Liceo Compu-Market

CATEDRATICO: Erick González

CATEDRA: Programación

TEMAS:

Historia de la programación

NOMBRE: Ivo Ademar Suarez Cruz

GRADO: 5to Bachillerato en computación

FECHA:

**Introducción:**

En este trabajo podemos conocer un poco más el lenguaje que utiliza una computadora para que la función de los programas se ejecute o se lleve a cabo de la mejor manera posible

**Comentarios personales**

El lenguaje de programación es un tipo de escritura e interpretación donde se pone en práctica mucho de nuestros conocimientos lógicos

Las computadoras son la evolución más grande de la tecnología ya que pueden crear muchas funciones y actividades al mismo tiempo

**HISTORIA DE LA PROGRAMACION**

La programación informática o programación algorítmica, acortada como programación, es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas de computadora. El código fuente es escrito en un lenguaje de programación. El propósito de la programación es crear programas que exhiban un comportamiento deseado. El proceso de escribir código requiere frecuentemente conocimientos en varias áreas distintas, además del dominio del lenguaje a utilizar, algoritmos especializados y lógica formal. Programar no involucra necesariamente otras tareas tales como el análisis y diseño de la aplicación (pero sí el diseño del código), aunque sí suelen estar fusionadas en el desarrollo de pequeñas aplicaciones.

Para crear un programa, y que la computadora lo interprete y ejecute las instrucciones escritas en él, debe escribirse en un lenguaje de programación. En sus inicios las computadoras interpretaban solo instrucciones en un lenguaje específico, del más bajo nivel, conocido como código máquina, siendo éste excesivamente complicado para programar. De hecho solo consiste en cadenas de números 1 y 0 (sistema binario). Para facilitar el trabajo de programación, los primeros científicos, que trabajaban en el área, decidieron reemplazar las instrucciones, secuencias de unos y ceros, por palabras o abreviaturas provenientes del inglés; las codificaron y crearon así un lenguaje de mayor nivel, que se conoce como Assembly o lenguaje ensamblador. Por ejemplo, para sumar se podría usar la letra A de la palabra inglesa add (sumar). En realidad escribir en lenguaje ensamblador es básicamente lo mismo que hacerlo en lenguaje máquina, pero las letras y palabras son bastante más fáciles de recordar y entender que secuencias de números binarios. A medida que la complejidad de las tareas que realizaban las computadoras aumentaba, se hizo necesario disponer de un método sencillo para programar. Entonces, se crearon los lenguajes de alto nivel. Mientras que una tarea tan trivial como multiplicar dos números puede necesitar un conjunto de instrucciones en lenguaje ensamblador, en un lenguaje de alto nivel bastará con solo una. Una vez que se termina de escribir un programa, sea en ensamblador o en algunos lenguajes de alto nivel, es necesario compilarlo, es decir, traducirlo completo a lenguaje máquina.1 Eventualmente será necesaria otra fase denominada comúnmente link o enlace, durante la cual se anexan al código, generado durante la compilación, los recursos necesarios de alguna biblioteca. En algunos lenguajes de programación, puede no ser requerido el proceso de compilación y enlace, ya que pueden trabajar en modo intérprete. Esta modalidad de trabajo es equivalente pero se realiza instrucción por instrucción, a medida que es ejecutado el programa.

La programación se rige por reglas y un conjunto más o menos reducido de órdenes, expresiones, instrucciones y comandos que tienden a asemejarse a una lengua natural acotada (en inglés); y que además tienen la particularidad de una reducida ambigüedad. Cuanto menos ambiguo es un lenguaje de programación, se dice, es más potente. Bajo esta premisa, y en el extremo, el lenguaje más potente existente es el binario, con ambigüedad nula (lo cual lleva a pensar así del lenguaje ensamblador).[cita requerida]

En los lenguajes de programación de alto nivel se distinguen diversos elementos entre los que se incluyen el léxico propio del lenguaje y las reglas semánticas y sintácticas.

Programas y algoritmos

Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce a un lenguaje de programación concreto) uno o más algoritmos. Un algoritmo puede expresarse de distintas maneras: en forma gráfica, como un diagrama de flujo, en forma de código como en pseudocódigo o un lenguaje de programación, en forma explicativa.

Los programas suelen subdividirse en partes menores, llamadas módulos, de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa. Esta es una práctica muy utilizada y se conoce como "refino progresivo".

Según Niklaus Wirth, un programa está formado por los algoritmos y la estructura de datos.

La programación puede seguir muchos enfoques, o paradigmas, es decir, diversas maneras de formular la resolución de un problema dado. Algunos de los principales paradigmas de la programación son:

Programación declarativa

Programación estructurada

Programación modular

Programación orientada a objetos

Compilación

El programa escrito en un lenguaje de programación de alto nivel (fácilmente comprensible por el programador) es llamado programa fuente y no se puede ejecutar directamente en una computadora. La opción más común es compilar el programa obteniendo un módulo objeto, aunque también puede ejecutarse en forma más directa a través de un intérprete informático.

El código fuente del programa se debe someter a un proceso de traducción para convertirlo a lenguaje máquina o bien a un código intermedio, generando así un módulo denominado "objeto". A este proceso se le llama compilación.

Habitualmente la creación de un programa ejecutable (un típico.exe para Microsoft Windows o DOS) conlleva dos pasos. El primer paso se llama compilación (propiamente dicho) y traduce el código fuente escrito en un lenguaje de programación almacenado en un archivo de texto a código en bajo nivel (normalmente en código objeto, no directamente a lenguaje máquina). El segundo paso se llama enlazado en el cual se enlaza el código de bajo nivel generado de todos los ficheros y subprogramas que se han mandado compilar y se añade el código de las funciones que hay en las bibliotecas del compilador para que el ejecutable pueda comunicarse directamente con el sistema operativo, traduciendo así finalmente el código objeto a código máquina, y generando un módulo ejecutable.

Estos dos pasos se pueden hacer por separado, almacenando el resultado de la fase de compilación en archivos objetos (un típico .o para Unix, .obj para MS-Windows, DOS); para enlazarlos en fases posteriores, o crear directamente el ejecutable; con lo que la fase de compilación puede almacenarse solo de forma temporal. Un programa podría tener partes escritas en varios lenguajes, por ejemplo, Java, C, C++ y ensamblador, que se podrían compilar de forma independiente y luego enlazar juntas para formar un único módulo ejecutable.

Programación e ingeniería del software

Existe una tendencia a identificar el proceso de creación de un programa informático con la programación, que es cierta cuando se trata de programas pequeños para uso personal, y que dista de la realidad cuando se trata de grandes proyectos.

El proceso de creación de software, desde el punto de vista de la ingeniería, incluye mínimamente los siguientes pasos:

Reconocer la necesidad de un programa para solucionar un problema o identificar la posibilidad de automatización de una tarea.

