**Turing, Alan**: Matemático inglés (1912-1954) que estableció las bases de la computación moderna. En el campo de la inteligencia artificial creo la prueba de Turing (1950), un concepto mediante el cual se puede establecer si una maquina exhibe un comportamiento inteligente igual al de un ser humano (no si es inteligente o no, esto sería más difícil de responder).

**Prueba de Turing**: Con esta prueba se evalúa el si una máquina es capaz de exhibir un comportamiento inteligente similar o indistinguible del de un ser humano. En esta prueba, un evaluador humano a través de preguntas realizadas en lenguaje natural a dos evaluados, uno es una máquina y otro es un humano, tiene que ser capaz de distinguir si quien le responde es la máquina o el ser humano. Para ello se basa sólo en las respuestas escritas.

**Inteligencia Artificial**: Se puede definir como un conjunto de algoritmos que tiene como fin crear maquinas que presentan las mismas capacidades que el ser humano. Coloquialmente, el término "inteligencia artificial" se usa a menudo para describir máquinas (o computadoras) que imitan funciones "cognitivas" que los humanos asocian con la mente humana, como "aprender" y "resolver problemas". A medida que las máquinas se vuelven cada vez más capaces, las tareas que se considera que requieren "inteligencia" a menudo se eliminan de la definición de IA, un fenómeno conocido como efecto de IA. Una broma en el teorema de Tesler dice que "la IA es lo que no se ha hecho todavía".

Se pueden distinguir varios tipos de Inteligencia Artificial:

Sistemas que piensan como humanos: Toman decisiones, razonan, aprenden.Redes Neuronales.

Sistemas que actúan como humanos: Robots.

Sistemas que piensan racionalmente: Sistemas que imitan el pensamiento lógico racional de los humanos. Sistemas expertos.

Sistemas que actúan racionalmente: idealmente, son aquellos que tratan de **imitar de manera racional el comportamiento humano,** como los agentes inteligentes

**Programación**: en el ámbito de la informática mediante la programación transformamos en código o lenguaje de programación una serie de instrucciones para qué un determinado sistema realice las tareas que el usuario solicita.

**Lenguaje de programación**: **es un idioma artificial prediseñado formado por signos, palabras y símbolos** que permite la [comunicación](https://concepto.de/comunicacion/) entre el programador y el ordenador.

**Lenguaje máquina:** El primer lenguaje que se utilizó para comunicarse con una máquina, es muy básico y limitado basado en el [sistema binario](https://concepto.de/sistema-binario/) (uso de los números 0 y 1 en distintas combinaciones) que es el lenguaje que los ordenadores reconocen, por lo que aún hoy todo lenguaje es convertido a este. Fue reemplazado, ya que resultaba una forma de programación tediosa y difícil.

**Lenguaje ensamblador:** Es un tipo de lenguaje que surgió después del lenguaje máquina, estos lenguajes hacían uso de códigos de palabras. Se utilizaban palabras simples, mnemotécnicas y abreviaturas que tenían su correlativo y eran traducidas al código máquina. El lenguaje ensamblador resultaba más fácil de recordar y realizar por el usuario que el código máquina.

**Lenguaje de alto nivel:** A finales de la década del 50 surgió el Fortran, un lenguaje de programación desarrollado por IBM que dio inicio a la aparición de lenguajes basados en conjuntos de algoritmos mucho más complejos. Estos lenguajes se adaptaban a distintos ordenadores y eran traducidos por medio de un software al lenguaje de máquina.

**Programación**: en el ámbito de la informática refiere a **la acción de crear programas o aplicaciones** a través del desarrollo de un [código fuente](https://concepto.de/codigo-fuente/), que se basa en el [conjunto](https://concepto.de/que-es-un-conjunto/) de instrucciones que sigue el ordenador para ejecutar un [programa](https://concepto.de/programa-informatico/).

**Programación estructurada**: Busca mejorar y reducir el tiempo del proceso al utilizar subrutinas (subalgoritmos dentro del algoritmo principal que resuelve una tarea).

**Programación modular**. Divide los programas en módulos para trabajar con ellos y resolver los problemas de manera más simple.

**Programación orientada a objetos**. Utiliza objetos (entes con características, estado y comportamiento) como elementos fundamentales para la búsqueda de soluciones.

**Transistor**: Componente electrónico semiconductor de tamaño nanométrico que permite convertir una señal analógica en una señal digital. Los transistores normalmente se incluyen dentro de un circuito integrado, los cuales, a su vez, constituyen los circuitos electrónicos que forman parte de los ordenadores. Los transitores se pueden utilizar para dos tareas, para amplificar la señal eléctrica (función de amplificación) o para dejar pasar o bloquear la corriente (función de interruptor). https://www.areatecnologia.com/TUTORIALES/EL%20TRANSISTOR.htm

**Puertas lógicas**: Una puerta lógica es un **bloque en la construcción de un circuito digital**. La mayoría de las puertas lógicas **tienen dos entradas y una salida y están basadas en álgebra booleana**. En un momento dado, cada terminal está en una de las dos condiciones binarias false (alto) o true (bajo). Falso representa 0, y verdadero representa 1. Dependiendo del tipo de puerta lógica que se utilice y de la combinación de entradas, la salida binaria será diferente. Una puerta lógica puede ser pensada como un interruptor de luz, donde en una posición la salida está apagada-0, y en otra, está encendida-1. Las puertas lógicas son comúnmente usadas en circuitos integrados (IC). https://descubrearduino.com/puertas-logicas/

**Puerta AND:** Con este operador la única manera de que la señal de salida sea 1, es que las dos señales de entrada sean 1. Para el resto de combinaciones, la señal de salida es 0. Actúa de la misma forma que el operador lógico “y”.

**Puerta OR**: Con este operador la única manera de que la señal de salida sea 0, es que las dos señales de entrada sean 0. Actúa de la misma forma que el operador lógico “o”

**Puerta XOR:** También llamada “eXclusive-OR”. Esta puerta funciona como el operador lógico “uno u otro”. Para que la señal de salida sea 1, las dos señales de entrada tienen que ser diferentes.

**Puerta NOT:** Este operador es un inversor lógico, tiene sólo una entrada. Si la señal de entrada es 1, la de salida es 0. Si la señal de entrada es 0, la señal de salida es 1

**Puerta NAND:** Este operador actúa como una puerta NAND seguida de una puerta NOT. Funciona como la operación lógica “y” seguida de negación. La señal será 0 cuando las dos señales de entrada sean 1. En el resto de los casos, la señal de salida es 1.

