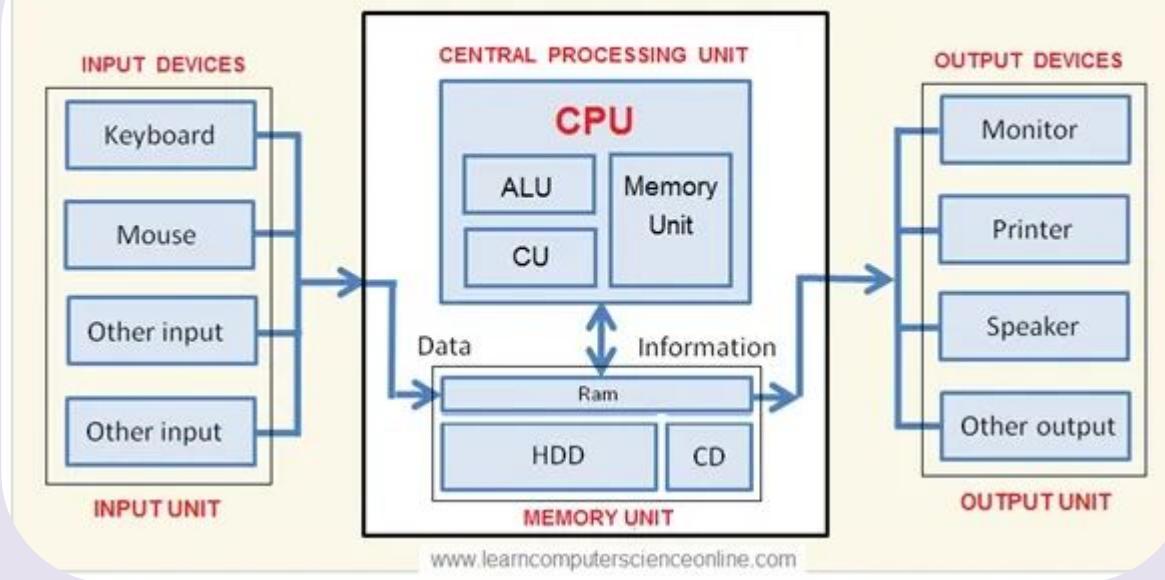
Assembly language is the programming language used to write **low-level computer programs**, and is the **most direct representation of the machine code** specific to each **computer architecture** readable by a programmer.

```
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
```



# Older programs

Computer System Block Diagram



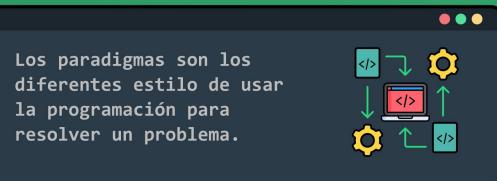
# Programming Paradigms

A programming paradigm is a **way** or **style of programming software**. There are different ways of **designing** a programming **language** and various ways of working to obtain the results that programmers need. It is a set of **systematic methods applicable** at all levels of program design to solve computational problems.

Python or JavaScript, which are **multi-paradigms**

## ¿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](https://ed.team/cursos/paradigmas)



# Programming Paradigms

It allows **separating the different components** of a program, thus simplifying its creation, debugging and subsequent improvements. Object-oriented programming **reduces errors** and **promotes code reuse**. It is a special way of programming, which is somewhat close to how we would express things in real life.

Examples of object-oriented programming languages would be Java, Python or C#.





# Internet

---

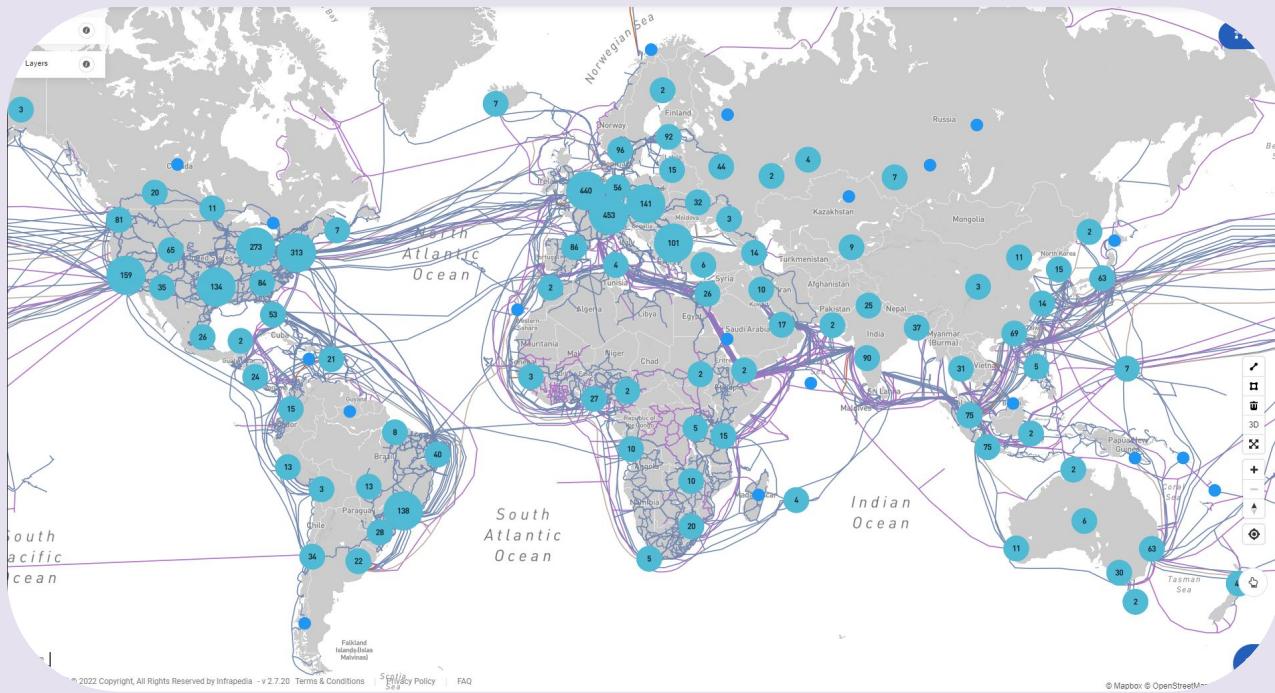


# Internet

Global, **decentralized** computer network, formed by the **direct connection between computers** by means of a special communication protocol. - RAE



# Internet view



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



# Origin of the Internet

There are two versions:

1. The most popular one points to its creation as a response from the US Department of Defense, who in the 1960s were looking for a way in which all the computers being used within the organization would work in a network, even if one of the computers suffered a failure due to an enemy attack.
2. The Information Processing Technologies Office (IPTO), a man named Robert Taylor (who was just becoming the director of the office) came up with the idea of creating a system that would allow researchers to share resources through the use of links.



# Internet

In 1969 the U.S. Department of Defense began to look for communication alternatives in the event of an atomic war. Three years later, the first demonstration was carried out between 4 universities (3 in California and 1 in Utah) by establishing a connection known as ARPANET (Advanced Research Projects Agency Network).