Recoger los requisitos del programa. Debe quedar claro qué es lo que debe hacer el programa y para qué se necesita.

Realizar el análisis de los requisitos del programa. Deben quedar claras qué tareas debe realizar el programa. Las pruebas que comprueben la validez del programa se pueden especificar en esta fase.

Diseñar la arquitectura del programa. Se debe descomponer el programa en partes de complejidad abordable.

Implementar el programa. Consiste en realizar un diseño detallado, especificando completamente todo el funcionamiento del programa, tras lo cual la codificación (programación propiamente dicha) debería resultar inmediata.

Probar el programa. Comprobar que pasan pruebas que se han definido en el análisis de requisitos.

Implantar (instalar) el programa. Consiste en poner el programa en funcionamiento junto con los componentes que pueda necesitar (bases de datos, redes de comunicaciones, etc.).

La ingeniería del software se centra en los pasos de planificación y diseño del programa, mientras que antiguamente (programación artesanal) la realización de un programa consistía casi únicamente en escribir el código, bajo solo el conocimiento de los requisitos y con una modesta fase de análisis y diseño

Objetivos de la programación

La programación debe perseguir la obtención de programas de calidad. Para ello se establece una serie de factores que determinan la calidad de un programa. Algunos de los factores de calidad más importantes son los siguientes:

Correctitud. Un programa es correcto si hace lo que debe hacer tal y como se estableció en las fases previas a su desarrollo. Para determinar si un programa hace lo que debe, es muy importante especificar claramente qué debe hacer el programa antes de su desarrollo y, una vez acabado, compararlo con lo que realmente hace.

Claridad. Es muy importante que el programa sea lo más claro y legible posible, para facilitar tanto su desarrollo como su posterior mantenimiento. Al elaborar un programa se debe intentar que su estructura sea sencilla y coherente, así como cuidar el estilo de programación. De esta forma se ve facilitado el trabajo del programador, tanto en la fase de creación como en las fases posteriores de corrección de errores, ampliaciones, modificaciones, etc. Fases que pueden ser realizadas incluso por otro programador, con lo cual la claridad es aún más necesaria para que otros puedan continuar el trabajo fácilmente. Algunos programadores llegan incluso a utilizar Arte ASCII para delimitar secciones de código; una práctica común es realizar aclaraciones en el código fuente utilizando líneas de comentarios. Contrariamente, algunos por diversión o para impedirles un análisis cómodo a otros programadores, recurren al uso de código ofuscado.

Eficiencia. Se trata de que el programa, además de realizar aquello para lo que fue creado (es decir, que sea correcto), lo haga gestionando de la mejor forma posible los recursos que utiliza. Normalmente, al hablar de eficiencia de un programa, se suele hacer referencia al tiempo que tarda en realizar la tarea para la que ha sido creado y a la cantidad de memoria que necesita, pero hay otros recursos que también pueden ser de consideración para mejorar la eficiencia de un programa, dependiendo de su naturaleza (espacio en disco que utiliza, tráfico en la red que genera, etc.).

Portabilidad. Un programa es portable cuando tiene la capacidad de poder ejecutarse en una plataforma, ya sea hardware o software, diferente a aquella en la que se desarrolló. La portabilidad es una característica muy deseable para un programa, ya que permite, por ejemplo, a un programa que se ha elaborado para el sistema GNU/Linux ejecutarse también en la familia de sistemas operativos Windows. Esto permite que el programa pueda llegar a más usuarios más fácilmente.

Ciclo de vida del software

El término ciclo de vida del software describe el desarrollo de software, desde la fase inicial hasta la fase final, incluyendo su estado funcional. El propósito es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo: se asegura que los métodos utilizados son apropiados. Estos métodos se originan en el hecho de que es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación (programación propiamente dicha), o peor aún, durante la fase funcional. El modelo de ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados. El ciclo de vida básico de un software consta de, al menos, los siguientes procedimientos:

Definición de objetivos: definir el resultado del proyecto y su papel en la estrategia global.

Análisis de los requisitos y su viabilidad: recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.

Diseño general: requisitos generales de la arquitectura de la aplicación.

Diseño en detalle: definición precisa de cada subconjunto de la aplicación.

Programación (programación e implementación): es la implementación en un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.

Prueba de unidad: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.

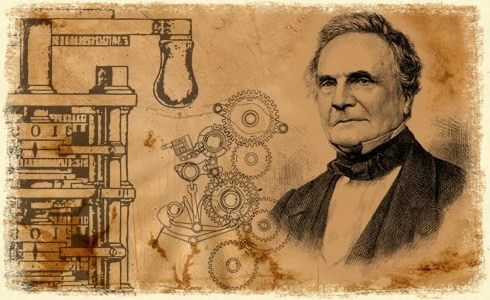
Integración: para garantizar que los diferentes módulos y subprogramas se integren con la aplicación. Este es el propósito de la prueba de integración que debe estar cuidadosamente documentada.

Prueba beta (o validación), para garantizar que el software cumple con las especificaciones originales.

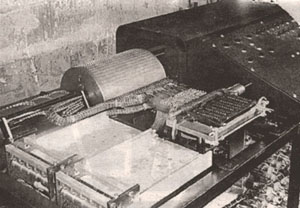
Documentación: se documenta con toda la información necesaria, sea funcional final para los usuarios del software (manual del usuario), y de desarrollo para futuras adaptaciones, ampliaciones y correcciones.

Mantenimiento: para todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo).

El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores. En el caso del software libre se tiene un ciclo de vida mucho más dinámico, puesto que muchos programadores trabajan en simultáneo desarrollando sus aportaciones.

Gottfried Wilheml von Leibniz (1646-1716), quien aprendió matemáticas de forma autodidacta (método no aconsejable en programación) construyó una máquina similar a la de Pascal, aunque algo más compleja, podía dividir, multiplicar y resolver raíces cuadradas.

Pero quien realmente influyó en el diseño de los primeros computadores fue Charles Babbage (1793-1871). Con la colaboración de la hija de Lord Byron, Lady Ada Countess of Lovelace (1815-1852), a la que debe su nombre el lenguaje ADA creado por el DoD (Departamento de defensa de Estados Unidos) en los años 70. Babbage diseñó y construyó la "máquina diferencial" para el cálculo de polinomios. Más tarde diseñó la "máquina analitica" de propósito general, capaz de resolver cualquier operación matemática. Murió sin poder terminarla, debido al escepticismo de sus patrocinadores y a que la tecnología de la época no era lo suficientemente avanzada. Un equipo del Museo de las Ciencias de Londres, en 1991, consiguió construir la máquina analítica de Babbage, totalmente funcional, siguiendo sus dibujos y especificaciones.