**Puerta NOR:** Este operador actúa como la puerta OR seguida de la puerta NOR. El único caso en que la salida es 1, es si las dos entradas son 0.

**Puerta NXOR:** Es la negación de la puerta XOR. Para que la señal de salida sea 1, las dos señales de entrada tienen que ser iguales.

**Lógica proposicional**, o a veces lógica de orden cero, es un sistema formal cuyos elementos más simples representan proposiciones, y cuyas constantes lógicas, llamadas conectivas lógicas, representan operaciones sobre proposiciones, capaces de formar otras proposiciones de mayor complejidad.

**Algebra:** es la rama de la matemática que estudia la combinación de elementos de estructuras abstractas acorde a ciertas reglas. Originalmente esos elementos podían ser interpretados como números o cantidades, por lo que el álgebra en cierto modo originalmente fue una generalización y extensión de la aritmética.

**Algebra booleana**: Se denomina así en honor a George Boole (1815-1864), matemático inglés autodidacta, que fue el primero en definirla como parte de un sistema lógico. El álgebra de Boole fue un intento de utilizar las técnicas algebraicas para tratar expresiones de la lógica proposicional.

**Ley de Moore**: Escrita por el ingeniero Gordon Moore en 1965. Este ingeniero y co-fundador de Intel, fue el primero que se dio cuenta que el número de transistores por unidad de superficie se duplicaba cada año. En 1975 corrigió su propia ley, que ahora dice que el número de transistores por unidad de superficie, MTr/mm2 (millones de transistores por milímetro cuadrado) se duplica cada dos años. Es decir, cada dos años el tamaño de los transistores disminuye a la mitad. En la actualidad (2020) los transistores de Intel han alcanzado los 10 nm, con lo que en 1 mm2 caben alrededor de 100 millones de transistores.

**Litografia 10 nm:** Termino relacionado con el tamaño de los transistores que integran un chip. Una litografia de 10 nm, quiere decir que se usan transistores de 10 nm, cuanto más pequeño sea el tamaño del transistor, mas unidades entran en un microchip, mejorando eficiencia, rendimiento y generando menos calor.

**Bit**: Unidad más pequeña de información a la que se traduce cualquier lenguaje de programación para que el ordenador lo pueda entender. Esta información puede ser un 1 o un 0, es decir, encendido o apagado.

**Byte**: Conjunto de 8 bits, que se utiliza para expresar más información, como en el caso de los bits, sólo utiliza 1 y 0, pero dependiendo del lugar que ocupan toman diferente valor. Si pusiéramos estos dígitos en 8 casillas, cada una de ellas es una potencia de 2, la de la derecha sería 2E0, la siguiente 2E1, la siguiente 2E2…, es decir la más a la derecha tendría un valor de 1, la siguiente de 2, la siguiente de 4 y así sucesivamente. Se pueden hacer más operaciones sumando las casillas que estén “encendidas”, si todas estuvieran “encendidas”, la suma daría un total de 255 (2E8-1)

**Circuito electrónico:** Es un conjunto de componentes que permiten transmitir la electricidad y convertirla en luz, sonido, movimiento..etc. Son la base de la tecnología moderna.

**Circuito digital**: En este caso las ondas eléctricas son convertidas en ceros y unos y esta información se guarda en la memoria **RAM “**Random Access Memory**”**, mientras el circuito tenga energia, o en el disco duro, haya o no energía. La unidad mínima de información es un bit, y 8 bits hacen un byte. El procesador puede hacer operaciones con los bytes y convertirlos en otros mediante instrucciones, que no son más que código de programación.

**Periférico**: En informática un periférico es cualquier aparato o dispositivo independiente conectado al procesador central de un ordenador. Nos permite comunicarnos con el ordenador.

**Periférico de entrada**: Dispositivos que proporcionan información de cualquier tipo(pueden ser dígitos, palabras, imágenes..) a la unidad de procesamiento central del ordenador. Los más conocidos serían, el teclado, el ratón, el micrófono, una webcam. El procesador conmuta los datos proporcionados y devuelve al usuario a través de los dispositivos periféricos de salida.

**Periférico de salida**: Dispositivos que devuelven los resultados del procesamiento de los datos al usuario, estos datos pueden ser números, sonidos, imágenes. Algunos ejemplos de dispositivos de salida son los monitores, altavoces, impresoras.

**Periférico de entrada/salida**: Dispositivos tanto para que el usuario envíe información a la CPU, como para recibir la información de ella. Pantallas táctiles o los USB son dos ejemplos.

**CPU**: Central Processing Unit, el cerebro del ordenador. Tradicionalmente, se compone de la **ALU**, “Arithmetic Logical Unit”, los **registros de memoria**, que suministran los datos a la ALU y almacenan los resultados y la **CU** o “Control Unit”, que se encarga de acceder a las instrucciones situadas en la memoria principal, interpretarlas y comunicarse con los registros y la ALU y otros elementos, para que cumplan su función de manera coordinada. **https://en.wikipedia.org/wiki/Central\_processing\_unit#Microprocessors**

**ALU**, “Arithmetic Logical Unit”, circuito digital dentro del procesador que se encarga de realizar operaciones aritméticas y lógicas. Se comunica con los registros que le envían los datos y la CU que le indica qué operación realizar.

**Registros de memoria**, un tipo de memoria muy rápida y de poca capacidad, que suministra los datos a la ALU y almacenan los resultados.

**CU** o “Control Unit”, unidad en el interior de la CPU que dirige el funcionamiento del procesador, controla la ALU, los registros, la memoria y la interfaz I/O. Dirige el flujo de datos entre la CPU y otros componentes.

**Microprocesadores**: Hoy en día las CPUs de la mayoría de ordenadores son microprocesadores, es decir todos los elementos se encuentran integrados en un chip.

**Bus**: Sistema de comunicación entre los distintos componentes del ordenador

**Arquitectura-von-neumann**: El rasgo más importante de este tipo de arquitectura de ordenadores, es que los datos y las instrucciones comparten el mismo bus. Por lo que no se puede acceder a una instrucción y manejar un dato a la vez. Esto afecta a la velocidad de procesamiento y se conoce como “von Neumann bottleneck”. En los ordenadores modernos se ha conseguido mitigar esta limitación mediante el uso de caches (pequeñas memorias de acceso muy rápido) entre la CPU y la memoria central, en concreto, los caches más cercanos a la CPU, están divididos en caches de datos y de instrucciones (split cache) y utilizan diferentes buses. La ventaja es que las instrucciones son tratadas como datos, esto hace que los programas puedan escribir programas. Hace posible los ensambladores, los compiladores y otras herramientas.