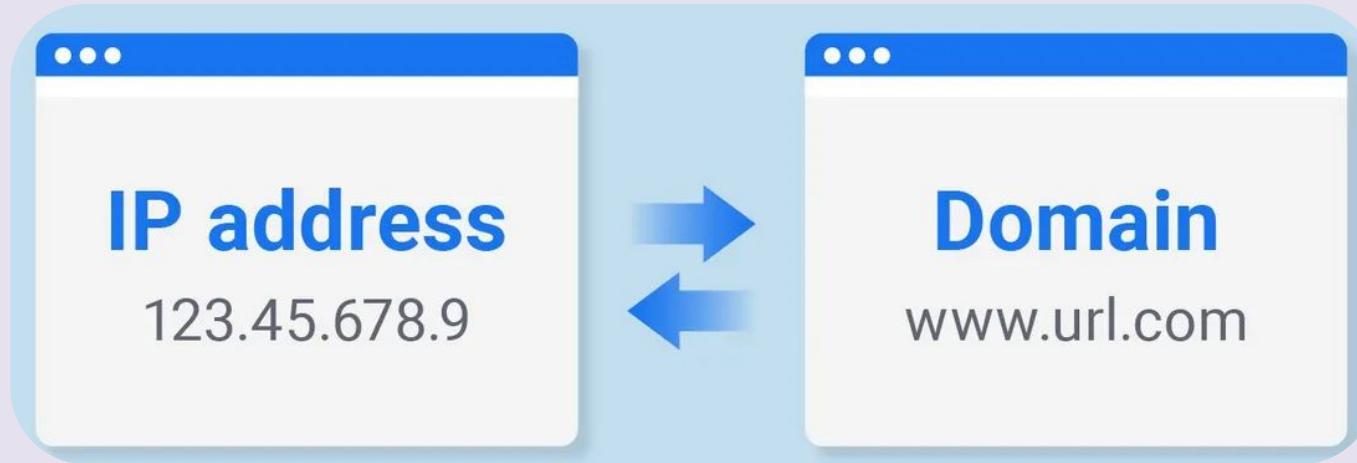




# IP - Internet Protocol



# 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
```

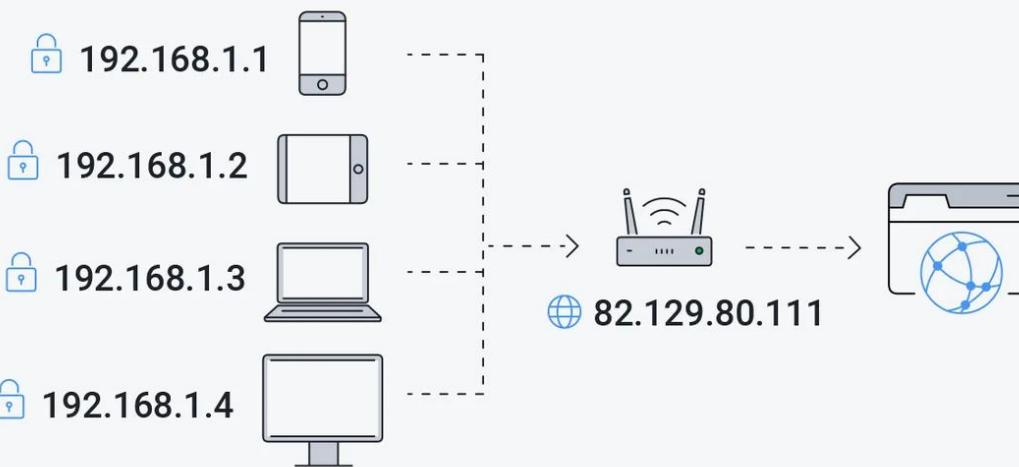


# IP - Internet Protocol





# IP - Internet Protocol

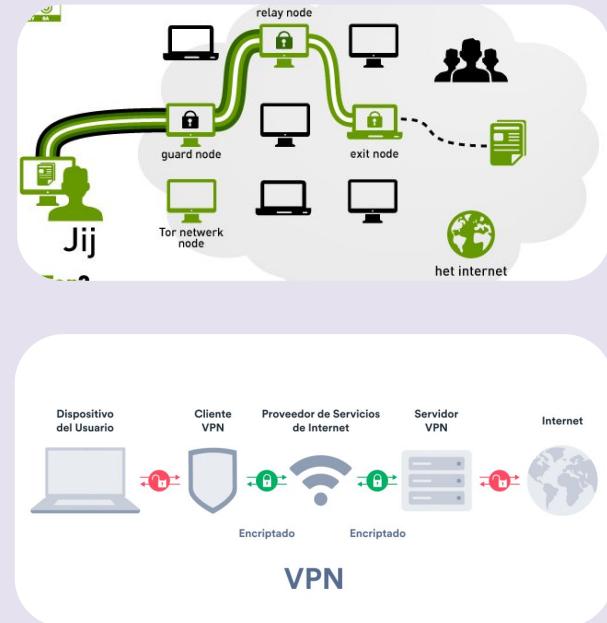


## Public IP vs Private IP

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



# Curiosity



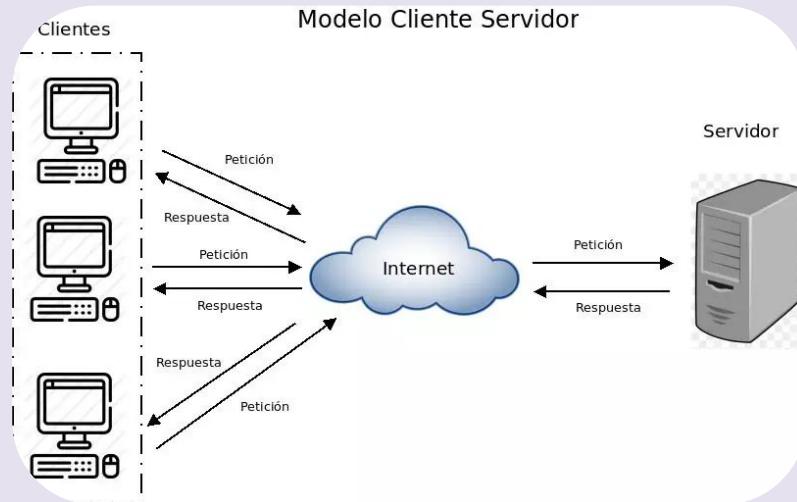


# Client Server

# Client Server

The client-server model is a distributed **application structure** that **divides tasks or workloads** between the **providers of a resource** or service, called servers, and the **service requesters**, called clients.

That is, the client **makes** service **requests** to the server, which is responsible for **fulfilling the requests**.



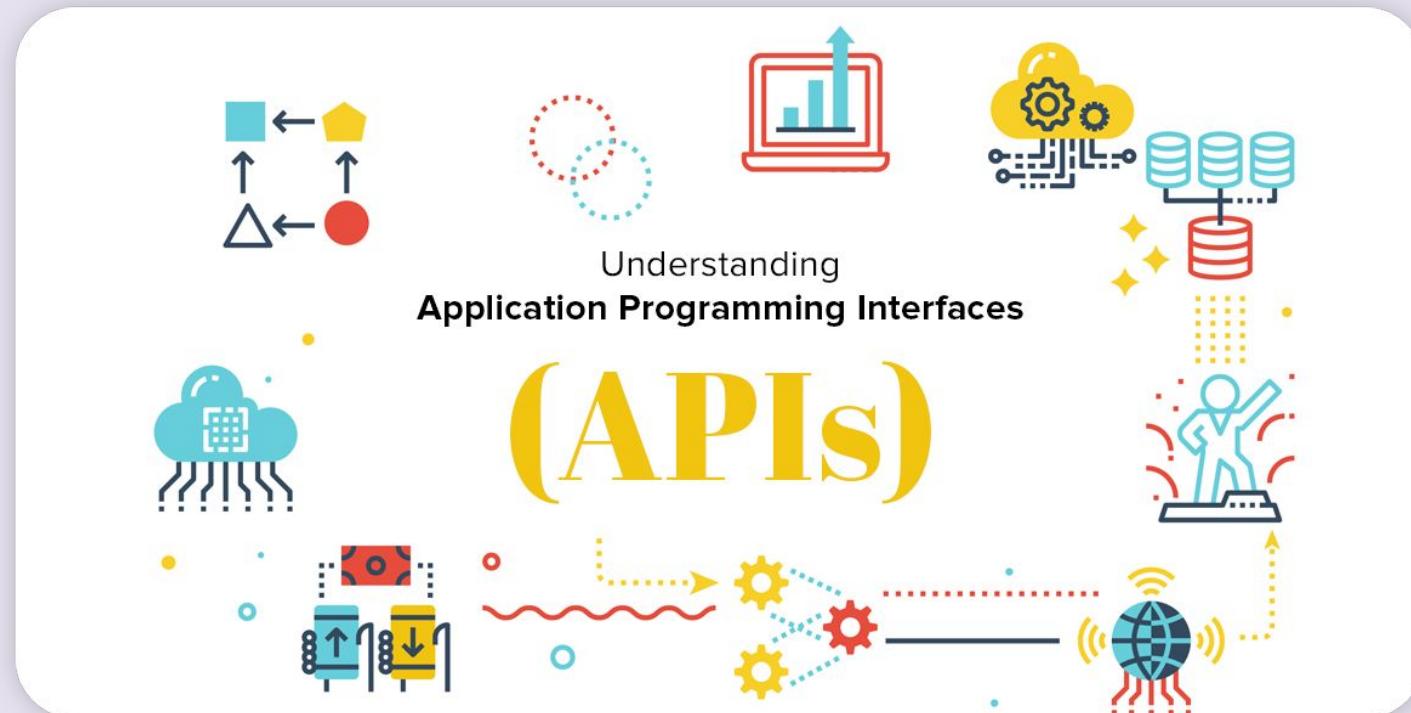
# Curiosities

- What happens if the server goes down?
- How do I fix it?
- Is there anything else?





# API - Application Program Interface



# → API - Application Program Interface



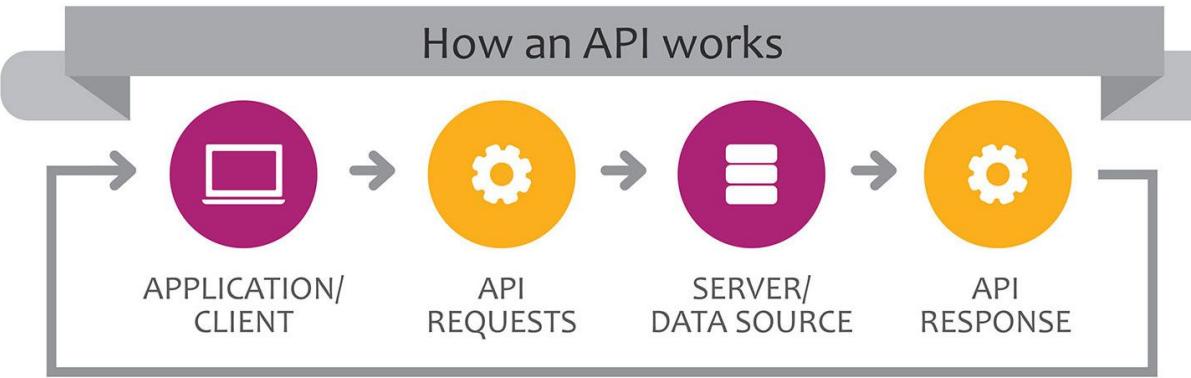
## The API

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



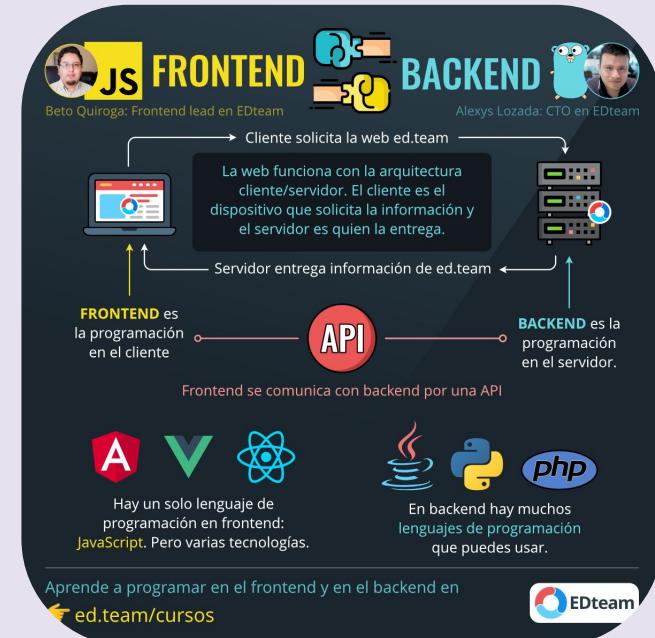
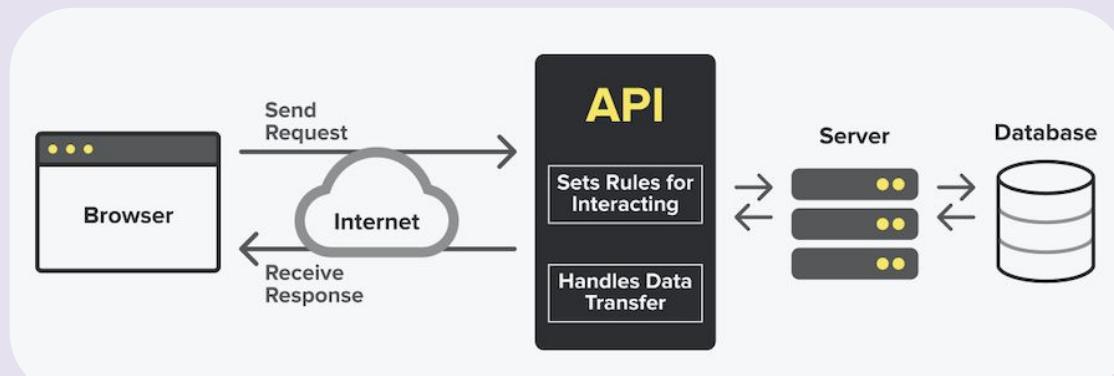
APPLICATION/  
CLIENT

API  
REQUESTS

SERVER/  
DATA SOURCE

API  
RESPONSE

# → API - Application Program Interface





# Linux





# Linux

It is a Unix-like **operating system** (or family of operating systems) composed of **free and open source software**. GNU/Linux arises from the contributions of several software projects, among which **GNU** (started by Richard Stallman in 1983) and the "Linux" **kernel** (started by Linus Torvalds in 1991) stand out.

Its capacity for **customization** and power to be truly **customizable** for unique software and application requirements is unparalleled. Its benefits include **stability, super-efficient management of memory resources, processors and storage**; its different types of **distributions** for each scenario make it a must-have for hosting the most demanding and peculiar applications or software platforms.

## 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



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



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



**linux mint**

Linux Mint, basada en Ubuntu, intenta dar una experiencia cercana a Windows.

Domina la administración de servidores Linux en EDteam:



[ed.team/cursos/linux](http://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, <a href="#">open-source software</a>	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	<a href="#">graphical user interface</a>
USER EXPERIENCE	requires an relatively experienced Linux administrator	more beginner-friendly
DATABASE SUPPORT	MySQL, PostgreSQL	Microsoft SQL, Microsoft Access
SCRIPT SUPPORT	Python, PHP, <a href="#">Perl</a> , and other Unix languages	ASP and ASP.NET



# Computer components





# Computer components

CPU: Main processing unit

RAM: Random Access Memory

Hard disk

Power supply: Power supply

Motherboard: Motherboard

Cabinet

I.e.

CPU: Does all the processing

RAM: Keeps programs open

Hard Disk: Stores files

Power supply: Provides the electricity

Motherboard: Connects all components

Cabinet: Contains all components





# Types of use

- Ophthalmic

Very basic use such as for offices, doctors' offices, etc. Only requires opening a few web pages and one or two programs.

- Students (Elementary - High School)

Basic use such as running office, opening various web pages, performing tasks and opening various programs.

- Work

Moderate use such as opening multiple programs, saving many documents and using multiple platforms

- Gaming

Intensive use such as running video games that require general processing and graphics

- Compute intensive

Intensive use for performing Deep Learning, complex mathematical operations or requiring a lot of processing or video processing, similar to gaming

---



# Types of use

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



# Recommendations (2023)

- Ophthalmic

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

- Students (Elementary - High School)

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

- Work

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

- Videogames

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

- Compute intensive

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

---



# Curiosities

- Intel - Current: 13 Generations | Minimum purchase gen 10th
- AMD - Current: Serie 7000 | Minimum purchase series 5

## Examples

- i7 13700K → iX = #Processor | XX = Generation| XXX - Processor grade
- R7 7800X → RX = #Processor | XX = Generation| XXX - Processor grade

## NOTE:

Always the processor grade are 3 digits and ends with one or two letters.

---



# 03

# Software Development



# What is it?

# What is it?

Software development refers to a set of IT activities dedicated to the **process of creating, designing, deploying and supporting software**.

There are three basic types:

- **System software** to provide basic functions such as operating systems, disk management, services, hardware management and other operational needs.
- **Programming software** to provide programmers with tools such as text editors, compilers, linkers, debuggers and other tools to create code.
- **Application software (applications or apps)** to help users perform tasks. Office productivity suites, data management software, media players and security programs are some examples. Applications also refers to web and mobile applications





# What is it?

- **Programmers, or coders**, write source code to program computers to perform specific tasks such as merging databases, processing online orders, routing communications, performing searches, or displaying text and graphics. Programmers typically **interpret instructions from software developers and engineers**.
- **Software engineers** apply **engineering principles to create software and systems to solve problems**. They use modeling language and other tools to devise solutions that can often be applied to problems in a general way, rather than simply solving only a specific instance or customer. Software engineering solutions adhere to the **scientific method** and must work in the real world. **Their responsibility has increased as products** have become increasingly **intelligent** with the addition of microprocessors, sensors and software.
- **Software developers** have a less formal role than engineers and may be closely involved in specific areas of the project, including writing code. At the same time, they **drive the overall software development lifecycle** by working in functional teams to **transform requirements into features, managing development teams and processes, and performing software testing and maintenance**.

# Why does it arise?

IBM Vice President and blogger Dibbe Edwards notes, "Software has emerged as a **key differentiator in many products**, from automobiles to washing machines to thermostats, with a growing Internet of Things connecting them."

Companies that use software development have **cleaner, faster and more effective processes** than companies that use entirely manual processes, since **response times, downtime and human error are eliminated**.



# → Steps in the software development process

1. Select a methodology
2. Gather requirements
3. Choose or create an architecture
4. Develop a design
5. Create a model
6. Create code
7. Perform testing
8. Manage configuration and defects
9. Deploy
10. Migrate data
11. Manage and measure the project

Summary of steps:

- 
1. Requirements analysis and specification
  2. Design and development
  3. Testing
  4. Deployment
  5. Maintenance and support

# → Software development methodologies



---

# 04

# Applied technologies





# Programming language

A programming language is a formal (or artificial, i.e., a **language with well-defined grammatical rules**) language that provides a person, in this case the programmer, with the **ability to write (or program) a series of instructions or sequences of commands in the form of algorithms** in order to control the **physical or logical behavior of a computer system**, so that various kinds of data can be obtained or certain tasks can be executed. This whole set of commands written in a programming language is called a computer program.

The infographic is titled "LENGUAJES DE PROGRAMACIÓN ALTO NIVEL VS BAJO NIVEL". It features a comparison between two code snippets: C++ (High-level) and Assembly (Low-level). The C++ code is:

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

The Assembly code is:

```
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
```

On the left, under "ALTO NIVEL", there is a blue icon of a head with code symbols inside, and the text: "Es un lenguaje que entienden los humanos." On the right, under "BAJO NIVEL", there is a green icon of a computer monitor with binary code (11001000000011010001010101010101) above it, and the text: "Son instrucciones para el procesador."

Below the code examples, there are four bullet points:

- Está orientado al software.
- Utilizan **menos instrucciones** para realizar una acción.
- Te permite programar **aplicaciones y videojuegos**.
- Puedes construir **sistemas operativos y núcleos**.

Icons for a smartphone, game controller, laptop, and monitor are shown next to their respective bullet points.

At the bottom, the URL [ed.team/programacion](http://ed.team/programacion) is displayed, along with the EDteam logo.



# Programming language



# **TYPES OF LANGUAGES**



# Programming language



# Programming language





# Types of typing

A typed and untyped programming language

- The **typed languages** are those that **necessarily** have to **define the type of data or variable**. C#, C, C++ and Java are some examples.
- **Non-typed languages** are those that **do not need any definition** and take the type of the value assigned to them. JavaScript, Python and PHP are some examples

Typed = Static | Untyped = Dynamic

Typed = Strong | Untyped = Weak

## LENGUAJES DE PROGRAMACIÓN

### TIPADOS VS NO TIPADOS

Los lenguajes estáticamente tipados requieren declarar las variables con su tipo de dato.

Los lenguajes dinámicamente tipados declaran variables sin necesidad de definir el tipo de dato (el intérprete infiere el tipo).

**TIPOS DE LENGUAJES**

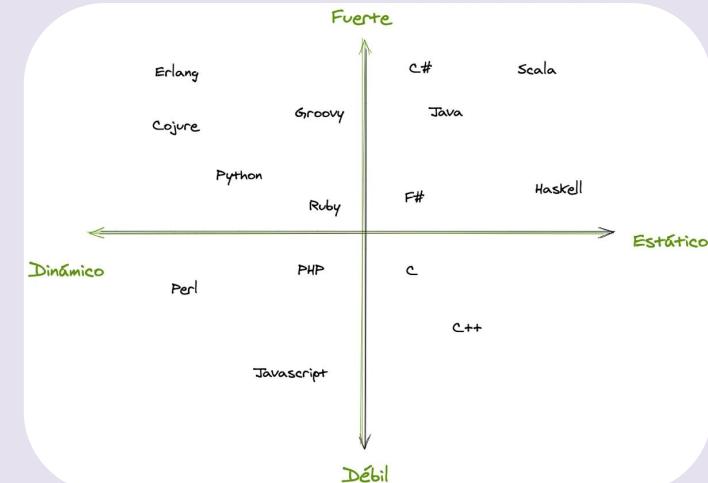
- TIPADOS**:
  - Más verbose, pero menos propenso a errores de sintaxis (el compilador los detecta).
  - Se puede saber qué tipo de dato retorna una función.
  - No se puede cambiar el tipo de dato después de declarada la variable.
- NO TIPADOS**:
  - Código más legible y curva de aprendizaje más sencilla.
  - No sabes qué tipo de dato retorna una función.
  - Se puede cambiar el tipo de dato después de declarada la variable.

Dominas cualquier lenguaje desde cero en:  
[ed.team/programacion](http://ed.team/programacion)



# Types of typing

- **Weak typing** allows you to work in **less time** and to check the results of the program in real time. For the **ideation phase** of the project, this option is ideal. It adapts very well to programs of smaller capacity.
- **Strong typing** allows you to perform operations with different types of variables. The direct consequence is that you **will not make as many mistakes** as with the language of the previous point. However, you need to **write more code**. This language is ideal for **larger projects** with a **larger number of specifications**.





# Paradigms

A paradigm is like a map in which there are many paths to reach the same end. There are languages that adopt a specific paradigm and there are others that adopt more than one paradigm such as JavaScript, Python and PHP.



# OOP - Object Oriented Programming

It allows **separating the different components** of a program, thus simplifying its creation, debugging and subsequent improvements. Object-oriented programming **reduces errors** and promotes **code reuse**. It is a special way of programming, which is somewhat close to how we would express things in real life.

Examples of object-oriented programming languages would be Java, Python or C#.





# OOP - Object Oriented Programming

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

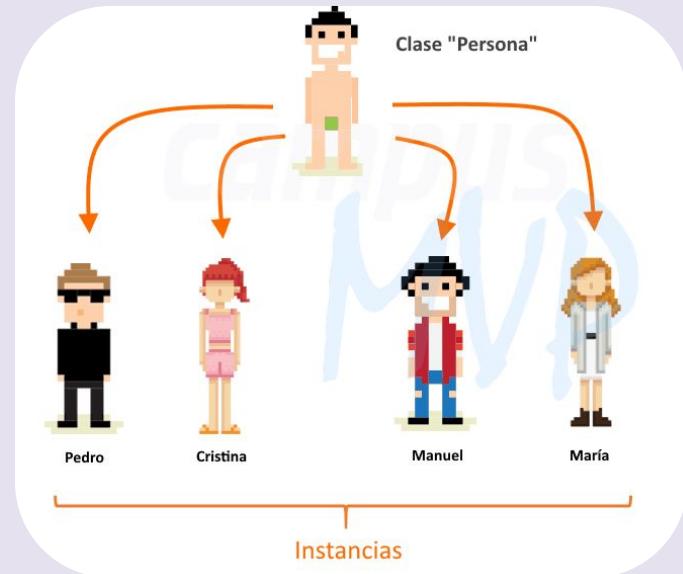
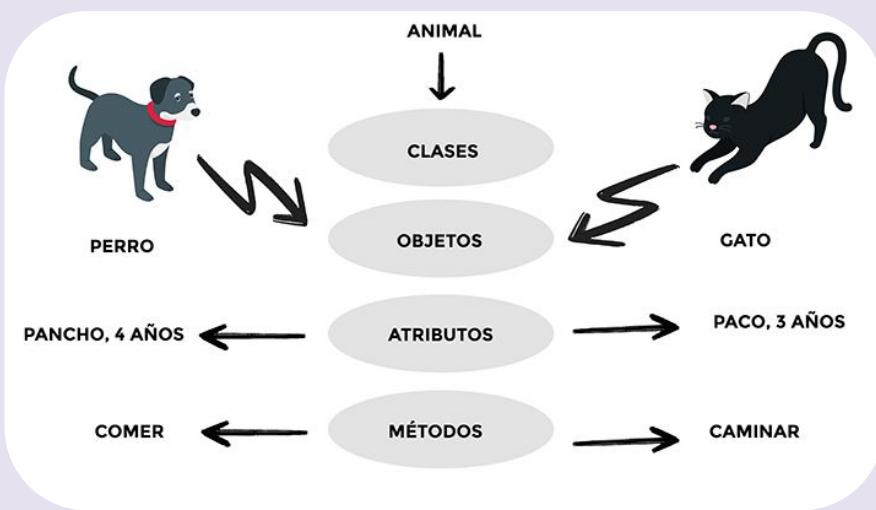
Más difícil de depurar

Una curva de aprendizaje más sencilla

Optimizado para programas sencillos



# → OOP - Object Oriented Programming



# → Development types

## RAMAS DE LA PROGRAMACIÓN

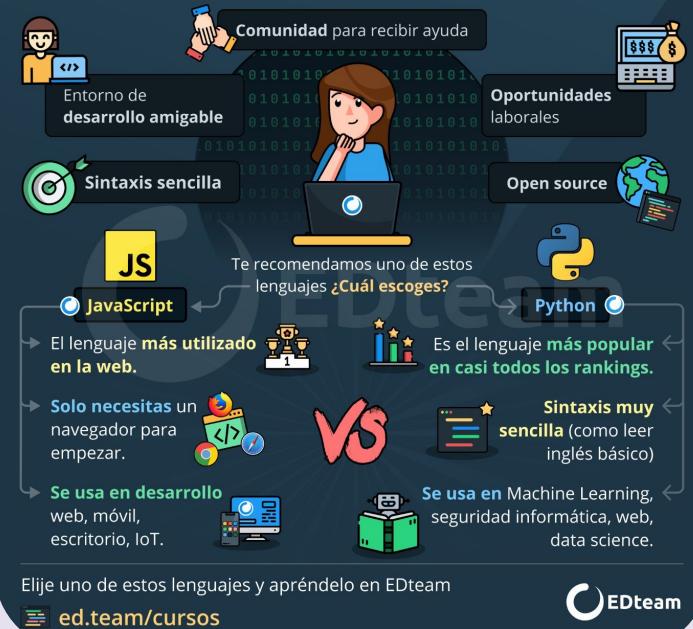


Aprende a programar en cualquier lenguaje (primer curso gratis) en:

👉 [ed.team/programacion](https://ed.team/programacion)



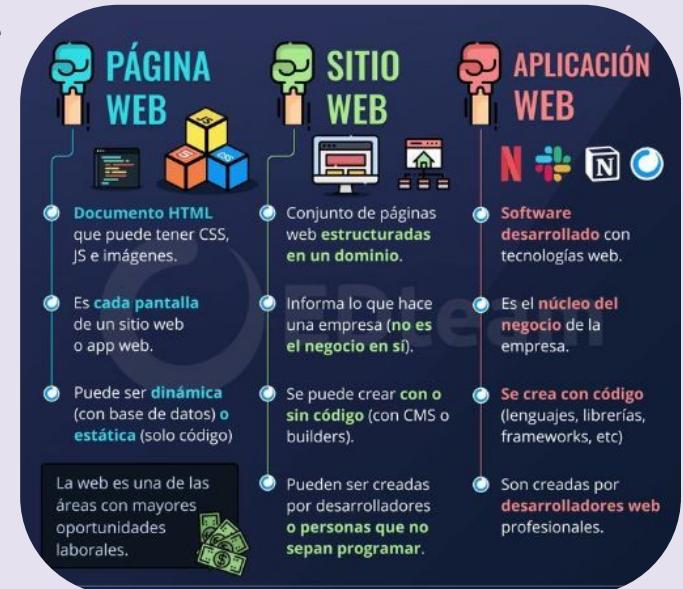
## ¿CON QUÉ LENGUAJE APRENDER A PROGRAMAR?



# Web Development

Web development is a term that defines the **creation of websites** for the **Internet or an intranet**. To achieve this, use is made of **server-side** and client-side software technologies that involve a combination of database processes with the use of a **web browser** in order to perform certain tasks or display information.

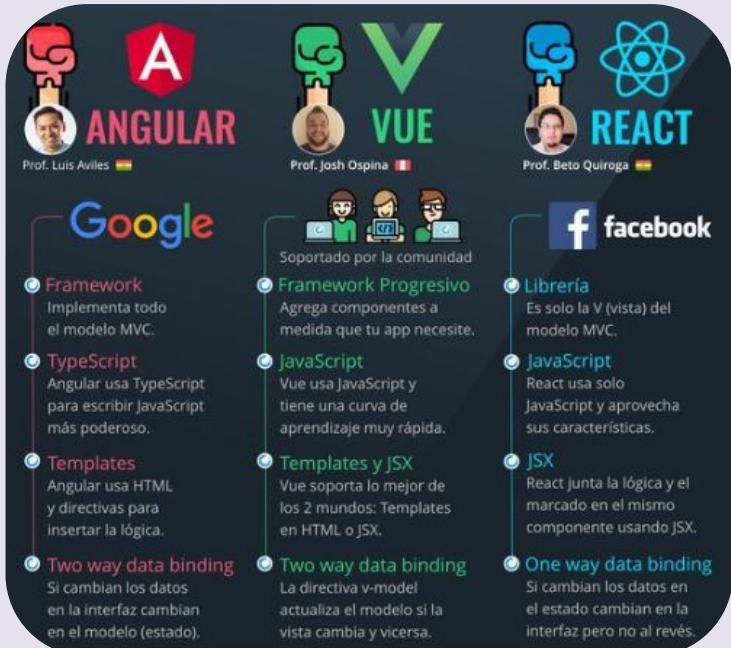
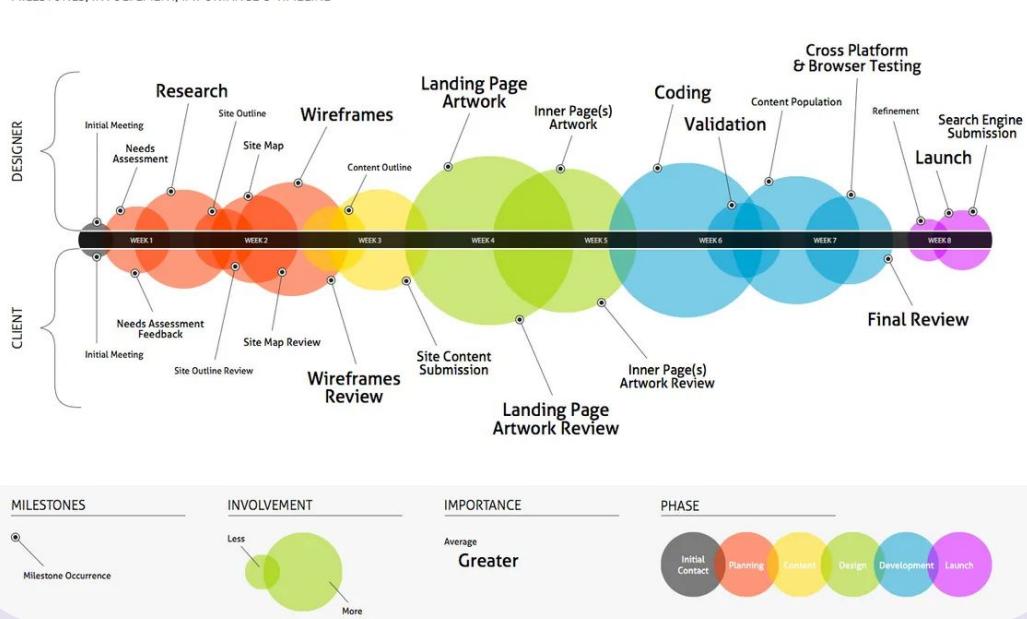
Functionally, the **web developer**, who is the one who performs this work, is usually only concerned with the **operation of the software**, it is the task of the web designer to worry about the final appearance (layout) of the page and the webmaster to integrate both parts.



# Web Development

## A Web Site Designed

MILESTONES, INVOLVEMENT, IMPORTANCE & TIMELINE

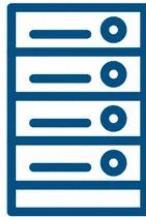


# Web Development



## Front End

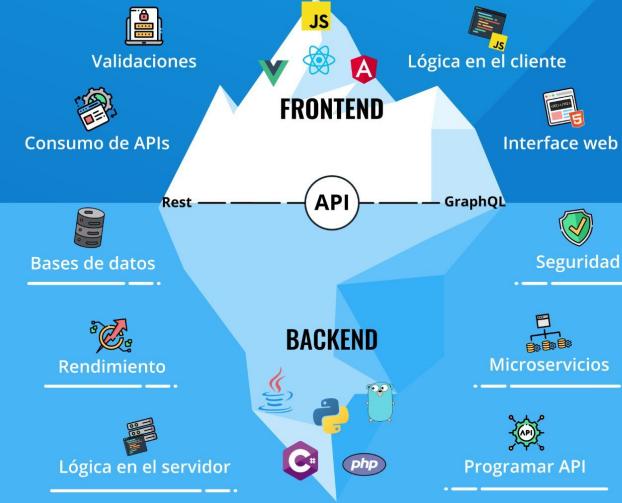
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



## Back End

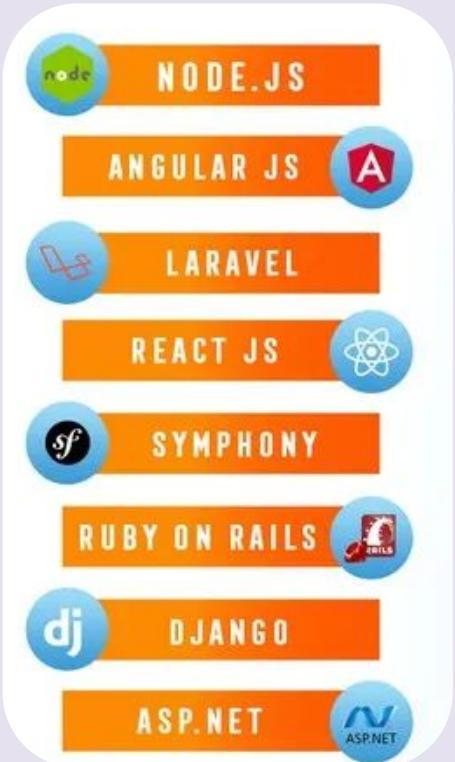
- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