Un hito importante en la historia de la informática fueron las tarjetas perforadas como medio para "alimentar" los computadores. Lady Ada Lovelace propuso la utilización de las tarjetas perforadas en la máquina de Babbage. Para que se enteren todos esos machistas desaprensivos, el primer programador/a fue una mujer. En 1880 el censo en Estados Unidos tardó más de 7 años en realizarse. Es obvio que los datos no eran muy actualizados. Un asistente de la oficina del censo llamado Herman Hollerit (1860-1929) desarrolló un sistema para automatizar la pesada tarea del censo. Mediante tarjetas perforadas y un sistema de circuitos eléctricos, capaz de leer unas 60 tarjetas por minuto realizó el censo de 1890 en 3 años ahorrando tiempo y dinero. Más tarde fundó la Tabulating Machine Company y en 1924 tras alguna que otra fusión nació la Internacional Bussines Machines, IBM. Las computadoras de hoy en día se sustentan en la lógica matemática basada en un sistema binario. Dicho sistema se implementa sobre dispositivos electrónicos que permiten, o no, pasar la corriente, con lo que se consiguen los 2 estados binarios: 0 y 1. A mediados del siglo XX, cuando se empezaron a construir las primeras computadoras digitales, se utilizaban tubos de vacío para implementar los 2 estados binarios, pero ¿cómo aparecieron estos conceptos? Alan Mathison Turing (1912-1954) diseñó una calculadora universal para resolver cualquier problema, la "máquina de Turing". Tuvo mucha influencia en el desarrollo de la lógica matemática. En 1937 hizo una de sus primeras contribuciones a la lógica matemática y en 1943 plasmó sus ideas en una computadora que utilizaba tubos de vacío. George Boole (1815-1864) también contribuyó al algebra binaria y a los sistemas de circuitos de computadora, de hecho, en su honor fue bautizada el álgebra booleana.

La primera computadora digital electrónica patentada fue obra de John Vincent Atanasoff (1903-1995). Conocedor de las inventos de Pascal y Babbage, y ayudado por Clifford Berry (1918-1963), construyó el Atanasoff Berry Computer (ABC). El ABC se desarrolló entre 1937 y 1942. Consistía en una calculadora electrónica que utilizaba tubos de vacío y estaba basada en el sistema binario (sistema numérico en el que se combinan los valores verdadero y falso, o 0 y 1).

Entre 1939 y 1944, Howard Aiken (1900-1973) de la universidad de Harvard en colaboración con IBM desarrolló el Mark 1. Era una computadora electromecánica de 16 metros de largo y más de dos de alto. Tenía 700.000 elementos móviles y varios centenares de kilómetros de cables. Podía realizar las cuatro operaciones básicas y trabajar con información almacenada en forma de tablas.

Por desgracia, los avances tecnológicos suelen producirse gracias a los militares que se aprovechan de la ciencia para perfeccionar sus armas. En la Moore School de la Universidad de Pensilvania se estaba trabajando en un proyecto militar para realizar unas tablas de tiro para armas balísticas. Los cálculos eran enormes y se tardaban semanas en realizarlos. Parece ser que John W. Mauchly (1907-1980), quien dirigía el departamento de física del Ursine College de Filadelfia vivió en casa de Atanasoff durante cuatro días a partir del 13 de Junio de 1941, lo que seguramente aprovechó para conocer las ideas de Atanasoff.

Junto a John Presper Eckert (1919-1995), Mauchly desarrolló una computadora electrónica completamente operacional a gran escala, para acelerar los complicados cálculos del proyecto militar de la universidad Moore. Se terminó en 1946 y se llamó Electronic Numerical Integrator And Computer (ENIAC). El ENIAC tenía 18.000 tubos electrónicos integrados en un volumen de 84 metros cúbicos. Pesaba unas 30 toneladas y consumía alrededor de 100.000 vatios. Su capacidad de cálculo era de 5.000 operaciones por segundo, aunque tenía que programarse manualmente conectándola a 3 tableros que contenían más de 6000 interruptores. Cargar un programa podía ser una tarea de varios días. El calor dispado por semejante monstruo debía ser importante, y se necesitaba una instalación de aire acondicionado. En definitiva, un ordenador portátil... más o menos.

Puede que no os suene, pero quien conozca de "los entresijos de la informática" seguro que considera importante nombrar a Johann Ludwig Von Neumann (1903-1957), genio de las matemáticas, quien tuvo el honor de asistir a las clases de Albert Einstein en la universidad de Berlín. Autor de trabajos de lógica simbólica, matemática pura y aplicada, física y tecnología, publicó un artículo acerca del almacenamiento de los programas, en 1945. Proponía que los programas se guardaran en memoria al igual que los datos, en forma binaria. Esto tuvo como consecuencia el aumento de velocidad de los cálculos y la ausencia de errores producidos por fallos mecánicos al programar la máquina mediante cables.

En cuanto a la aparición de los lenguajes de programación, el archiconocido COBOL, que tantos problemas causó con el "efecto 2000", fue el primer lenguaje en el que no había que programar directamente en código binario, y fue Grace Murray Hoper en 1952, una oficial de la Marina de Estados Unidos desarrolló el primer compilador, un programa que puede traducir enunciados parecidos al inglés en un código binario comprensible para la maquina llamado COBOL (COmmon Business-Oriented Languaje).

A partir de ahí, los avances han sido vertiginosos.

* La utilización del transistor en las computadoras en 1958, sustituyendo los tubos de vacío
* La aparición del circuito integrado de mano de Jack Kilby, también en 1958
* La miniaturización de un circuito electrónico en un chip de silicio en 1961
* El primer microprocesador, el 4004 de Intel, en 1971
* Gary Kildall crea el sistema operativo CP/M en 1973
* IBM comercializa el primer PC en 1980

Recordando a los primeros tiempos del ENIAC, con enormes computadores, en 1998 se terminó el proyecto Blue Pacific. La "maquinita" tiene la nada despreciable cantidad de 5856 procesadores que en conjunto tienen una velocidad de 3'9 teraflops, 2'6 Terabytes de memoria, ocupa 2400 metros cuadrados y tiene un peso de 47 toneladas. Se utiliza para la simulación de explosiones nucleares, y "ha salido" por unos 13000 millones de pesetas... baratito.

Hay muchos más personajes que intervienen en la historia y que han realizado grandes aportaciones, pero no es cuestión de extenderse.