**Arquitectura Harvard**: A diferencia de la Arquitectura von-Neumann, en esta disposición los datos y las instrucciones se almacenan en sets de direcciones de memoria diferentes (todos dentro de la memoria RAM pero separados) y no comparten buses. Se pueden escribir o leer de la memoria y a la vez acceder a una instrucción.

**Hardware**: Partes físicas, tangibles de un ordenador: microprocesador o CPU, placa madre, memoria RAM, disco duro, fuente de alimentación, tarjeta gráfica, carcasa, dispositivos periféricos.

1. [Microprocesador](https://techlandia.com/definicion-microprocesador-sobre_99685/) o unidad central de procesamiento (CPU)
2. [Placa o tarjeta madre](https://techlandia.com/funcion-placa-madre-computadora-sobre_168309/)
3. La memoria de acceso aleatorio o [memoria RAM](https://techlandia.com/utiliza-memoria-ram-hechos_444633/)
4. [Dispositivo de almacenamiento](https://techlandia.com/cuales-son-dispositivos-almacenamiento-datos-sobre_349592/) de datos o disco duro
5. Fuente de poder o [fuente de alimentación](https://techlandia.com/cuales-son-distintas-partes-fuente-alimentacion-computadora-hechos_93809/)
6. [Tarjeta gráfica](https://techlandia.com/definicion-tarjeta-grafica-hechos_172986/) u otras placas opcionales (GPU)
7. Carcasa, caja, “case” o “chasis”
8. [Dispositivos periféricos](https://techlandia.com/tipos-dispositivos-entrada-sobre_37283/): Mouse, teclado

**Software**: Partes intangibles de un ordenador. Por ejemplo, el sistema operativo y los distintos programas o ampliaciones.

**Sistema operativo (OS):** Es el software que, se comunica con el *hardware* y gestiona sus recursos, además provee de herramientas al resto de aplicaciones del sistema para que puedan funcionar. Tiene privilegios de ejecución sobre el resto de programas.

**BIOS:** El BIOS fue creado en 1975, y sus siglas significan Basic Input Output System o sistema básico de entrada y salida. Su función principal es la de **iniciar los componentes de *hardware* y lanzar el sistema operativo** de un ordenador cuando lo encendemos

**Firmware**: Es un programa muy básico que controla los circuitos electrónicos de un dispositivo. Se almacena en una memoria ROM especial, en un ordenador está integrado en la placa base. El firmeware es un código con instrucciones para decirle al hardware como tiene que funcionar. También se conoce como soporte lógico inalterable.

**Raspberry pi**: Se trata de un ordenador de bajo coste. El Rasperry pi4 cuenta con un procesador, una memoria RAM de 2, 4, u 8 Gigas, puertos USB, HDMI, conexión Ethernet, wireless o bluetooth . Se le pueden conectar distintos periféricos, teclado, ratón, pantalla. Se le pueden instalar diferentes sistemas operativos, según el uso que se le quiera dar.

**Arduino:** El proyecto arduino nació en 2003 en Italia, de mano de unos estudiantes con idea de crear una placa de electrónica de bajo coste, para promover el acceso y uso de la electrónica y la programación y con la filosofía de “learning by doing”. Arduino es una plataforma de creación electrónica y de código abierto, basada en hardware y software libre, para que cualquier persona o empresa pueda crear su propia placa. https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

**Lenguaje de programación**: Lenguaje especial que un programador utiliza para comunicarse con una máquina. Son una serie de instrucciones que tienen que ser escritas con unos signos, palabras, sintaxis o estructura determinadas y que forman un programa.

**Lenguaje maquina**: Lenguaje de bajo nivel, el único lenguaje que puede entender un dispositivo electrónico, se trata simplemente de una colección de 0 y 1, también se llama lenguaje binario, porque sólo usa dos dígitos.

**Lenguaje ensamblador**: Primer intento de sustitución del lenguaje máquina. Es también un lenguaje de bajo nivel, pero algo más sofisticado que el lenguaje maquina sirve para pasar de un lenguaje de alto nivel a

**Lenguaje de alto nivel**: Lenguaje de programación cercano al lenguaje natural, al que utilizamos los humanos, también se llama código fuente (del que se parte). Para que una máquina lo entienda es necesario convertirlo a lenguaje máquina, para ello se usan traductores, compiladores y ensambladores. El usar uno u otro depende de en que lenguaje esté escrito el código fuente.

**Traductores**: Son programas que convierten las instrucciones escritas en otro lenguaje a lenguaje binario y la ejecutan a medida que se traducen.

**Compilador**: Es un tipo de traductor a través del cual todos las instrucciones de un programa escrito en otro lenguaje se traducen a lenguaje maquina a la vez y después se ejecutan. La entrada es una sentencia en lenguaje de alto nivel y la salida un ejecutable. Se puede almacenar el programa traducido y si se necesita no hace falta traducirlo otra vez, el lenguaje ya está compilado.

**Interprete**: Es como un compilador, solo que la salida es una ejecución. El programa de entrada se reconoce y ejecuta a la vez. No se produce un resultado físico (código máquina) sino lógico (una ejecución).

**Lenguaje-compilado:** Un lenguaje compilado se puede ejecutar por sí solo, no necesita de otra herramienta. Eso sí, solo se puede utilizar en la máquina para la que se ha compilado. Cada lenguaje compilado tiene su versión para las diferentes plataformas.

**Lenguaje-interpretado**: A diferencia de un lenguaje compilado, un lenguaje interpretado necesita de un intérprete para poder ser convertido a lenguaje máquina. Es más flexible porque se puede utilizar la misma versión en distintas plataformas, pero se necesita un interprete diferente para cada una de ellas.<https://ingenieriadesoftware.es/diferencia-compilador-interprete-transpilador/> <https://ocw.uma.es/course/view.php?id=61>

**Frontend**: En el entorno del desarrollo web, frontend se refiere a la parte de una aplicación o dispositivo a la que un usuario puede acceder. Los desarrolladore frontend trabajan para que la interfaz del usuario sea fácil de usar y atractiva. Los lenguajes más usados en frontend: HTML, CSS y JavaScript.