## ¿QUÉ ES BACKEND Y FRONTEND?



Domina las tecnologías Backend y Frontend en:  
[ed.team/cursos](http://ed.team/cursos)

# Web Development



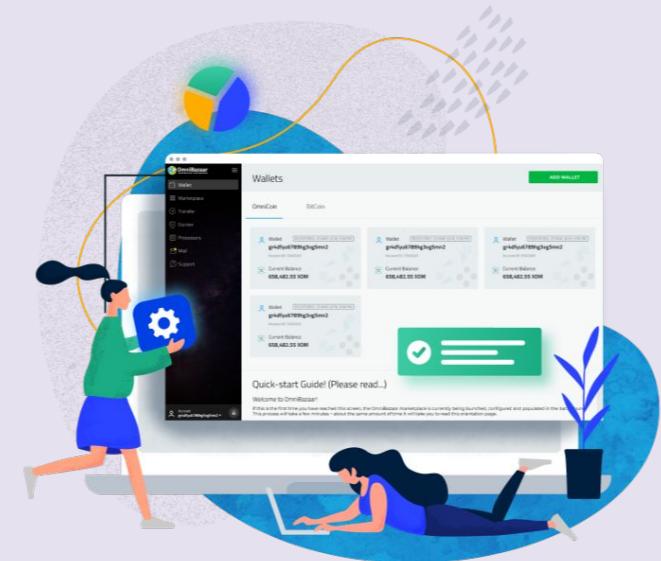


# Desktop Development

Desktop applications are the **set of programs or tools that we have installed on our** desktops or laptops and that **we can only use on those devices**. We cannot move them to a different device, which is why they have this limitation.

On the contrary, web applications, which have their presence in the cloud, do not have this portability problem.

Examples of desktop applications could be OpenOffice, Excel or Photoshop.





# Desktop Development

Among the most notable **advantages** of desktop applications are:

- Data is centralized.
- They are much more stable and robust programs.
- Data loading is faster.
- If the Internet is down, do not worry because you will still be able to use it.
- They are usually cheaper.
- Security is higher in these applications.
- You can make backup copies at any time.





# Desktop Development

Starting with the **disadvantages** or defects that can be put to the desktop applications are:

- They require installation on each client or device.
- After all, they are developed for a specific operating system. If you have a different system, you have to wait for them to develop the version for yours.
- An update is required on each device. You can't push an update to everyone with that application.





# Desktop Development

The most known and used languages are Java and Python, but as we are going to see there are many more.

- In Windows we have Visual C++ and Visual Basic.
- C/C++ with Qt or GTK.
- Java with AWT or Swing.
- On Mac we have Objective-C/Swift with Cocoa.
- On Linux we have C/C++ with Qt or GTK.





# Mobile development

Mobile application development is the set of processes and procedures involved in writing software for **small, wireless** computing devices, such as **smartphones** and other handheld devices.

Mobile applications **are often developed specifically to take advantage of the unique features of a particular mobile device**. For example, a gaming application might be written to take advantage of the iPhone's accelerometer, or a mobile health application might be written to take advantage of a smartwatch's temperature sensor.





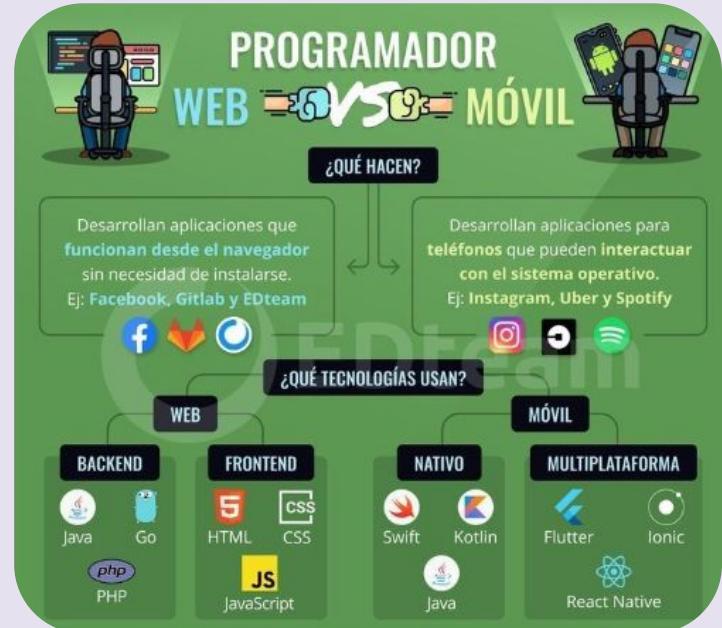
# Mobile development

- **Native applications.** These applications are created using integrated development environments (IDE) and languages for mobile operating systems such as **Apple iOS or Google Android**. Native applications allow you to customize the necessary functions, but can be more expensive than other technologies.
- **Hybrid applications.** These are **web applications** that act like native applications. They are developed using technologies such as **HTML, JavaScript** and Cascading Style Sheets (**CSS**). Hybrid apps are **more cost-effective to develop than native apps** and can be **created faster**, but **they are not as feature-rich** as native apps and are not installed on the device.
- **Progressive Web Apps.** A PWA is a website that looks and **behaves as if it were a mobile app**. These apps are developed using web technologies such as React.
- **Encapsulated apps.** An encapsulated app runs inside a container app. Products such as the drag-and-drop app creation tool **Microsoft Power App** allow less experienced developers to **create a mobile app quickly**. But the lack of isolation from the core operating system, OS lock-in and relative novelty could pose problems.

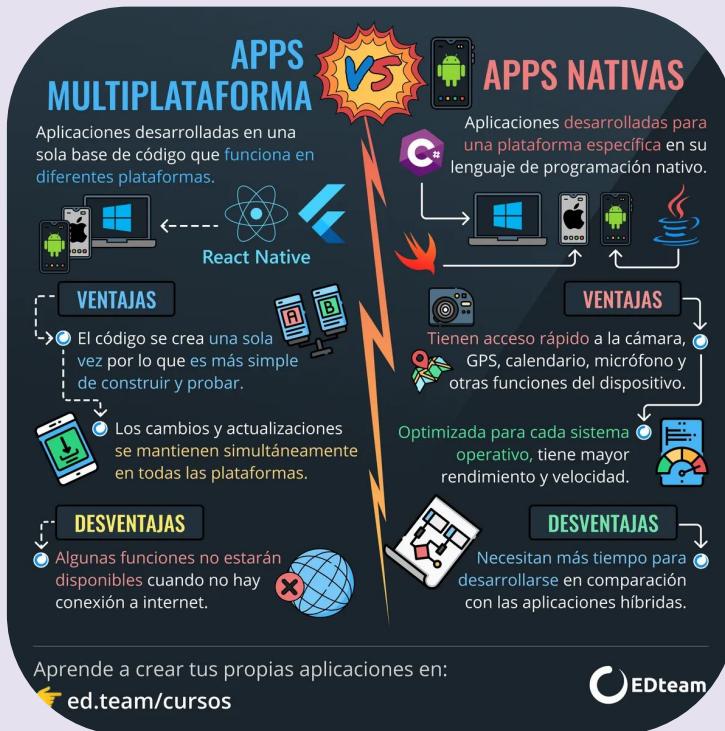
# Mobile development

A **mobile app** refers to a **native application**, created for certain smartphones, which can be downloaded from the corresponding store (each smartphone brand has its own), and **installed on the device's hard disk**.

A **mobile web** is a **web development designed to mimic a mobile app**, but accessed through a web browser. It is a **direct access to the web page** but the aspect it offers is adapted to our mobile device.



# Mobile development





# Cross-platform development

Cross-platform development consists in general in being able to program for multiple devices, it can be through a script, web application or an API in order to be compatible with multiple devices.

It is the most general way for a developer when he is not developing a program for a specific device.



---

# 05

# Roles

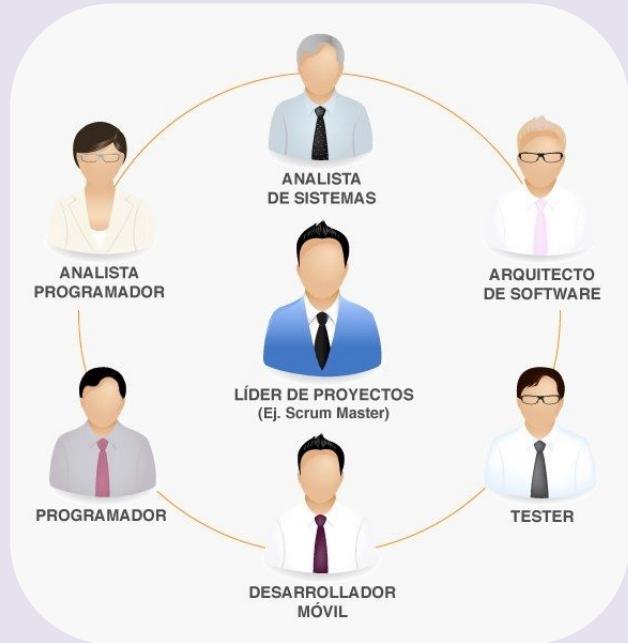
And its activities



# → Roles in Development

It takes more than developers and engineers to form an effective software development team. Many roles are needed in the software development lifecycle, and the ideal team consists of the following roles:

- Product owner
- Project manager
- UX and UI designers (UX/UI)
- Business analyst
- Software developers
- Programmers
- Team and technology leader (team lead or tech lead)
- Scrum master
- Full-stack developers
- Front-end developers
- Back-end developers



# Roles in Development



## 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





# How to choose a role?

The following should be considered when choosing a role:

- Knowledge
- Tools
- Tastes
- Experience
- Skills





# Role Ranks

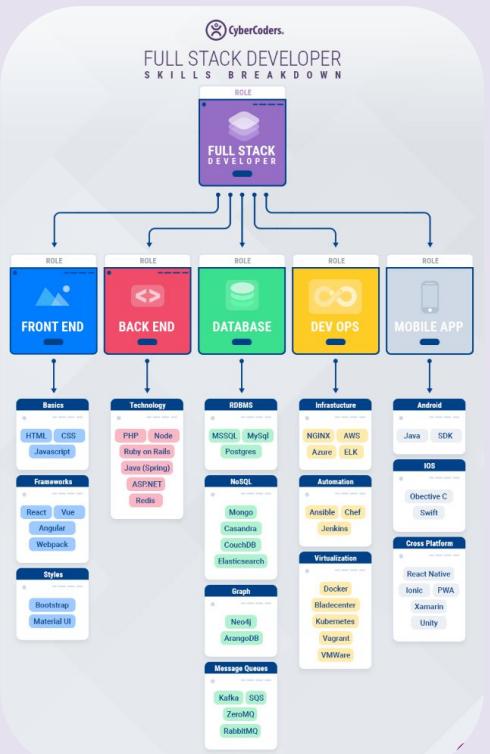