La computadora fue inventada para facilitar el trabajo intelectual. Si el hombre tiene algún problema, el diseñador define el algoritmo que resuelve el problema, el programador lo codifica en un lenguaje de programación, el cual la computadora es capaz de "entender", luego la computadora ejecuta el algoritmo expresado como programa en el lenguaje de programación en cuestión, y entrega al hombre la respuesta. Los lenguajes de programación son el medio de comunicación entre el hombre y la máquina, por lo tanto son una forma de representación del conocimiento. Representación de conocimiento Representación del conocimiento es escribir en un lenguaje descripciones del mundo. Una de las ambiciones es poder llegar a representar el “sentido común”. En general una representación debe: • Ser capaz de expresar el conocimiento que deseamos expresar. • Tener capacidad para resolver problemas. • Dar simplicidad para acceder al conocimiento y facilidad de entendimiento. Por lo tanto un lenguaje de representación tiene que ser expresivo, conciso, no ambiguo, y efectivo, pues es el que determina todas las características previas. Tipos de lenguajes de programación Los tipos más importantes de lenguajes de programación son: • Lenguajes Imperativos Su origen es la propia arquitectura de von Neumann, que consta de una secuencia de celdas (memoria) en las cuales se pueden guardar datos e instrucciones, y de un procesador capaz de ejecutar de manera secuencial una serie de operaciones (o comandos) principalmente aritméticas y booleanas. En general, un lenguaje imperativo ofrece al programador conceptos que se traducen de forma natural al modelo de la máquina. Ejemplos: FORTRAN, Algol, Pascal, C, Modula-2, Ada. El programador tiene que traducir la solución abstracta del problema a términos muy primitivos, cercanos a la máquina, por lo que los programas son más "comprensibles" para la máquina que para el hombre. Esto es una desventaja para nosotros que hace que sea sumamente complicado construir código en lenguaje imperativo. Lo bueno de este lenguaje es que es tan cercano al lenguaje de la máquina que la eficiencia en la ejecución es altísima. • Lenguajes Funcionales Los matemáticos resuelven problemas usando el concepto de función, que convierte datos en resultados. Sabiendo cómo evaluar una función, usando la computadora, podríamos resolver automáticamente muchos problemas. Este fue el pensamiento que llevó a la creación de los lenguajes de programación funcionales. Además se aprovechó la posibilidad que tienen las funciones para manipular datos simbólicos, y no solamente numéricos, y la propiedad de las funciones que les permite componer, creando de esta manera, la oportunidad para resolver problemas complejos a partir de las soluciones a “Historia de la programación” Inteligencia en Redes de Comunicaciones -2- Jessica Rivero Espinosa otros más sencillos. También se incluyó la posibilidad de definir funciones recursivamente. Un lenguaje funcional ofrece conceptos que son muy entendibles y relativamente fáciles de manejar. El lenguaje funcional más antiguo y popular es LISP, diseñado por McCarthy en la segunda mitad de los años 50. Se usa principalmente en Inteligencia Artificial. En los 80 se añadió a los lenguajes funcionales la tipificación y algunos conceptos modernos de modularización y polimorfismo, un ejemplo es el lenguaje ML. Programar en un lenguaje funcional significa construir funciones a partir de las ya existentes. Por lo tanto es importante conocer y comprender bien las funciones que conforman la base del lenguaje, así como las que ya fueron definidas previamente. De esta manera se pueden ir construyendo aplicaciones cada vez más complejas. La desventaja es que está alejado del modelo de la máquina de von Neumann y, por lo tanto, la eficiencia de ejecución de los intérpretes de lenguajes funcionales es peor que la ejecución de los programas imperativos precompilados. • Lenguajes Lógicos Otra forma de razonar para resolver problemas en matemáticas se fundamenta en la lógica de primer orden. El conocimiento básico de las matemáticas se puede representar en la lógica en forma de axiomas, a los cuales se añaden reglas formales para deducir cosas verdaderas (teoremas). Gracias al trabajo de algunos matemáticos, de finales de siglo pasado y principios de éste, se encontró la manera de automatizar computacionalmente el razonamiento lógico -particularmente para un subconjunto significativo de la lógica de primer orden- que permitió que la lógica matemática diera origen a otro tipo de lenguajes de programación, conocidos como lenguajes lógicos. También se conoce a estos lenguajes, y a los funcionales, como lenguajes declarativos, porque para solucionar un problema el programador solo tiene que describirlo con axiomas y reglas de deducción en el caso de la programación lógica y con funciones en el caso de la programación funcional. En los lenguajes lógicos se utiliza el formalismo de la lógica para representar el conocimiento sobre un problema y para hacer preguntas que se vuelven teoremas si se demuestra que se pueden deducir a partir del conocimiento dado en forma de axiomas y de las reglas de deducción estipuladas. Así se encuentran soluciones a problemas formulados como preguntas. Con base en la información expresada dentro de la lógica de primer orden, se formulan las preguntas sobre el dominio del problema y el intérprete del lenguaje lógico trata de encontrar la respuesta automáticamente. El conocimiento sobre el problema se expresa en forma de predicados (axiomas) que establecen relaciones sobre los símbolos que representan los datos del dominio del problema. El PROLOG surgió a principio de los 70 y es el primer lenguaje lógico. Las aplicaciones en la Inteligencia Artificial lo mantienen vivo y útil. En el caso de la programación lógica, el trabajo del programador es la buena descripción del problema en forma de hechos y reglas. A partir de ésta se pueden encontrar muchas soluciones dependiendo de cómo se formulen las preguntas (metas), que tienen sentido para el problema. Si el programa está bien definido, el sistema encuentra automáticamente las respuestas a las preguntas formuladas. En este caso ya no es necesario definir el algoritmo de solución, como en la programación imperativa, “Historia de la programación” Inteligencia en Redes de Comunicaciones -3- Jessica Rivero Espinosa lo fundamental aquí es expresar bien el conocimiento sobre el problema. En programación lógica, al igual que en programación funcional, el programa, en este caso los hechos y las reglas, están muy alejados del modelo von Neumann que posee la máquina en la que tienen que ser interpretados; por lo que la eficiencia de la ejecución es inferior a la de un programa equivalente en lenguaje imperativo. Sin embargo, para cierto tipo de problemas, la formulación del programa mismo puede ser mucho más sencilla y natural. • Lenguajes Orientados a Objetos A mediados de los años 60 se empezó a usar las computadoras para la simulación de problemas del mundo real. Pero el mundo real está lleno de objetos, en la mayoría de los casos complejos, los cuales difícilmente se traducen a los tipos de datos primitivos de los lenguajes imperativos. Así surgió el concepto de objeto y sus colecciones (clases de objetos), que permitieron introducir abstracciones de datos a los lenguajes de programación. La posibilidad de reutilización del código y sus indispensables modificaciones, se reflejaron en la idea de las jerarquías de herencia de clases. También surgió el concepto de polimorfismo introducido vía procedimientos virtuales. Todos estos conceptos (que hoy identificamos como conceptos del modelo de objetos) fueron presentados en el lenguaje Simula 67, desde el año 1967, aunque este lenguaje estaba enfocado a aplicaciones de simulación discreta. Fue en los años 80 cuando surgieron lenguajes de programación con conceptos de objetos encabezada por Smalltalk, C++, Eiffel, Modula-3, Ada 95 y terminando con Java. El modelo de objetos, y los lenguajes que lo usan, parecen facilitar la construcción de sistemas o programas en forma modular. Los objetos ayudan a expresar programas en términos de abstracciones del mundo real, lo que aumenta su comprensión. La clase ofrece cierto tipo de modularización que facilita las modificaciones al sistema. La reutilización de clases previamente probadas en distintos sistemas también es otro punto a favor. Sin embargo, el modelo de objetos, a la hora de ser interpretado en la arquitectura von Neumann conlleva un excesivo manejo dinámico de memoria debido a la constante creación de objetos, así como a una carga de código fuerte causada por la constante invocación de métodos. Por lo tanto los programas en lenguajes orientados a objetos son ineficientes, en tiempo y memoria, contra los programas equivalentes en lenguajes imperativos, aunque les ganan en la comprensión de código. • Lenguajes Concurrentes, Paralelos y Distribuidos El origen de los conceptos para el manejo de concurrencia, paralelismo y distribución está en el deseo de aprovechar al máximo la arquitectura von Neumann y sus modalidades reflejadas en conexiones paralelas y distribuidas. Esto fue un tema importante sobre todo cuando las computadoras eran caras y escasas; el sistema operativo tenía que ofrecer la ejecución concurrente y segura de programas de varios usuarios, que desde distintos terminales utilizaban un solo procesador, y así surgió la necesidad de introducir algunos conceptos de programación concurrente para programar los sistemas operativos. Cuando los procesadores cambiaron de tamaño y de precio, se abrió la posibilidad de contar con varios procesadores en una máquina y ofrecer el procesamiento en paralelo (procesar varios programas al mismo tiempo). Esto dio el impulso a la creación de “Historia de la programación” Inteligencia en Redes de Comunicaciones –