**HTML**: “HyperText Markup Language” (lenguaje de marcas de hipertexto) lenguaje de marcado para la elaboración de páginas web. En realidad no es un lenguaje de programación. HTML sólo sirve para indicar como va ordenado el contenido de una página web. Esto lo hace por medio de las marcas de hipertexto las cuales son etiquetas conocidas en inglés como tags. https://codigofacilito.com/articulos/que-es-html

**CSS**: Este lenguaje sirve para definir el diseño de una página web. La filosofía de CSS se basa en intentar separar lo que es la estructura del documento HTML de su presentación. Con CSS puedes crear animaciones, efectos, manejar colores, fuentes, tamaños y mucho más. En realidad no es un lenguaje de programación.

**JavaScript**: Es un lenguaje de programación que sirve para crear páginas webs dinámicas, es decir introducir animaciones, textos que aparecen y desaparecen... Es un lenguaje interpretado. Los programas escritos en JavaScript se pueden probar directamente en cualquier navegador, sin necesidad de intermediarios. No tiene nada que ver con Java.

**Backend:** Se refiere a la parte de una aplicación o dispositivo a la que el usuario no puede acceder, lo que no se ve. Además de datos que son inaccesibles al usuario, esta capa contiene la lógica que maneja dichos datos. También se encarga de enlazar con el servidor. Lenguajes relacionados: PHP, python, java, C++, lisp, R, Prolog.

**Python** Es un lenguaje muy flexible y sencillo. Es de los más usados en IA, pero también sirve para aplicaciones móviles, web, data science. Es uno de los lenguajes más utilizados para IA. Su simplicidad y las filosofías DRY (Don’t Repeat Yourself) y RAD (Rapid Application Development) en las que se basa lo convierten en un candidato ideal. Puede utilizarse tanto para estructurar datos como para generar algoritmos de IA. Además, dispone de un catálogo de librerías muy extenso que permite hacer realidad cualquier tipo de proyecto. Ningún soporte se resiste a Python, puesto que sirve para trabajar en aplicaciones de todo tipo, sean mobile, web, data science o IA.

**R**: Este lenguaje dispone de paquetes de programación muy numerosos. Algunos de ellos se utilizan en el ámbito del Machine Learning, como RODBC. Para garantizar la funcionalidad de la IA, implementan algoritmos de aprendizaje automático. Se trata de uno de los mejores lenguajes para analizar y tratar con datos. Por ello, es posible crear buenas IA con finalidades estadísticas.

**Lisp:** Desde su nacimineto en 1958 de la mano de John McCarthy, Lisp no ha dejado de crecer. De hecho, su creador trabajó en el MTI junto a Marvin Minsky, uno de los padres de la Inteligencia Artificial. Lisp trabaja con expresiones simbólicas y prototipado, herramientas útiles en el campo del Machine Learning. Además, se utiliza en proyectos como CYC, cuyo objetivo es permitir a las aplicaciones basadas en IA ejecutar razonamientos similares a los humanos.

**Prolog:** Suele utilizarse en proyectos médicos y para diseñar sistemas de IA especializados. Se trata de un lenguaje de referencia en el entorno de la ingeniería. Una de sus funcionalidades más destacadas es que permite automatizar el backtracking, que consiste en buscar errores y retroceder hasta el punto anterior para tomar otra alternativa. Para conseguir esto, se basa en estructuras de datos arbóreas que facilitan la búsqueda de patrones. Todas estas características combinadas convierten este lenguaje en uno de los más flexibles.

**Java:** es un lenguaje de programación orientado a objetos que posee todas las herramientas necesarias para trabajar en proyectos de Inteligencia Artificial. Las características más destacadas de Java son la transparencia, la mantenibilidad y la portabilidad. Permite codificar algoritmos muy fácilmente y es un lenguaje escalable. Teniendo en cuenta que una IA está basada en gran medida en estos algoritmos, Java es una muy buena opción. Además, dispone de interfaces de datos muy atractivas para mejorar la experiencia del usuario. Si tenemos alguna duda, dispone de una nutrida comunidad de usuarios que pueden ayudarnos.

**C++**: se dice que es el lenguaje de programación más rápido del mundo. En este sentido, se trata de uno de los mejores lenguajes de programación para IA, puesto que esta tecnología requiere de ejecuciones y tiempos de respuesta rápidos. C++ convierte el código creado por el usuario en datos legibles para las máquinas, cosa que lo convierte en una opción muy eficiente.

**PHP**: Un lenguaje backend del que se dice que es muy fácil de aprender, tiene programación HTML integrada, se utiliza para la creación de páginas web dinámicas. Es de código abierto y puede ser usado con cualquier sistema operativo y con cualquier servidor.

**SQL:** Por sus siglas en ingles “**Structured Query Language”**. Es un lenguaje que surgió para ser usado con las bases de datos relacionales. Sirve para interaccionar con los gestores de bases de datos. Para consultar datos, extraer datos, modificar datos, escribir o eliminar datos.

**Repositorio de control de versiones**: Es una herramienta fundamental para los programadores. Sirven para guardar un registro de todos los cambios realizados en un proyecto.

**Git:** Es un software libre creado por Linus Torvalds, es muy útil para seguir los cambios de un proyecto. Cuando queremos guardar una versión del proyecto, git puede hacer una instantánea del proyecto en ese momento y le asigna un código. Podemos seguir haciendo cambios, y cuando queremos volver a guardar el proyecto con esos cambios, decirle a git que haga otra instantánea, si en un momento dado por cualquier causa, queremos volver a una versión concreta anterior, git puede restaurarla. <https://www.youtube.com/watch?v=ANF1X42_ae4> https://git-scm.com/docs

**Branch**: Cuando hay varias personas trabajando en el mismo proyecto y que se van a encargar de partes diferentes del mismo, se pueden sacar “ramas” o flujos de trabajo del proyecto principal y cada programador trabaja en su rama, una vez finalizan, git se encarga de fusionar las ramas. Git también es capaz de detectar conflictos, es decir, si el trabajo de dos personas se solapa, nos advierte sobre el conflicto a la hora de fusionar las ramas.

**Github**: Github es por un lado un gestor de proyectos y un repositorio online de control de versiones y por otro lado una plataforma social de trabajos en grupo. Para depositar o descargar proyectos a este o desde este repositorio, se utiliza git. Github es una apliaciónde Microsoft.