# Role Ranks



# → Roles in the industry



- 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 technical know-how, identify an organization's 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 sentiment analysis 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 Network Administrator**
- A network administrator is responsible for keeping communications and information flowing smoothly by overseeing communication systems and networks.
- Median annual salary: £40,000
- 12 Cybersecurity**
- Today, it is not uncommon for organizations to be breached in order to protect enterprise IT initiatives, and these IT professionals are the ones to do it. As the threat becomes increasingly important, those with experience in data, information, network, system and cloud security are in high demand this year.
- Median annual salary: £57,500
- 13 BI Analyst**
- BI Analysts need 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
- 14 Software Developer**
- Software development positions account for the largest number of job openings in the tech world.
- 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 technicians and engineers.
- Median annual salary: £32,500



# Developer

The main role is the creation and adaptation of software, although this is obviously a somewhat simplistic description. Its scope covers a range of applications, such as programs, processes, networks, version upgrades, patches, migrations, DevOps and testing.

- **Creation of specific code** and subsequent testing.
- Collaboration with customers on required **reports** and **oversight** of the process that turns them into reality.
- Use of a range of development tools to **facilitate the use of processes and systems**.
- Mapping the **design of a software application** and using **flow charts to highlight** each stage of the process.
- Organizing **upgrades and repairs** to existing **software** applications.
- Communicating progress with management through **reports, meetings and presentations**.

**¿QUÉ ÁREA DE LA PROGRAMACIÓN ELEGIR?**

Para comenzar en el mundo de la programación **debes aprender a programar desde cero**. Luego puedes ir a **cualquier lenguaje de programación**, dependerá mucho de la rama en la que quieras especializarte, **estas son algunas que puedes seguir:**

- **Programación desde cero**: A path starting from a person at a computer leading to a terminal window icon.
- **WEB FRONTEND**: Represented by a laptop icon.
- **WEB BACKEND**: Represented by a database icon.
- **JS**: Represented by a yellow square icon.
- **PHP**: Represented by a blue owl icon.
- **Kotlin**: Represented by a purple and orange icon.
- **Swift**: Represented by a red and orange icon.
- **Python**: Represented by a blue and yellow Python logo icon.
- **MACHINE LEARNING**: Represented by a book icon.
- **DESARROLLO MÓVIL**: Represented by icons of an iPhone and an Android phone.
- **Si eres programador nunca te faltará trabajo.**: A speech bubble containing this text.
- **Prof. Álvaro Felipe**: A profile picture and name.

Comienza tu carrera como programador (primer curso gratis) en: [ed.team/programacion](http://ed.team/programacion)

EDteam



# Data development

A **data analyst** needs to **process and interpret** data. A **data scientist** must be able to **create and develop** tools that process information. Let's take a look at each role in a little more depth. In addition, a **data engineer** needs to be able to **create programs or systems** that can **take data and turn it into meaningful information** that can be studied.

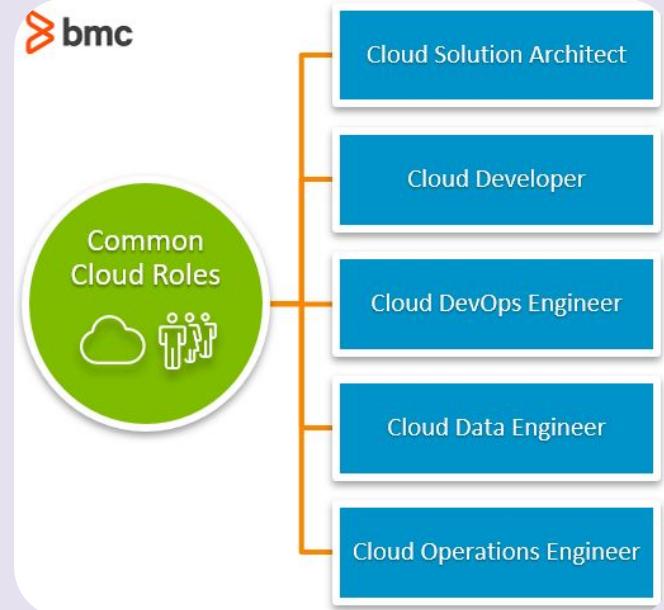


# Cloud development

A **cloud developer** focuses on **developing cloud components and serverless functions** for organizations.

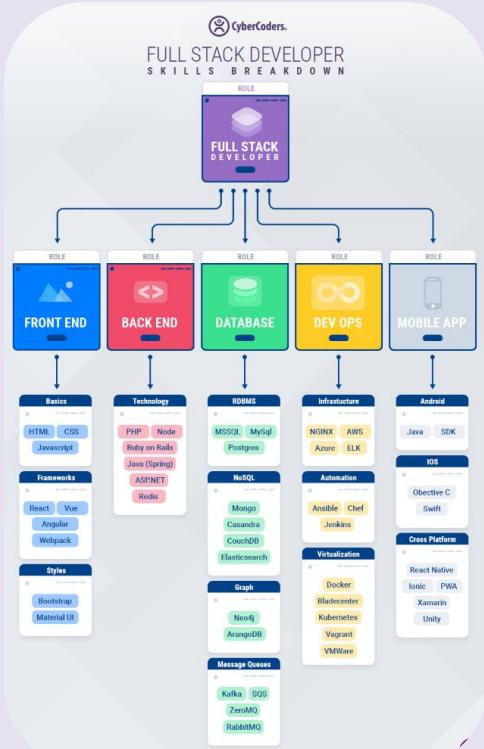
The main role of a cloud developer is to develop scripts and business processes, always trying to leverage cloud functionalities to automate the configuration and deployment process.

As a Cloud Developer, it is **crucial that you understand cloud capabilities**, as well as network and **connectivity** elements (e.g. firewalls, WAF, routing and security, and segmentation). The Cloud Developer is responsible for **monitoring processes and evaluating the resources consumed** in the process of **auto-scaling and deploying** dynamic components.





# Technology used





# Technology used

## 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



# Technology used

## 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

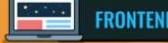
[ed.team/cursos/python](http://ed.team/cursos/python)



## ¿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



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

### BACKEND



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

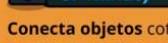


Conecta objetos cotidianos (neveras, televisores, etc.) a Internet.

### ESCRITORIO



Crea apps compatibles para Mac, Windows y Linux con frameworks como Ionic, React Native, Native Script, etc.



### MÓVIL

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



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

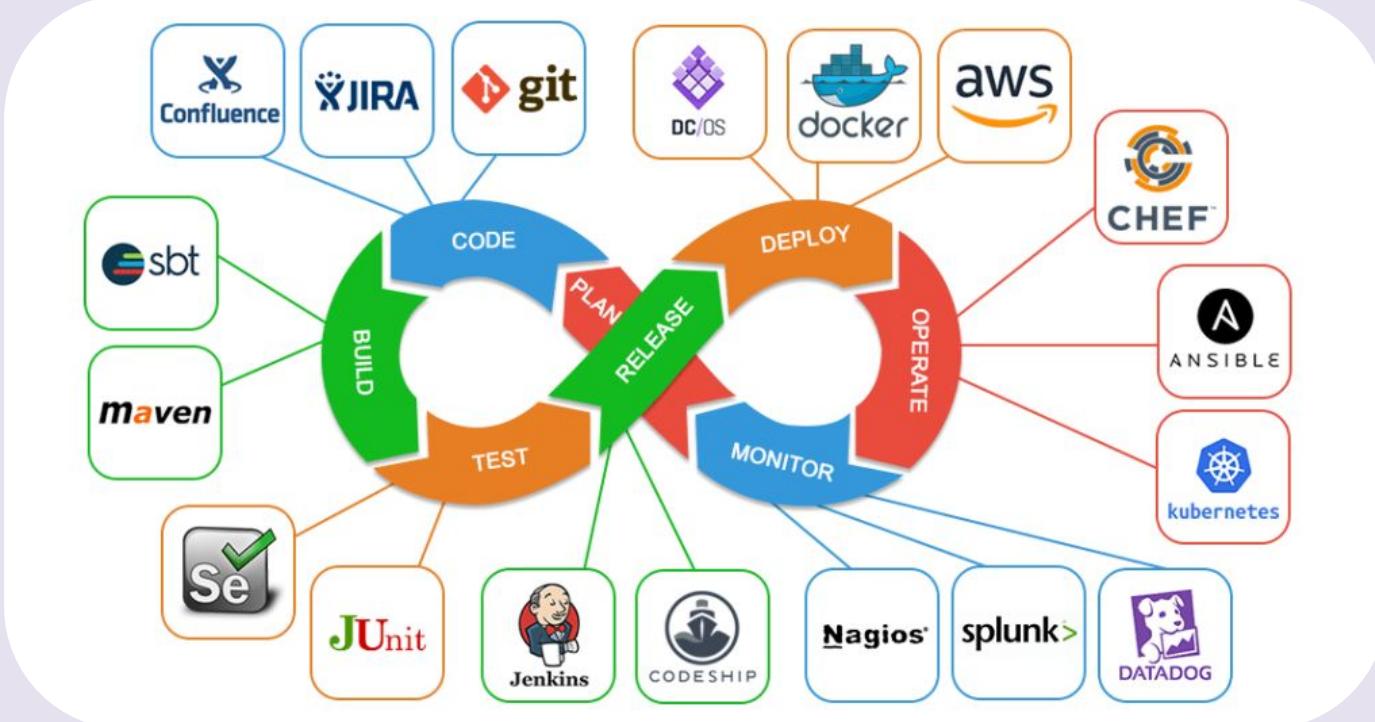
Aprende JavaScript desde cero a avanzado en:

[ed.team/javascript](http://ed.team/javascript)



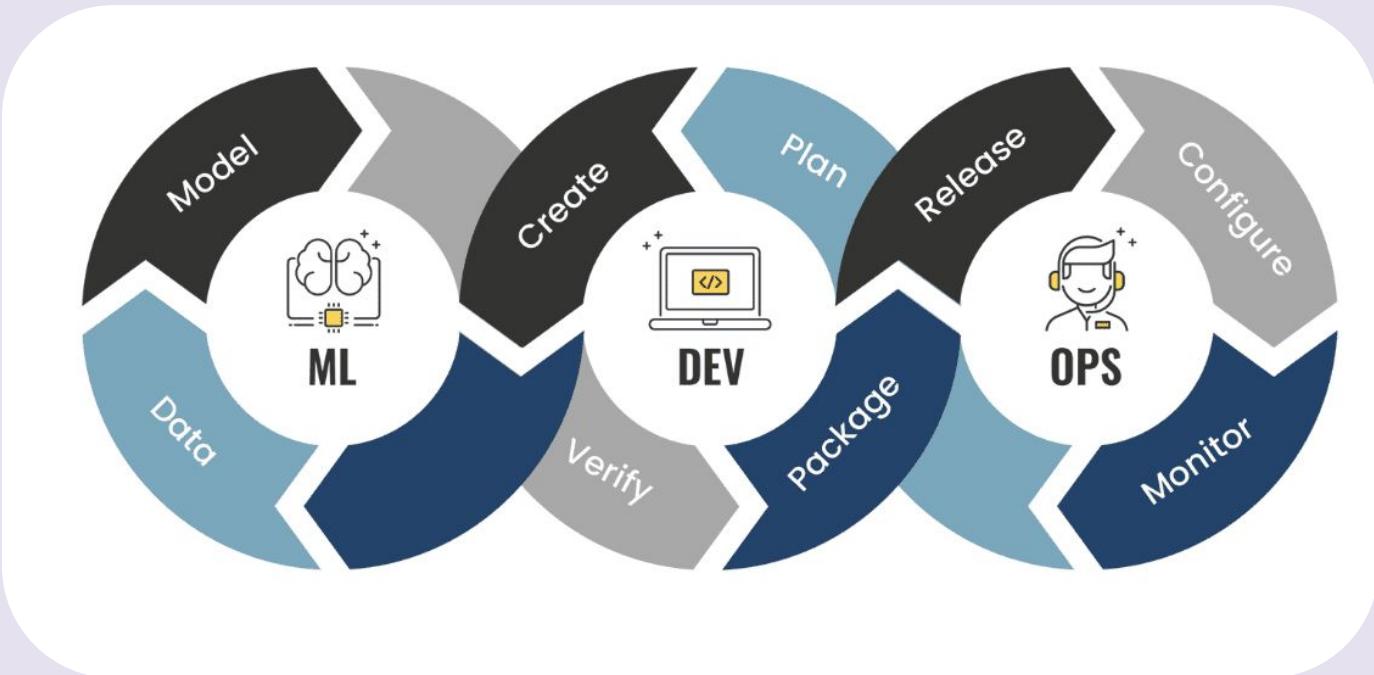