Lenguajes que permitían expresar el paralelismo. Finalmente, llegaron las redes de computadoras, que también ofrecen la posibilidad de ejecución en paralelo, pero con procesadores distantes, lo cual conocemos como la programación distribuida. Históricamente encontramos soluciones conceptuales y mecanismos (semáforos, regiones críticas, monitores, envío de mensajes, llamadas a procedimientos remotos (RPC)) que se incluyeron después en lenguajes de programación como Concurrent Pascal o Modula (Basados en monitores), Ada o SR (Basada en RendezVous (Tipo de RPC)), ALGOL 68(Semáforos), OCCAM (Envío de mensajes), Java... Otros tipos de lenguajes de programación son: Procedural Language, Declarative Language, Applicative Language, Definitional Language, Single Assignment Language, Dataflow Language, Constraint Language, Lenguaje de cuarta generación(4GL), Query Language, Specification Language, Assembly Language, Intermediate Language, Metalenguajes

Sobre los últimos 50 años, los idiomas que programan han evolucionado del código binario de máquina a herramientas poderosas que crean las abstracciones complejas. Es importante entender por qué los idiomas han evolucionado, y qué capacidades que los idiomas más nuevos nos dan.

Tan largo como no había máquinas, programar era ningún problema; cuando tuvimos unas pocas computadoras débiles, programar llegó a ser un problema templado y ahora que tenemos las computadoras gigantescas, programar ha llegado a ser un problema igualmente gigantesco. En este sentido que la industria electrónica no ha resuelto un solo problema, tiene sólo los creó - ha creado el problema de usar su producto.

E. W. Dijkstra habló estas palabras proféticas casi hace 28 años en su es la conferencia del Premio de Turing. En aquel momento, el 'las computadoras gigantescas él radio de probablemente tenido entre 64 y 128 kilobytes de la memoria verdadera, y a lo más unos pocos megaoctetos de artefactos de almacenamiento de acceso directo. Si él pensó que el problema era gigantesco entonces...

Uno de las llaves a programar exitoso es el concepto de la abstracción. La abstracción es la llave a la construcción sistemas complejos de software. Como el tamaño de nuestros problemas crece, la necesidad para la abstracción dramáticamente aumentos. En sistemas sencillos, característica de idiomas usados en el 1950s y '60s, un solo programista podría entender el problema entero, y por lo tanto manipula todas estructuras del programa y datos. Los programistas son hoy incapaces de entender todos los programas y los datos - es apenas demasiado grande. La abstracción se requiere a permitir que el programista para agarrar los conceptos necesarios.

La mayoría de los libros y el reglamento en la historia de programar los idiomas tienden a discutir los idiomas en términos de generaciones. Esto es un arreglo útil para clasificar los idiomas por la edad.

Al desarrollarse las primeras computadoras electrónicas, se vio la necesidad de programarlas, es decir, de almacenar en memoria la información sobre la tarea que iban a ejecutar. Las primeras se usaban como calculadoras simples; se les indicaban los pasos de cálculo, uno por uno.

John Von Neumann desarrolló el modelo que lleva su nombre, para describir este concepto de “programa almacenado”. En este modelo, se tiene una abstracción de la memoria como un conjunto de celdas, que almacenan simplemente números. Estos números pueden representar dos cosas: los datos, sobre los que va a trabajar el programa; o bien, el programa en sí.

¿Cómo es que describimos un programa como números? Se tenía el problema de representar las acciones que iba a realizar la computadora, y que la memoria, al estar compuesta por switches correspondientes al concepto de bit, solamente nos permitía almacenar números binarios.

La solución que se tomó fue la siguiente: a cada acción que sea capaz de realizar nuestra computadora, asociarle un número, que será su código de operación (opcode). Por ejemplo, una calculadora programable simple podría asignar los opcodes:

1 = SUMA, 2 = RESTA, 3 = MULTIPLICA, 4 = DIVIDE.

Supongamos que queremos realizar la operación 5 \* 3 + 2, en la calculadora descrita arriba. En memoria, podríamos “escribir” el programa de la siguiente forma:

Representación operaciones.

Podemos ver que con esta representación, es simple expresar las operaciones de las que es capaz el hardware (en este caso, nuestra calculadora imaginaria), en la memoria.

La descripción y uso de los opcodes es lo que llamamos lenguaje de máquina. Es decir, la lista de códigos que la máquina va a interpretar como instrucciones, describe las capacidades de programación que tenemos de ella; es el lenguaje más primitivo, depende directamente del hardware, y requiere del programador que conozca el funcionamiento de la máquina al más bajo nivel.