**Repositorio de control de versiones**: Es una herramienta fundamental para los programadores. Sirven para guardar un registro de todos los cambios realizados en un proyecto.

**Git:** Es un software libre creado por Linus Torvalds, es muy útil para seguir los cambios de un proyecto. Cuando queremos guardar una versión del proyecto, git puede hacer una instantánea del proyecto en ese momento y le asigna un código. Podemos seguir haciendo cambios, y cuando queremos volver a guardar el proyecto con esos cambios, decirle a git que haga otra instantánea, si en un momento dado por cualquier causa, queremos volver a una versión concreta anterior, git puede restaurarla. <https://www.youtube.com/watch?v=ANF1X42_ae4> https://git-scm.com/docs

**Branch**: Cuando hay varias personas trabajando en el mismo proyecto y que se van a encargar de partes diferentes del mismo, se pueden sacar “ramas” o flujos de trabajo del proyecto principal y cada programador trabaja en su rama, una vez finalizan, git se encarga de fusionar las ramas. Git también es capaz de detectar conflictos, es decir, si el trabajo de dos personas se solapa, nos advierte sobre el conflicto a la hora de fusionar las ramas.

**Github**: Github es por un lado un gestor de proyectos y un repositorio online de control de versiones y por otro lado una plataforma social de trabajos en grupo. Para depositar o descargar proyectos a este o desde este repositorio, se utiliza git. Github es una apliaciónde Microsoft.

**Algoritmo**: Conjunto de instrucciones bien definidas y estructuradas para realizar una tarea determinada. Un algoritmo debe ser preciso, debe estar perfectamente definido, debe ser finito, debe ser legible y debe constar de tres pasos: entrada, proceso y salida.

**Diagrama de flujo:** Es una representación gráfica de los pasos de un algoritmo**.** En él podemos representar mediante símbolos los pasos del algoritmo y mediante flechas marcar el flujo entre estos pasos, desde el inicio hasta el final.

**Ada Lovelace**: Fue una pionera de la informática (Londres, 1815-1852), publicó el primer algoritmo informático. Creó este algoritmo para la máquina creada por el **matemático inglés Charles Babbage**, esta “máquina analítica” era capaz de calcular secuencias de números. Este código nunca fue probado, porque la maquina nunca se construyó. Introdujo también los conceptos de **bucle** (grupo de instrucciones que se ejecutan varias veces) o subrutina (parte de programa que puede ser llamado en cualquier momento).

**Algoritmos de cifrado simétrico:** la criptografía simétrica también conocida como Criptografía de una sola clave o Criptografía de Clave secreta se basa en que el emisor del mensaje y el receptor del mensaje comparten una clave secreta. El mensaje se cifra y se descifra con la misma clave. Por tanto, la seguridad de este tipo de cifrado recae únicamente en la clave y no en el algoritmo de cifrado y descifrado. Es importante que la clave sea muy difícil de adivinar.

**Algoritmo de cifrado asimétrico:** En este caso existen dos claves diferentes una clave pública que puede conocer cualquiera, con la que se cifra el mensaje y una clave privada, para descifrarlo, que sólo conoce el receptor del mensaje. Por tanto, no hay envío de claves y en este sentido, es más seguro que el cifrado simétrico. En contrapartida, consume más recursos y es más lento.

**Algoritmo de cifrado híbrido**: Este tipo de cifrado, es una mezcla de los anteriores, en un intento por aunar las ventajas de cada uno, pretende ser seguro a la vez que no tan lento. Se generan para el receptor la clave pública y la privada. Se cifra el mensaje, el receptor manda su clave pública con la que se cifra la clave que se ha usado para cifrar el mensaje. Se envía el archivo y la clave del menaje cifrada.

**Convertidor Analógico-Digital**: Es un dispositivo electrónico que a través de una serie de pasos convierte una señal analógica en una señal digital. Las señales digitales tienen ciertas ventajas frente a las analógicas. Son reproducibles (siempre dan el mismo resultado), fáciles de diseñar, flexibles, programables, más veloces. Para convertir una señal de analógica a digital, los pasos que se siguen son: Muestreo de la señal analógica, cuantificación y conversión a señal digital.

**Resolución de un conversor A/D**: Este parámetro se refiere a la cantidad de valores analógicos discretos que puede convertir el conversor digital. Por ejemplo, si el conversor puede convertir 256 valores analógicos, para codificarlos en digital se necesita una resolución de 8 bits (2 elevado a la 8.

**Tasa de muestreo**: Se refiere a la frecuencia con la que se recogen muestras de una señal analógica continua para convertirla en digital. Según el teorema de muestreo de Nyquist-Shannon, para poder digitalizar una señal analógica y transmitirla recuperarla con la máxima fidelidad posible, se requiere que la señal analógica sea muestreada al menos dos veces su frecuencia máxima. Es un error creer que una misma señal muestreada con una tasa elevada se reconstruye mejor que una muestreada con una tasa inferior.

**Transistor Bipolar**: Un transistor bipolar es un elemento tanto de los circuitos analógicos como de los circuitos digitales. Este tipo de transistores funcionan de dos maneras, como amplificador en el caso de circuitos analógicos o como interruptor (on/off) en circuitos digitales. Esta construido con un material semiconductor y consta de tres partes, la base, el emisor y el colector. Puede encontrarse en tres estados: activo, o saturado

**Arquitectura cliente-servidor**: La web está basada en esta arquitectura. Tanto el cliente (front-end) como el servidor (back-end) son dos máquinas conectadas. El cliente pide información al servidor y este se la “sirve”, se la proporciona, siempre y cuando el cliente tenga los permisos para obtener esta información. El servidor antes de dar la información validará los permisos. Aunque en los inicios de la web se usaba PHP tanto para front-end como para back-end, a medida que el desarrollo web avanza aparecen lenguajes específicos para front-end y back-end.

**Cliente**: Se comunica con el usuario y realiza las peticiones al servidor.

**Servidor**: Tiene acceso a ficheros compartidos, acceso a las bases de datos. Existen servidores de archivos, de bases de datos, servidores web, servidores de aplicaciones….

**API:** Del inglés “Application programming interface”, aplicación que se utiliza para conectar el front-end con el back-end, hacen de intermediario, no es necesario que se utilice en mismo lenguaje en los dos extremos para comunicarse. Facilitan mucho el desarrollo web. Un solo front-end puede comunicarse con muchos back-end a través de APIs. Los programadores back-end son los que crean las APIs.