# Tasks to perform





# Tasks to perform



# Wages in Mexico (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

# Wages in Mexico (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

# Wages in Mexico (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

The backend developer works with programming **languages and tools specific to his area**. Although it is **recommended** that he/she knows **the basics of frontend** to be able to communicate with that area, specialized profiles that can perform their work with higher quality are generally sought after.

The work of a backend developer is **not visible to the user** who will use the application or website, since his job is to **support the site and its maintenance** in terms of servers and logic of the digital product.

The developer specialized in **backend is one of the most important when creating a digital product** because he will be **present from its construction to the end**, and even after to provide **maintenance and support to it**.

## ¿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.

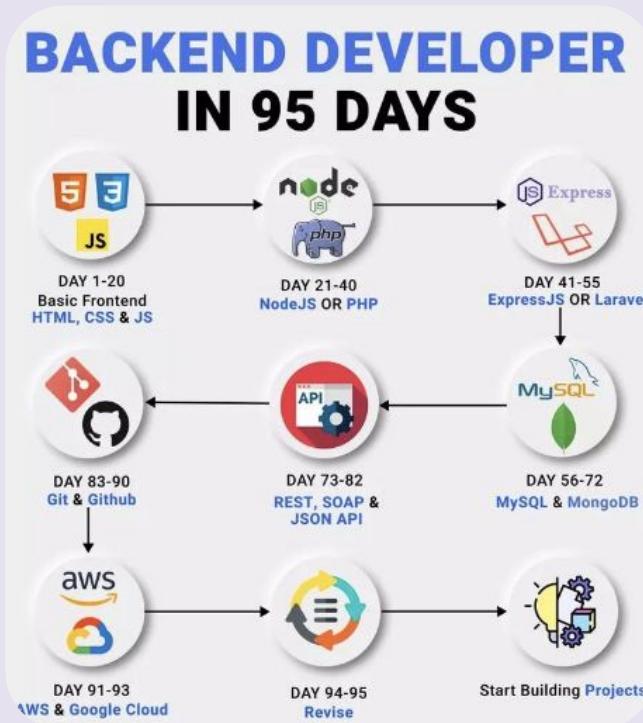
Domina las tecnologías para ser un desarrollador backend en:

 [ed.team/cursos](https://ed.team/cursos)





# Back-End



# Front-End

The frontend developer is in **charge of creating the interface of a website or application through code**.

In the development process of a digital product, the **frontend developer receives the designs created by the UI designer**, and translates them into programming languages.

The frontend developer is one of the pillars that support the creation of a website or application. This **profile tends to be creative**, since before the UI designer profile became more common and necessary, it was the frontend developers who proposed the design of the interfaces.

The development of the interface of **websites or applications is becoming more and more in demand** as, with the acceleration of the digital world, there are more and more products that solve **problems of people's daily lives**.

## ¿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](https://ed.team/frontend)



# Front-End



# FullStack

## Full-Stack DEVELOPER

### Front-End

### Back-end

 HTML5 CSS3

 JavaScript

 Bootstrap

 React

 JQuery

 Angular

 Python

 PHP

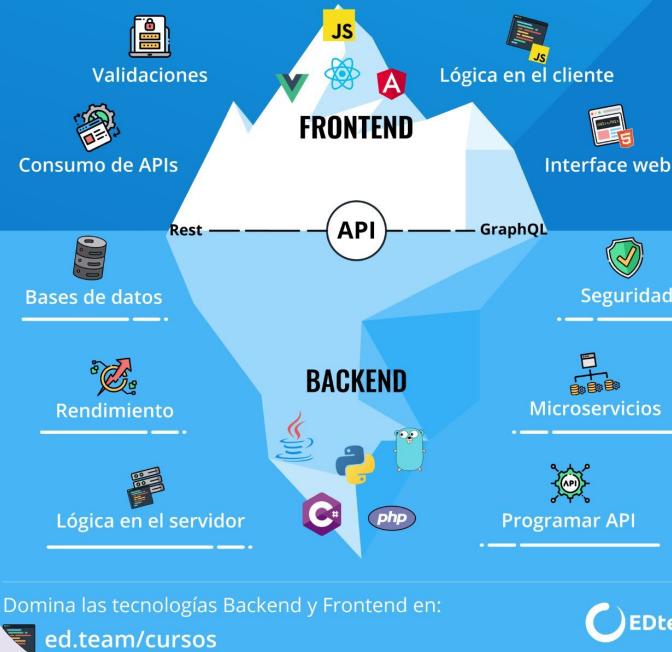
 Java

 SQL

 Django MY-SQL

 MongoDB

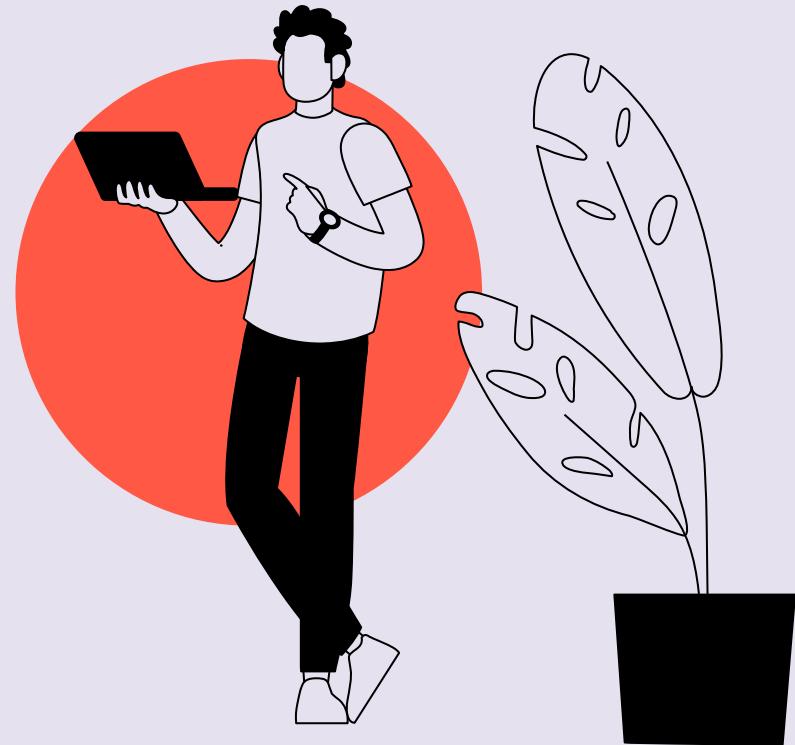
### ¿QUÉ ES BACKEND Y FRONTEND?



---

# 06

# Basic knowledge

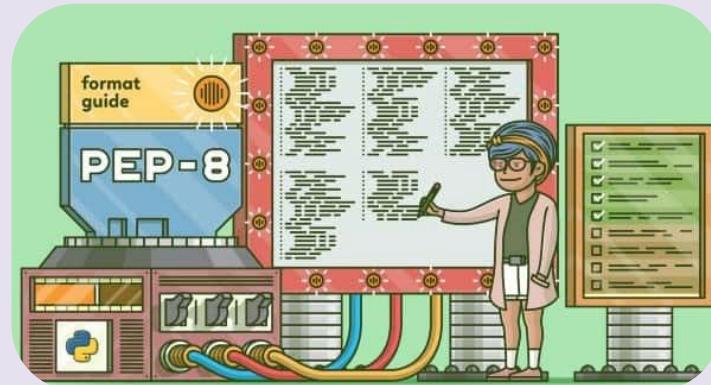




# Standards

**PEPs (Python Enhancement Proposals)** that **govern** different aspects of **Python code production**. The PEPs cover everything from general design principles, such as Python Zen, to conventions for writing documentation, such as the Docstring Convention.

The **standard for JavaScript is ECMAScript (ECMA-262)** and the ECMAScript Internationalization API specification (ECMA-402). The documentation in MDN is based entirely on the latest draft versions of ECMA-262 and ECMA-402. And in some cases where proposals for new features for ECMAScript have already been implemented in browsers, the documentation and some MDN articles may make use of some of these features.





# Standards

What do the standards define or recommend? Some examples are:

- Variable declaration
  - Variable names
  - Variable values
  - Declaration of multiple variables
  - Collection type variables
- Declaration of functions
  - Function names
  - Keys and function spacing
  - Types of functions
  - Arguments of a function
- String construction
- Operations
  - Logical operations of equality and inequality

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



# Functions

A **function** is a **block of code** that performs some **operation**. A function can **optionally define input parameters** that allow callers to pass arguments to the function. A function **can also return a value as output**. Functions are useful for **encapsulating common operations** in a single reusable block, ideally with a name that clearly describes what the function does.

There is no practical limit to function length, but good design aims at functions that perform **a single well-defined task**. Complex algorithms should be broken down into simpler, easier-to-understand functions whenever possible.

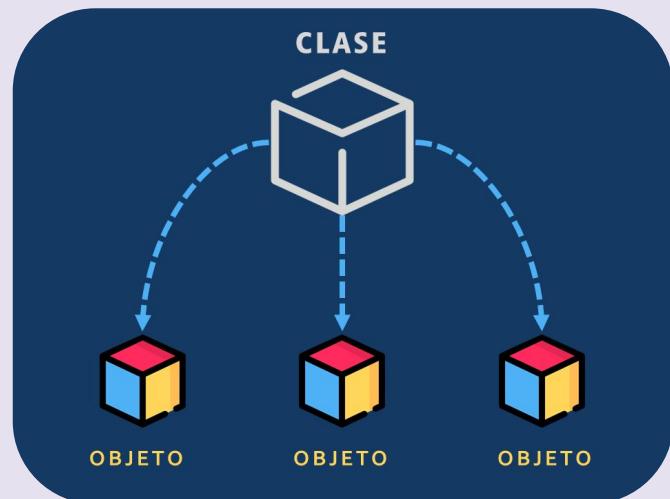
```
function cuadrado(n){  
    return n * n  
}
```

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



# Class

A **class** is an element of object-oriented programming that acts **as a template and will define the characteristics and behaviors of an entity**. The class is going to be like a mold from which we are going to be able to define entities. A class will define the characteristics and behaviors of an entity. If I define, for example, the class person, its characteristics or attributes could be gender, that is, if it is a man or a woman, age and name. The behaviors or methods that we would have in the case of the person class could be breathing, moving, walking, thinking, among others.





# Decorators

A **decorator** is a **software pattern** that is used to **alter the operation of a given piece of code**; either a function, or a class, without the need to use other mechanisms such as inheritance.