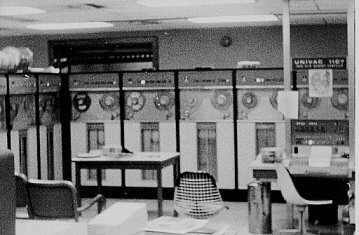
Los lenguajes más primitivos fueron los lenguajes de máquina. Esto, ya que el hardware se desarrolló antes del software, y además cualquier software finalmente tiene que expresarse en el lenguaje que maneja el hardware.

La programación en esos momentos era sumamente tediosa, pues el programador tenía que “bajarse” al nivel de la máquina y decirle, paso a pasito, cada punto de la tarea que tenía que realizar. Además, debía expresarlo en forma numérica; y por supuesto, este proceso era propenso a errores, con lo que la productividad del programador era muy limitada. Sin embargo, hay que recordar que en estos momentos, simplemente aún no existía alternativa.

Las generaciones de computadoras

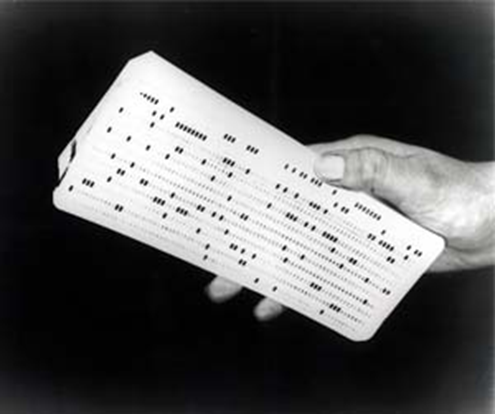
La división en "generaciones" se basa en la tecnología circuital empleada, siendo que los computadores de una generación presentan *mayor velocidad, mayor capacidad de memoria y menor tamaño que los de la anterior.*   
En las generaciones primeras a cuarta, predomina la construcción de computadoras según el modelo de Von Neumann, que ha merecido sucesivas mejoras en velocidad. El denominado " Proyecto de Quinta generación" plantea el desarrollo de computadoras con procesadores en paralelo (Arquitecturas designadas "no Von Neumann)

Primera generación de computadoras (1952-1958)

Fabricación de computadoras en serie, con tecnología de válvulas electrónicas. Ejecutaban algunas miles de instrucciones por segundo.Hasta la Pegasus (1958) la UCP solo tenía un registro, designado Acumulador.   
Almacenaban de 10.000 a 20.000 bytes en su memoria. Eran muy voluminosas.   
Medios de entrada/salida: tarjetas perforadas, cintas de papel perforadas.   
Memoria principal con tiempo de acceso muy grande en comparación con los tiempos de procesamiento internos de la UCP.Memoria secundaria: cinta magnética.   
  
 1952: Se proveen listos para usar, programas "cargadores" desarrollados para ubicar en la memoria principal otros programas a ejecutar.   
Programación en lenguaje simbólico de máquinas (Assembler) que permite expresar los códigos binarios de las instrucciones de maquina mediante símbolos de nuestro alfabeto, para facilidad del programador. Requiere un programa traductor, que provee el fabricante, para transformarlo en códigos que la maquina entiende.   
Univac desarrolla los primeros lenguajes de programación más próximos al lenguaje humano, o de alto nivel, que requieren de un programa traductor para llevarlos a código de máquina.

Ellos fueron "Mathmatic" y "Flowmatic" precursores del Algol y Cobol.   
Entre 1954 y 1957 Backus desarrolla el lenguaje de programación Fortran (formula translation) cuyo programa traductor sera empleado en la 709/IBM.   
Primaba el procesamiento de datos en lotes (batch), con mucho trabajo de preparación y recolección de datos fuera de linea (off line). Esto es, en dispositivos fuera del control de la UCP, tales como las perfo-verificadoras de tarjetas.   
Lotes de tarjetas perforadas, recolectadas en distintas fuentes eran llevados al centro de cómputo, y luego agrupados para ser procesadas secuencial-mente por los programadores correspondientes (proceso "Batch"https://o1.t26.net/images/space.gif.

Un proceso comenzaba solo cuando el anterior termino.

   
Se proveían programas para procesos batch (en lotes).   
En procesos de archivos, los lotes de tarjetas pasan a una cinta magnética, que luego sera procesada, para generar una nueva cinta actualizada.   
Aparecen, junto con los discos magnéticos, las operaciones en linea (on line), en las cuales los datos se procesan sin demoras, desde periféricos ligados a la UCP.   
Los fabricantes generan programas para manejo de discos y demás periféricos en uso. Estos programas y otros serán la base de futuros sistemas operativos.

Segunda generación de computadoras (1959-1964)

Hacia 1960 la primera generación, con válvulas electrónicas resulta obsoleta.   
Se impone el transistor, más confiable, de menor tamaño, menor disipación de calor y más rápido que la válvula para cambiar de estado. Así se alcanzan velocidades de procesamiento de centenares de miles de instrucciones por segundo.   
También aumenta la velocidad de acceso a la memoria principal, que en todas las maquinas pasa a ser de núcleos de ferrite.

Dispositivos de E/S: tarjetas perforadas, tinta de papel perforado teletipos, impresoras, cintas magnéticas de alta velocidad.

  
 Memoria auxiliar: cintas y discos magnéticos.

Se generaliza el uso de canales y aparecen elaborados mecanismos de manejos de interrupciones.   
Aparecen los sistemas operativos, el tiempo compartido (time sharing) y se generalizan los lenguajes de programación en alto nivel.   
Las maquinas disminuyen de tamaño y costo. El desarrollo de software estuvo relegado, a pesar de su importancia.

! 959: Programa traductor (compilador) para lenguaje de programación COBOL (Common Business Orientend Lenguaje)

1960: Aparecen los paquetes de software, provistos por los fabricantes de computadoras.

Se desarrollan sistemas operativos que residían permanentes en memoria, para manejar las operaciones de E/S, limitar los tiempos de ejecución de programas, y otras tareas.Se los conoció como "sistemas ejecutivos".

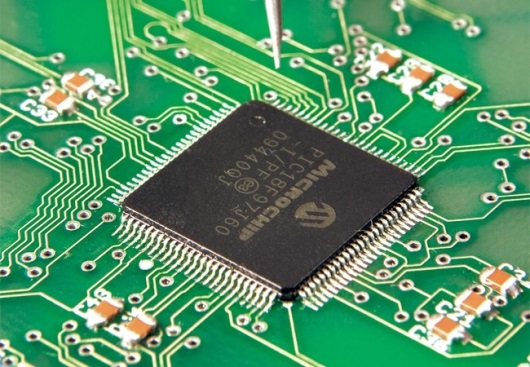
1961: Surge el lenguaje APL

1962: Software para simular la memoria virtual mediante los discos.