**HTTP**: Protocolo de transferencia de hipertexto ( “Hypertext Transfer Protocol” en inglés), es un protocolo para la transferencia de información HTML entre un cliente y un servidor web, se usa en cada transacción www (“world wide web”) para atender a las peticiones que hace el cliente. El navegador (cliente) realiza la petición al servidor web y este le devuelve la página (HTML) tal cual, sin interpretar, esta interpretación la hace el navegador.

**Servidor web**: Este tipo de servidores se encargan implementar el protocolo HTTP para enviar la información solicitada por el cliente.

**URL**: “Uniform resource locator” dirección en internet de un recurso (documentos, imágenes..), está dirección se da en un formato preestablecido: protocolo://máquina/directorio/fichero/ protocolo://usuario:contraseña@máquina:puerto/directorio/fichero

**Dirección IP**: Número que identifica a un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo **IP** (Internet Protocol). Se representa mediante un número binario de 32 bits (IPv4).

**API rest**: Son las APIs más usadas. Funcionan a través de endpoints, cada vez que el cliente solicita un tipo de datos tiene que usar un endpoint diferente (una URL diferente). Cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.

**Programador front-end**: Se ocupa de la parte del cliente, se conecta a través de las APIs con el back-end. No tiene porque saber de diseño de paginas web, eso es función del diseñador

**Programador back-end**: Se encarga de la parte del servidor. Tiene que saber crear APIs, además se conecta con el servidor, bien a través usando SQL o bien a través de OMR. No tiene porqué encargarse de las bases de datos ni de los servidores. Si no de preparar los datos para que los consuma el front-end.

**Programador full-stack**: Stack un conjunto de tecnologías que permiten desarrollar una aplicación de principio a fin, por tanto un programador full-stack sabe de front-end y back-end. Pero los stacks son muy diferentes para cada aplicación o en cada empresa, por tanto un programador full-stack sabe programar para un stack específico, pero no para todos los stacks, esto sería casi imposible.

**Bases de datos relacionales**: Por mucho tiempo todas las bases de datos eran relacionales. En este tipo de bases de datos existen vínculos entre los distintos tipos de datos. Los datos están estructurados de una determinada manera. Cuando se consultan, la base de datos montan la estructura, establece los vínculos y devuelve los datos. La más famosa MySQL.

**Protocolo TCP/IP**: Este modelo permite un intercambio de datos dentro de una red, definiendo los pasos a seguir desde que se envían los datos hasta que son recibidos. Para lograrlo utiliza un sistema de capas con jerarquías (se construye una capa a continuación de la anterior) que se comunican únicamente con su capa superior (a la que envía resultados) y su capa inferior (a la que solicita servicio). Este protocolo está basado en otro similar Protocolo OSI, aunque simplificado. El protocolo OSI tiene 7 capas, en el TCP/IP, se han fusionado algunas de estas capas y puede tener 4 o 5 capas. En el modelo TCP/IP de 4 capas, los niveles de enlace de datos y el nivel físico están fusionados.

Protocolo TCP/IP **Nivel físico**: Los protocolos de esta capa tienen que especificar cómo se van a transmitir los datos físicamente (forma de los conectores, que tipo de señales se transmiten y todas las características de estas señales, es la primera capa del modelo y ofrece la posibilidad de acceso **físico** a la red (que bien puede ser en anillo, ethernet, etc.), especificando el modo en que los datos deben enrutarse independientemente del tipo de red utilizado. La unidad de información en esta capa son los bits.

Protocolo TCP/IP **Nivel de enlace de datos**: El objetivo de la capa de enlace es conseguir que la información fluya, libre de errores, entre dos máquinas que estén conectadas directamente (servicio orientado a la conexión). Para lograr este objetivo tiene que montar bloques de información (llamados **tramas** en esta capa), dotarles de una dirección de capa de enlace (Dirección **MAC** o **dirección física**), gestionar la detección o corrección de errores, y ocuparse del “control de flujo” entre equipos (para evitar que un equipo más rápido desborde a uno más lento).

Protocolo TCP/IP **Nivel de red o Internet**: es la primera capa que tiene conciencia de la topología de la red, elige el mejor camino para que los datos lleguen a su destino, proporciona el paquete de datos o datagramas y administra las **direcciones IP (direccionamiento lógico)**. Esta capa es considerada la más importante, es el “pegamento que mantiene unida internet”, es capaz de funcionar sobre multitud de protocolos de la capa de enlace o la capa de transporte. Los protocolos más importantes de esta capa son el protocolo IPv4 e IPv6. Los **datagramas** son paquetes de datos que constituyen los bloques de información en este nivel.

Protocolo TCP/IP **Nivel de Transporte**: es la primera capa de extremo a extremo, se asegura de que llegan todos los paquetes. Esta capa permite conocer el estado de la transmisión, así como los datos de enrutamiento y utilizan los **puertos** para asociar un tipo de aplicación con un tipo de dato. En este nivel si el tamaño de los datos es muy grande se dividen en **segmentos** y estos segmentos se envían ordenadamente.

Protocolo TCP/IP **Nivel de Aplicación**: Es donde están los programas con los que nosotros interactuamos, hay un número muy elevado de protocolos. Es la parte superior del protocolo TCP/IP y suministra las aplicaciones de red tipo Telnet, FTP o SMTP, que se comunican con las capas anteriores (con protocolos TCP o UDP). Los datos se preparan para ser enviados.

**IP**: La dirección IP es un número que los proveedores de internet asignan y que identifica de manera única a cada dispositivo conectado a una red que utilice el protocolo IP, que corresponde al nivel de red del modelo TCP/IP. Se trata de cuatro bloques de números de tres cifras (desde 000 a 255), separados por un punto, por ejemplo 132.245.128.120. En realidad son cuatro bytes convertidos al modelo numérico decimal.

**MAC**: la dirección MAC (siglas en inglés de **Message Authentication Code** or **Media Access Control**) es un identificador de 48 bits (6 bloques de dos caracteres hexadecimales [8 bits]) que corresponde de forma única a una tarjeta o dispositivo de red. Se la conoce también como **dirección física**, y es única para cada dispositivo, tiene gran importancia en la red local, pero no es posible usarlo como dirección a larga distancia.