Specifically, we refer to **functions** or objects with similar **behavior that allow us to alter how other entities work** without having to explicitly modify their code.

An example of the use of design patterns

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



# Operations sync-async

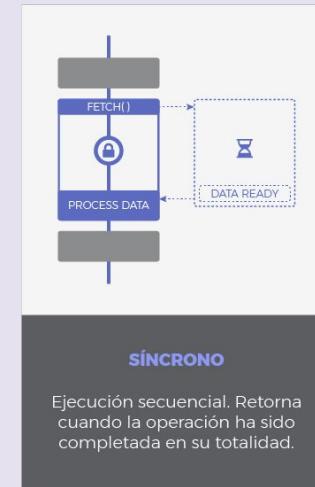
**Synchronous code is code where each instruction waits for the previous one to execute while asynchronous code does not wait for deferred instructions and continues with its execution.** In general, asynchronous code allows for more responsive applications and reduces client waiting time.

## Synchronous

Each instruction will be executed in sequence until completion.

## Asynchronous

In the asynchronous case, some of the instructions will be executed out of time.



# IDE vs Editors

A code editor is a **lightweight program** that does not require much RAM or processor, where you can **open and create one file at a time and save it in a folder**. You can **add plugins to an editor to perform many more functions** (e.g. it can support multiple languages) and make it more powerful.

An **IDE** (**I**ntegrated **D**evelopment **E**nvironment), unlike an editor, is a heavier program that requires much more RAM memory and a more powerful processor, plus it is a **space to work on complete projects, not just files**. They contain **integrated tools**, that is to say, now you will no longer create folders on your own, they can have a **compiler** (for compiled languages), **an emulator, version control and terminals**.

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:

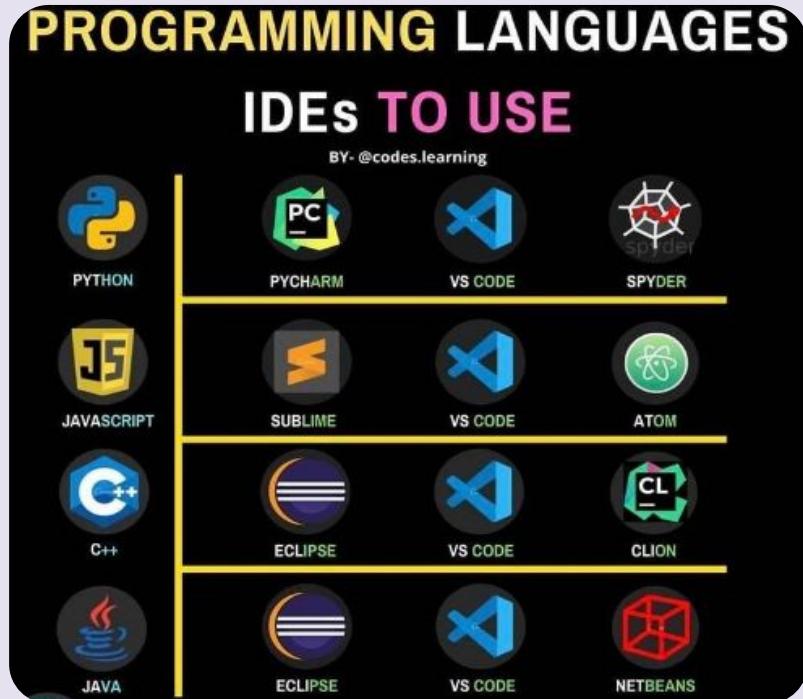
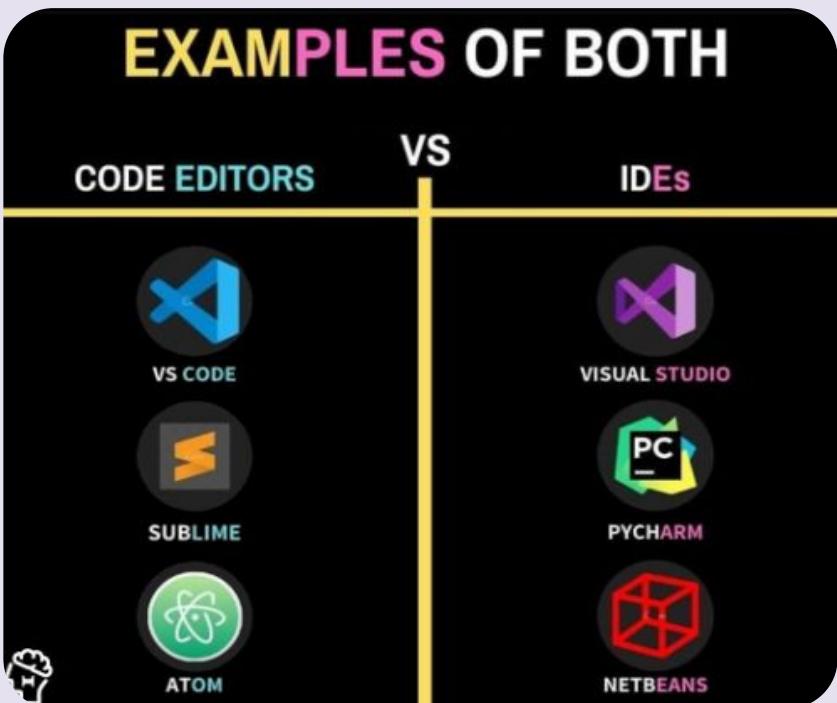
- Software ligero con ayudas para escribir código (resaltado de sintaxis, autocompletado, etc.)** (Editor)
- Integra un editor con las herramientas que necesita un desarrollador (debugger, compilador, etc.)** (IDE)
- Soporta múltiples lenguajes y tecnologías.** (IDE)
- Se especializa en un lenguaje o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

**EJEMPLOS DE EDITORES**: VS, Sublime, Atom.

**EJEMPLOS DE IDES**: VSCode, Android Studio, PyCharm.

At the bottom, it says: "Domina la tecnología con EDteam y #NuncaTeDetengas" and provides the URL "ed.team/cursos".

# IDE vs Editors





# Framework

A **framework** is an **environment or framework, a set of standardized** practices, concepts and criteria to follow. Following some rules, the framework forces us to use **good practices** for our code.

On the other hand, frameworks also **provide us with a set of pre-developed tools**. They are usually **common functions** in all projects. For example, if we have a web project we will surely have users and if we have users the most normal thing is that they have to "login". This functionality of logging in with an email and a password would already be done if we use a framework.



# Scripts vs Programs

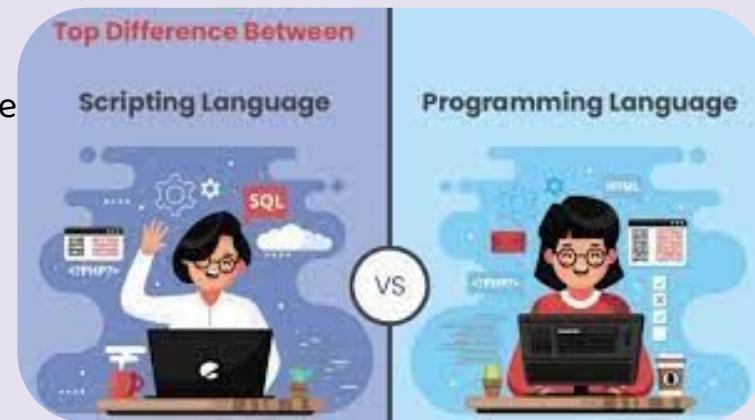
A script is a code file that can execute commands to perform a specific task. It is a single file.

A program, like a script, executes commands but it can have the difference of not being just a single file but a whole solution.

All scripts are programs but not all programs are scripts.

Examples of programs and scripts are the following:

- Script: Code that sends a message every day
- Program: Billing system





# Version control

Version control, also known as “source code control,” is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Version control software keeps track of all code modifications in a special type of database. If a mistake is made, developers can go back in time and compare previous versions of the code to help resolve the error, while minimizing disruption to all team members.

## ¿SABES QUÉ ES GIT?

Es un sistema de **control de versiones**. Lleva un registro de todos los **cambios** y **avances** de tu proyecto.

- Funciona como una máquina de tiempo, puedes ir al pasado de tu código o volver al presente.
- GIT TRABAJA CON RAMAS**  
Ayuda a que **varias personas trabajen** en un mismo proyecto y pueden realizar **modificaciones sin afectar a los demás** archivos. Una vez que estén listos los cambios se **fusinan con la rama principal**.
- Github es un servicio que te ayuda a **almacenar tu proyecto en la nube**, además existen otros servicios como Gitlab o Bitbucket.
- Todo desarrollador sin importar el lenguaje debe dominar Git.

Prof. Beto Quiroga

Aprende a trabajar en equipo con Git en:  
[ed.team/cursos/git-workflow](http://ed.team/cursos/git-workflow)

EDteam



# Version control





# API

## The API

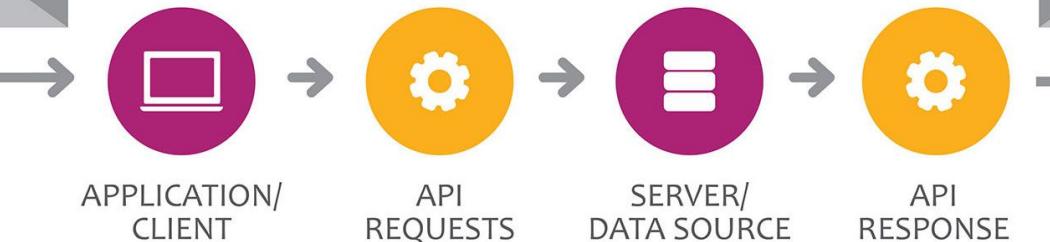
Application Programming Interface



### API Definition

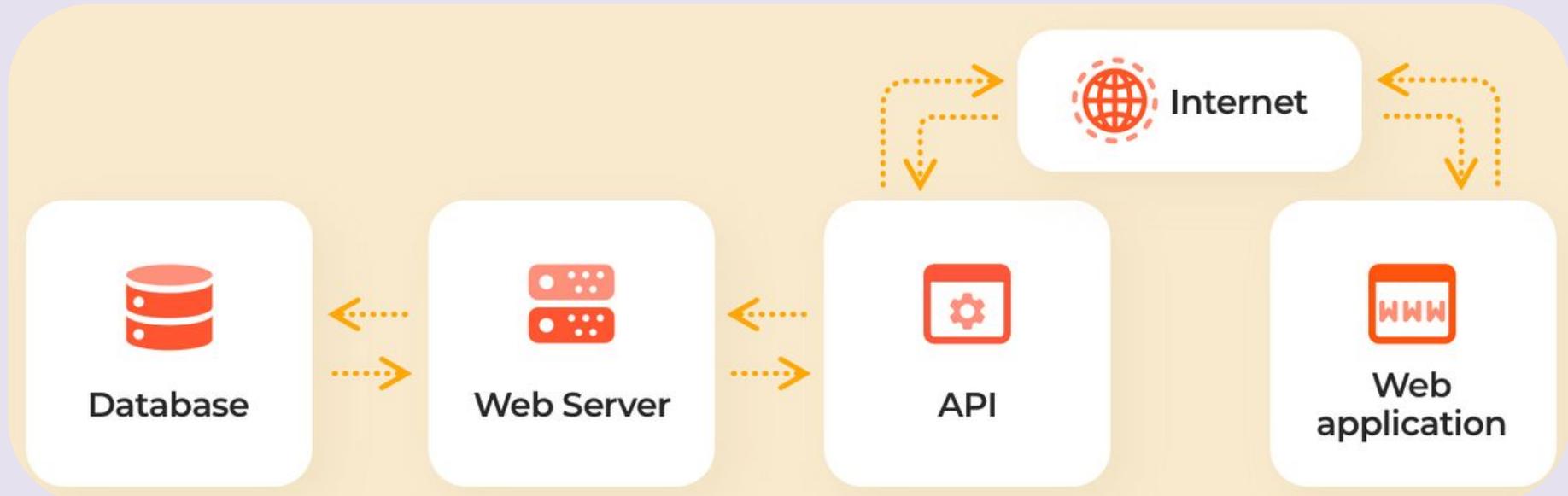
An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data.

### 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 



# Design Patterns

**Design patterns** are a general, reusable solution applicable to various software design problems. They are templates that identify issues in the system and provide appropriate solutions to common problems that developers have faced over a long time, through trial and error.

In essence, **design patterns** help you ensure the validity of your code, as they are solutions that work and have been tested by many developers, making them less prone to errors.