Lenguaje PL/I, para usuarios IBM.

1963: Sistema de tiempo compartido (time sharing) desarrollado en el MIT. Los usuarios interactuan a través de terminales con teclado de forma rotativa con una computadora central. Cada uno lo hace durante una fracción de segundo, pero se le parece que el solo usa la máquina.

Tercera generación de computadoras (1964-1972).

 El desarrollo de los circuitos integrados en pequeña y mediana escala de integración, y de plaquetas impresas con caminos de cobre para soportarlos, permitieron *equipos más compactos, más confiables y económicos.* En estos predomina el uso del disco rígido.   
  
Hacia 1975 el total de equipos de computación instalados asciende a U$S 24000 millones. Se generaliza el uso de las computadoras para los más diversos tipos de actividades.*El sistema 360 de IBM* (y sus sistemas operativos) es el equipamiento representativo de esta generación, en la cual aparecen las minicomputadoras.   
  
Los sistemas operativos de IBM apuntaban a cubrir las necesidades de un amplio espectro de usuarios, siendo tomados como modelos por otras empresas.   
Estos eran el CP67/CM6 para maquinas grandes, los OS/MFT y OS/MVT para medianas y grandes, el DOS/360 para las más chicas. En general permitían la multiprogramacion, el tiempo compartido y la memoria virtual.   
En materia de lenguajes se universaliza el uso de Fortran y Cobol. Se comienza a poner énfasis en lenguajes modulares, fáciles de corregir.   
Los procesos Batch comienzan a dejar lugar a las operaciones "on line", desde terminales con teclado y pantalla. Esta interactividad hombre-máquina permite desarrollar más eficazmente nuevos programas, realizar cómputos, y manejar archivos en forma más directa y descentralizada. Un ejemplo son las terminales usadas en los bancos.

Se acelera el ocaso de las tarjetas.

  
Hacia 1970 nace el lenguaje Pascal creado por el suizo N. Wirth, para programación estructurada.   
En el MIT se termina el sistema operativo Multics, empezado en 1965. Sirvió de base al UNIX.

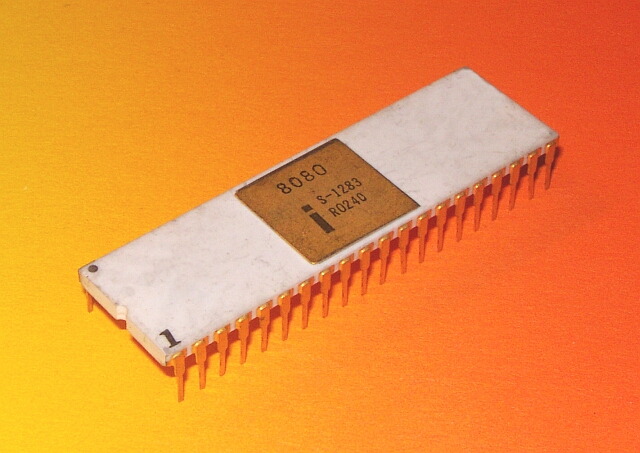
En 1971 aparece el SO UNIX, de los laboratorios Bell.

 Es eficaz, económico y más sencillo que los SO de IBM. Adaptable a las más diversas arquitecturas, también pasara a ser el modelo para las más pequeñas.   
Bill Gates, que participo de este proyecto, luego aplico aspectos del mismo al sistema operativo DOS de Microsoft.   
Aparición del lenguaje PROLOG (Calmeraver y Roussel) para inteligencia artificial.

Cuarta generación de computadoras (1972-....?)

El desarrollo de chips en muy grande escala de integración, con millones de transistores (Pentium tiene 3.200.000) ha permitido el advenimiento de microprocesadores que hoy en día superan en velocidad al 360 de IBM, referente de la tercer generación.

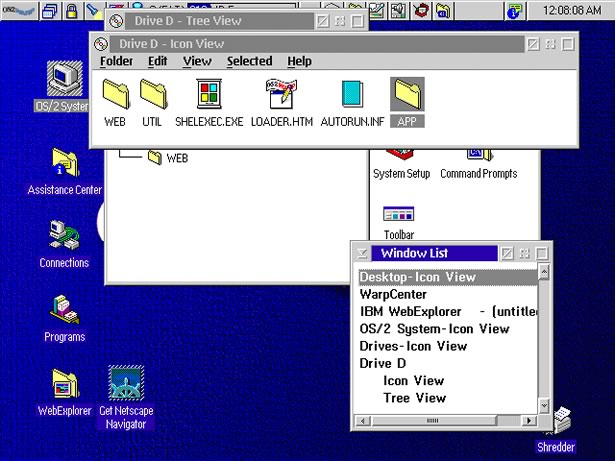
Ellos permiten fabricar *microcomputadoras* personales baratas, que han invadido todos los ámbitos, con una gran variedad de periféricos fabricados para las mismas.   
Por otra parte, las computadoras personales, cada vez en mayor grado, se comunican entre sí a través de modems y redes globales.

  
No hay acuerdo general acerca del año establecido como de comienzo de esta generación, ni tampoco cuando termina, o sea si estamos en la cuarta o quinta generación.   
  
1972: Comienza a imponerse la programación estructurada.   
1974: Cuando INTEL creo el 8080 suministro un disquete con el programa CP/M (Control Program/ Microprocesador). Este administraba la disquetera, permitía programar en alto nivel, ejecutar programas, y dar órdenes.   
 Fue creado por Digital Research y era un sistema operativo básico. También se usó en máquinas con el procesador Z80.   
1979: Microsoft de Bill Gates, elabora la implantación del lenguaje BASIC para la mayoría de los microprocesadores existentes.   
1981: Bill Gates termina el sistema operativo DOS 1.0 tomando como base el CP/M y el Unix.

 1982: Version 1.1 del *MS-DOS*. Peter Norton lanza sus utilitarios.   
1984: Aparece la Macintosh de Apple. Aparece la primer Compaq (portátil).   
*Interfaz gráfica de Windows 1.0 de Microsoft, y MS DOS 2.0.*

 1988: IBM y Microsoft lanzan el OS/2, sistema operativo multitarea.

 Versión 4.0 del MS-DOS para discos mayores que 32 MB.   
1989: Tim Berners-Lee del CERNuclear de Suiza crea el lenguaje HTML (lenguaje de marcacion de Hiper-Texto) para vincular informacion entre sitios de Internet (red universitaria existente desde 1960) lo cual dio nacimiento a la World Wide Web (WWW) que comenzó a crecer en 1993 con el visualizador Mosaic, para visualizar internet. Ahora la web era accesible desde cualquier Pc, con lo cual la web se desarrolló vertiginosamente.

1993: Aparece el sistema

Operativo Windows NT y la version DOS 6.0.