**Sistema de Nombres de Dominio (DNS):** esto permite traducir a palabras la dirección IP. Es mucho más sencillo para nosotros recordar una palabra que una serie de números. Lo mismo ocurre con las direcciones de correo electrónico, es mucho más fácil recordar email@somedomain.com que email@192.168.1.1

**FTP**: “**File transfer protocol**” es un protocolo de transferencia de archivos entre un ordenador que actúa como servidor y otro que actúa como cliente. Existen programas que ayudan a realizar conexiones FTP, el más conocido es Filezilla.

**SMTP**, **POP3** e **IMAP:** Podemos enviar correos electrónicos gracias a estos protocolos TCP/IP que actúan al nivel de aplicación.

**SMTP**: “**Simple mail transfer protocol”**. A través de este protocolo se envían correos electrónicos a un servidor de correo. También se usa para enviar mensajes desde un servidor de correo a otro, en caso de que el remitente y el receptor tengan diferentes proveedores del servicio de correo electrónico. Normalmente, usa el **puerto 25** como estándar. También puede usar el puerto 587 y el 465.

**POP3**: versión 3 del “**Post Office Protocol” se usa para recuperar mensajes de correo desde un servidor de correo a un cliente de correo.**

1. **Conecta el servidor de correo en el puerto 110 (995 para conexiones SSL/TLS).**
2. **Recupera el correo electrónico.**
3. **Se deshace de las copias de los mensajes almacenados en el servidor.**
4. **Se desconecta del servidor en cuestión.**

**IMAP**: “**Internet Message Access Protocol”**, en su versión actual IMAP4, es el protocolo más sofisticado. Les permite a los usuarios agrupar los mensajes relacionados y guardarlos en folders, los cuales luego pueden ser organizados jerárquicamente. También tiene marcadores en los mensajes que nos indican cuando un correo ha sido leído, borrado o reenviado Deja copias de los mensajes en el servidor.

1. Se conecta al servidor de correo en el puerto **143** (o **993** en el caso de conexiones SSL/TLS).
2. Recupera el mensaje de correo electrónico.
3. Se mantiene conectado hasta que la aplicación del cliente de correo cierre y descargue los mensajes deseados

**VPN**: **“Virtual Private Network”** es una tecnología de red de ordenadores que permite una extensión segura de la red de área local (LAN) sobre una red pública o no controlada como Internet. Permite que el ordenador en la red envíe y reciba datos sobre redes compartidas o públicas como si fuera una red privada.

**ELT: Del inglés “Extract, Transform and Load”, se trata de procesos a través de los cuales se extraen datos de diversas fuentes, estos datos se transforman para puedan ser útiles y se vuelcan en el almacén de destino.**

**Extracción de datos: En esta parte del proceso se extraen los datos, los normal es que se recojan datos de diferentes tipos de fuentes, se pueden extraer los datos de una sola vez o hacerlo en pequeños lotes, por ejemplo, que cada día se extraigan sólo los datos que se generaron el día anterior. Hay que tener en cuenta varias cosas a la hora de extraer los datos, como que los datos tengan la estructura que se esperaba, pero lo más importante es que el proceso de extracción tenga el mínimo impacto posible en el sistema de origen, que no lo ralentice o haga que colapse.**

**Transformación de datos: Antes de cargar los datos en el almacén de destino, hay que depurarlos y limpiarlos, esto puede consistir en darles homogeneidad, eliminar duplicados, eliminar columnas, transformar algunos datos etcétera.**

**Carga de datos: Una vez lo datos están preparados hay que cargarlos en el sistema de destino, existen diferentes opciones para cargar los datos: se pueden acumular datos durante un periodo de tiempo, calcular el valor promedio o el sumatorio y cargar sólo esos resultados, o por el contrario, conservar cierta granularidad de los datos (conservar información a distintos niveles (*rolling*)), por otro lado según la base de datos, se pueden cargar en “bulk”, se pueden insertar registro a registro.**

**Bases de datos: Las bases de datos nacen de la necesidad de conservar información, de que no desaparezca y de que esté accesible cuando sea necesario consultarla**

**DBMS**: por sus siglas en inglés Database Management Systems, se trata de sistemas de gestión que permiten el almacenamiento y el acceso rápido a la información de las bases de datos, actualmente son digitales y automatizados.

***Data Mining*: o minería de datos, es un conjunto de técnicas y tecnologías** que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos que expliquen el comportamiento de estos datos. Estos patrones se pueden encontrar usando algoritmos de búsqueda o técnicas estadísticas próximos a la AI.

**Data pre-processing**: Se trata del tratamiento de los datos previamente a que puedan ser usados para su análisis, algunos de estos procesos consisten en eliminar el “ruido”, las inconsistencias o las redundancias.

**Bucle**: en programación un bucle es una parte de un programa que se repite hasta que la condición asociada deja de cumplirse, deja de ser verdad. Los bucles más comunes son el bucle **while** y el bucle **for**. En el caso del bucle **for** podemos marcar el número de veces que queremos que el bucle se repita.

**API**: en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s), *Application programming Interface*, este término se refiere a ciertas soluciones que permiten la comunicación entre diferentes aplicaciones o software y compartan datos. Gracias a las APIs nuestros aplicaciones pueden utilizar soluciones creadas por otros, las APIs nos permiten la abstracción en programación, usamos herramientas, dando importancia a lo que hacen, no a como lo hacen, no necesitamos saberlo para usarlas.

**API-rest**: Es un tipo de API, rest se refiere a un tipo de arquitectura de software, es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

**Protocolo de comunicación**: es un conjunto de reglas que permiten el intercambio de datos entre dos o más entidades. Es decir, un conjunto de normas que están obligadas a cumplir todas las máquinas y programas que intervienen en una transferencia de datos entre ordenadores sin las cuales la comunicación resultaría caótica y por tanto imposible.

**XML**: No es un lenguaje en sí, es parte de SGML(*Estándar Generalised Mark-up Language*), pero simplificado y adaptado a internet. Sirve para especificar el diseño de lenguajes de marcado, HTML, sería en realidad una aplicación de XML. Tiene como ventajas que es fácilmente procesable, separa claramente el contenido y el formato, está diseñado para cualquier lenguaje y alfabeto. https://www.mundolinux.info/que-es-xml.htm

**JSON**: son las siglas de *JavaScript Object Notation*, es un formato ligero para almacenar y transportar datos, se usa a menudo cuando los datos se envían desde un servidor a una página web, es "autodescriptivo" y fácil de entender.