## ¿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](https://ed.team/programacion)





# Microservices

**Microservices** are both an architectural style and a way of programming software. With **microservices**, applications are divided into their smallest, independent components.

Each of these components or processes is a **microservice**. This software development approach values granularity, simplicity, and the ability to share a similar process across multiple applications. It is a fundamental element of optimizing application development towards a **cloud-native model**.

## ¿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](http://ed.team/cursos)





# Containers

**Containers** are, in essence, processes that run within the operating system and have characteristics that make them "special." They share the ecosystem along with "common" processes, and it is thanks to the Linux **KERNEL** that, through some of its features, provides these processes with remarkable capabilities.

A **container** is a process that runs isolated from other containers and even from other "common" processes.

**Containers** are a form of operating system virtualization. A single **container** can be used to run anything from a **microservice** or software process to a larger application.



The infographic is titled "¿QUÉ ES docker?" and features a blue header with the title and a small Docker logo. Below the title is a portrait of Prof. Mauricio Ballesteros Valladares and a brief description of Docker as an open-source project for creating and running applications using containers, mentioning the classic phrase "En mi computadora si funciona". To the right is a Docker star icon.

**¿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**

Una vez que se construye una imagen, **se pueden cargar en un registro como Docker Hub**.

Cuéntanos tu experiencia trabajando con Docker.  
[ed.team/cursos/docker](http://ed.team/cursos/docker)

**EDteam**



# Cloud

The **cloud** is not a **physical entity** but an **enormous network of remote servers worldwide** that are connected to function as a **single ecosystem**. These servers are designed to **store and manage data, run applications, or deliver content or services**, such as **video streaming, webmail, office software, or social media**. Instead of accessing **files and data** from a **personal or local computer**, you access them **online** from any **device connected to the Internet**, meaning **information is available wherever you go and whenever you need it**.



# Operative system and environment variables

Environment variables are variables available to your program/application dynamically at runtime. The value of these variables can come from various sources: text files, third-party secret managers, calling scripts, etc.

What's important here is that the value of these environment variables is not hardcoded into your program. They are truly dynamic and can change depending on the environment in which your program runs.





# Architecture

**Software architecture** consists of **patterns** or **guidelines** that aid in the **construction of a program** (application). These **patterns** serve as a **guide** for **developers, analysts**, and all **related roles** to meet the **application's requirements**.

Defining **technologies** is one of the most **important aspects of software architecture**, but it doesn't mean that if a decision is made, it is something **final** that cannot be **changed in the future**.

One of the objectives of **software architecture** is to create an **application structure** that is **easily scalable** and **not heavily coupled**.

## 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

# Latest

# developments

in technology



# Latest developments



## Virtual Reality

- Education
- Culture
- Events sector
- Customer experience



## 3D Printing

- Prostheses
- Automotive industry
- Construction
- Food
- Space
- Etc



## IoT

- Medicine
- Farms
- Schools
- Space
- Domotics
- Etc

# Latest developments



## Machine learning

Machine learning is a branch of artificial intelligence that allows machines to learn without being specifically programmed to do so.



## Big data

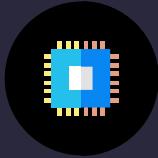
Big Data is a term that describes the sheer volume of data, both structured and unstructured, that floods businesses every day.



## Criptomonedas

They use a complex code to create a single, verifiable currency that can be traded online and used to make purchases. They are decentralized.

# Latest developments



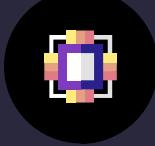
## Quantum computing

It is based on the use of cubits, a special combination of ones and zeros. It is a computing paradigm different from that of classical computing.



## 5G

5G technology offers a theoretical maximum speed of 20 Gbps, while the maximum speed of 4G technology is only 1 Gbps.



## Neural networks

A set of neurons connected to each other and working together, without a specific task for each one.



# Main advances in AI

## DALL-E 2





# Main advances in Ai

## Hugging Face

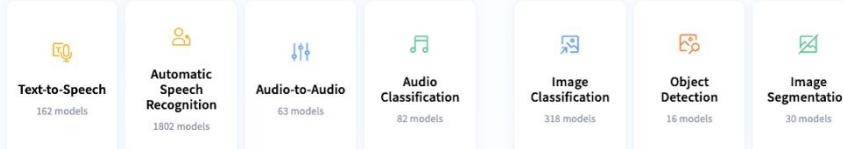
### Tasks

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



# Start of 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