**Bases de datos**: Almacenes de datos, donde la información se guarda de manera segura y sistemática, lo que permite un acceso rápido a los datos, básicamente es un conjunto de **tablas**, similares a las tablas de una hoja de cálculo, formadas por filas (**registros**) y columnas (**campos**). Los registros o **entidades** representan cada uno de los objetos descritos en la tabla y los campos los atributos (variables de cualquier tipo) de los objetos.

Bases de datos relacionales o **SQL**: En este tipo de bases de datos son las más antiguas y las más usadas, en ellas las tablas comparten alguno de los campos, por tanto, las tablas se pueden relacionar a través de este campo. Se supone que son más lentas que las no relacionales, pero más fiables.

Bases de datos no relacionales: este tipo de base de datos es mucho más flexible que la bases de datos SQL, la información se consigue de manera mucho más rápida, la información relacionada se puede guardar en un solo documento. Es ideal para bases de datos gigantes (amazon, facebook), son muy rápidas, aunque puede haber información redundante.

Gestor de base de datos (**SGBD**): *DataBase Managenent System* (**DBMS**) Diferente de la propia base de datos, se trata del software que se utiliza para trabajar con las bases de datos, para crearlas, manipularlas, consultarlas o borrarlas, por ejemplo. Los principales Sistemas gestores de bases de datos relacionales (**SGBD SQL**) actualmente son MySQL, MariaDB, SQLite, PostgreSQL, Microsoft SQL Sever, ORACLE. Entre los SGBD de bases de datos no relaciones estarían MongoDB, Cassandra, Redis.

**Red LAN**: “Local Area Network”, es una red de área local. Es una red en un espacio reducido como una casa, o como mucho un edificio.

**Red MAN**: “Metropolitan Area Network”, un red más amplia que la local, pero no llega a ser internacional. Un ejemplo sería una red de un campus universitario.

**Red WAN**: “Wide Area Network”, una red que cubre un área geográfica extensa, pueden ser nacionales o internacionales, un ejemplo puede ser internet.

Topología de **Red en bus**: En la que un mismo cable (o backbone) conecta a las computadoras y permite la transmisión de datos en línea recta, haciéndola sencilla pero susceptible a daños del cable o a la interrupción del tráfico.

Topología de **Red en estrella**: En la que todos los computadores se conectan a un servidor central que administra los recursos de la red y los asigna según se le solicite.

Topología de **Red en anillo**. En la que todos los computadores están conectados con sus vecinos mediante una transmisión unidireccional, que interrumpe la red de haber un fallo en algún nivel de la misma.

Topología de **Red mixta**. Combina dos o más modelos de los anteriores.

**Máscara de (sub)red**: Debido a que actualmente no hay tantas IPs diferentes como máquinas, se ha implementado un sistema de subredes. En las distintas subredes puede haber direcciones IP que coinciden. La máscara de subred (que también son 32 bits), permite saber si dos máquinas están en la misma subred o no. Cuando se suman con el operador lógico AND la dirección IP y la máscara de subred, si los ordenadores están en la misma red la **dirección de red** (el resultado de la operación) es el mismo. Cuando se envía un paquete entre máquinas de la misma subred se puede hacer directamente. Si no, hay que usar un router o un Gateway.

**Tipos de redes**: Según cuantos de los dígitos de la dirección IP sean asignados a identificar la red o el host las redes se dividen en tres tipos principales, A, sólo el primer octeto de bits se dedica a identificar la red, B, se dedican dos octetos a identificar la red y dos a identificar al host, C: se dedican 3 octetos a identificar a la red y sólo 1 octeto de bits a identificar al host. Existen también las redes de tipo D (multicast) o E (sólo experimentales), pero son algo diferentes a las anteriores.

**IP pública**: Es la dirección IP que tiene el router cuando se conecta a internet.

**IP privada**: Es la dirección individual de cada ordenador con la que se le identifica en la red privada a la que pertenece.

**Hub**: Dispositivo para conectar ordenadores en una red local. Cuando recibe información de alguno de los ordenadores, la replica y la envía al resto de ordenadores en la red local, a todos ellos, ya que no es capaz de identificar cual es el ordenador destino.

**Switch**: Al igual que el hub es un dispositivo para conectar ordenadores en una red local, pero en este caso es capaz de identificar a que ordenador debe enviar la información, lo que permite que el resto quede libre para poder comunicarse entre ellos.

**Router**: A través de este dispositivo se pueden conectar dispositivos en una red local o se puede acceder a otra red, por ejemplo a Internet, para ello se usará el puerto de acceso.

**Gateway**: (puerta de enlace) es un dispositivo, con frecuencia un ordenador, que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Normalmente se utiliza para dotar a los dispositivos de una red LAN de un acceso a la red exterior. Traduce las direcciones IP de los dispositivos de la red local (NAT: Network Address Translation) y puede aplicar una técnica llamada enmascaramiento de IP para convertir las direcciones IP privadas de los dispositivos de la red LAN en una única dirección IP pública que se utilizará para conectarse a internet. La diferencia fundamental entre routers y gateways radica en que los gateways se usan para gestionar el tráfico entre redes diferentes en cuanto a protocolos y arquitecturas, a diferencia de los routers que gestionan el tráfico entre redes similares.

**Servidor DNS**: (“Domain Name System”) Sistema encargado de asignar uno o varios nombres a una dirección IP y también de realizar la asignación inversa. El servidor encargado de prestar este servicio se llama Nameserver. Estos nombres están estructurados con una jerarquía, los componentes de la estructura se separan por puntos. El dominio del primer nivel “Top Level Domain” es asignado por los servidores llamados “Root Name Server”.

**Diagrama de flujo:** Es una representación gráfica de los pasos de un algoritmo**.** En él podemos representar mediante símbolos los pasos del algoritmo y mediante flechas marcar el flujo entre estos pasos, desde el inicio hasta el final.

**Algoritmo**: Conjunto de instrucciones bien definidas y estructuradas para realizar una tarea determinada. Un algoritmo debe ser preciso, debe estar perfectamente definido, debe ser finito, debe ser legible y debe constar de tres pasos: entrada, proceso y salida.

**Convención de nombres en programación:** Se trata de usar una serie de reglas a la hora de nombrar las diferentes identidades de un lenguaje de programación, como son clases, variables o funciones. Usar estas reglas haría más sencillo el que el código fuese más comprensible para otros programadores, permitiría centrar la atención en revisar el código y no en revisar el estilo de los nombres o la sintaxis. Es un tema controvertido, ya que el elegir una u otra convención es más una cuestión de gusto y no prevalece ninguna de ellas.