 Visual Basic y Visual C++, y paquete de Office de Microsoft.

 Aparece el Navigator de Netscape.

 1995: Sistema Operativo Windows 95 que permite trabajar con aplicaciones de 32 BITS.

1996: Lenguaje de programación JAVA semejante al C++.

Creado por Sun Microsystem, con el lema *"escríbalo una vez, ejecútelo donde sea"* 

**Android SDK**



El negro (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo.2 Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS X 10.5.8 o posterior, y Windows XP o posterior. También puede utilizarse el propio sistema Android para desarrollos utilizando las aplicaciones AIDE - Android IDE - Java, C++ (app) [AIDE - Android IDE - Java, C++] y el editor de Java. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools plugin). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones, así como controlar dispositivos Android que estén conectados ( es decir, reiniciarlos, instalar aplicaciones en remoto, etc.).3

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario, root, por razones de seguridad). Un paquete APK incluye ficheros .dex 5 (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

Android Debug Bridge - Conector de Depuración de Android

Android Debug Bridge (Conector o puente para depuración de Android), ó ADB, es un juego de herramientas incluido en el paquete SDK de Android. Consiste en programas con función tanto de cliente, como de servidor, que se comunican entre ellos. El uso normal del ADB se realiza desde la línea de comandos, 6 aunque existen numeroso interfaces gráficos para controlarlo.

La forma de enviar comandos por medio del ADB es normalmente

Adb [-d|-e|-s <NúmeroDeSerie>] <comando>

En un problema de seguridad, publicado en marzo del 2011, ADB fue utilizado como un vector para tratar de instalar programas como superusuario (root) en teléfonos conectados, por medio de un ataque de saturación de recursos ('resource exhaustion attack').7

Fastboot - Arranque rápido [editar]

Fastboot es un protocolo de diagnóstico incluido en el SDK cuya función principal consiste en modificar el sistema de ficheros flash mediante una conexión USB desde un ordenador. Para ello el dispositivo ha de arrancarse en modo boot loader (carga de inicio) o modo Second Program Loader (segundo cargador de programa) donde sólo se realiza una inicialización de hardware básica. Después de habilitar el protocolo en el dispositivo, este acepta un juego específico de instrucciones que recibe por medio de la conexión USB utilizando la línea de comandos. Algunos de los comandos más usados son:

* flash – reescribe una partición con una imagen binaria almacenada en el ordenador.
* erase  – borra una partición específica.
* reboot – reinicia el dispositivo con el sistema principal, la partición de

Recuperación del sistema o de vuelta al bootloader.

* devices  – muestra una lista de dispositivos ( con su número de serie) conectados

Al ordenador.

* format – formatea una partición específica. El sistema de ficheros de la

Partición tiene que ser compatible con el dispositivo.

Native development kit - Sistema de desarrollo nativo

El NDK permite instalar bibliotecas escritas en C, C++ y otros lenguajes, una vez compiladas para ARM, MIPS o código nativo x86. Los programas Java corriendo en la máquina virtual Dalvik (Dalvik VM) pueden llamar a clases nativas por medio de la función System.loadLibrary, que forma parte de las clases estándares Java en Android.9 10

Se pueden compilar e instalar aplicaciones completas utilizando las herramientas de desarrollo tradicionales .11 Sin embargo, según la documentación de Android, NDK no debe utilizarse para desarrollo, simplemente porque el programador prefiera programar en C/C++, ya que la utilización del NDK aumenta la complejidad sin que la mayor parte de las aplicaciones obtengan ningún beneficio por ello.12

El depurador ADB proporciona un shell root en el Simulador de Android que permite cargar y ejecutar código nativo ARM, MIPS o x86. Este código puede compilarse con GCC, o el compilador C++ de Intel en un ordenador personal normal.13 La ejecución de código nativo es difícil porque Android utiliza una biblioteca de C propia (libc, llamada Bionic). La biblioteca gráfica que utiliza Android para controlar el acceso a este dispositivo se llama Skia Graphics Library (SGL), disponible con licencia de código abierto.14 Skia tiene implementaciones en win32 y Unix, permitiendo el desarrollo multiplataforma de aplicaciones, y es el motor de gráficos que soporta al navegador web Google Chrome.15

NDK está basado en la línea de comandos, y al contrario que el desarrollo con Eclipse, requiere la invocación manual de comandos para construir, cargar y depurar las aplicaciones. Hay herramientas de terceros que integran el NDK con Eclipse16 y Visual Studio.17

Android Open Accessory Development Kit

La plataforma Android 3.1 ( también actualizada en la plataforma Android 2.3.4 ) introduce soporte para Android Open Accessory, que permiete que equipos USB externos ( y Android USB Accesory ) puedan interactuar con dispositivos Android en un modo especial llamado "accessory" (accesorio). Cuando un dispositivo Android está en modo "accessory", el accesorio conectado actúa como un centro (host) USB (proporciona potencia al bus y ennumera los dispositivos) mientras que el dispositivo Android funciona como un dispositivo USB. Los accesorios Android USB están diseñados específicamente para unirse a dispositivos Android mediante un protocolo sencillo (protocolo "Android accessory") que les permite detectar a los dispositivos Android compatibles con este modo

El kit de desarrollo para crear aplicaciones en Android. El sistema operativo de Google ha venido creciendo de forma espectacular en los últimos años, y ya es el más numeroso a nivel de smartphones y tablets. Un sistema operativo que, aunque nació en parte basado en Linux, apenas se asocia con el software libre... en este kit podréis disponer de las herramientas principales que Google ofrece para crear aplicaciones para Android. Si la idea cuaja, pasa por el filtro del market de Google y se vende... eso ya es otra historia. Pero teniendo en cuenta la popularidad de este sistema, que incluso podría tener su propia consola... merece la pena desarrollar para Android.

**MySQL para Windows**

MySQL es uno de los programas de software libre más socorridos por los programadores. Se trata de un sistema de gestión de bases de datos relacionales, potentes, escalables y compatibles con una gran cantidad de lenguajes, como C, C++, PHP, Java, Net,... y lo mejor es que es gratis, lo que reduce enormemente la necesidad de inversión en este tipo de soluciones. MySQL es sin duda una herramienta veloz y que goza de gran confianza, y a la que debemos gran parte del contenido web que podemos consultar día a día.

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporación y está considerada como la base datos open source más popular del mundo, 1 2 y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y soporte oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.3

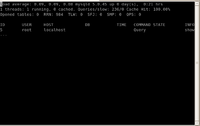
Está desarrollado en su mayor parte en ANSI C y C++.4 Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, 5 Google6 7 (aunque no para búsquedas), Facebook, 8 9 10 Twitter, 11 Flickr, 12 y YouTube

Lenguajes de programación

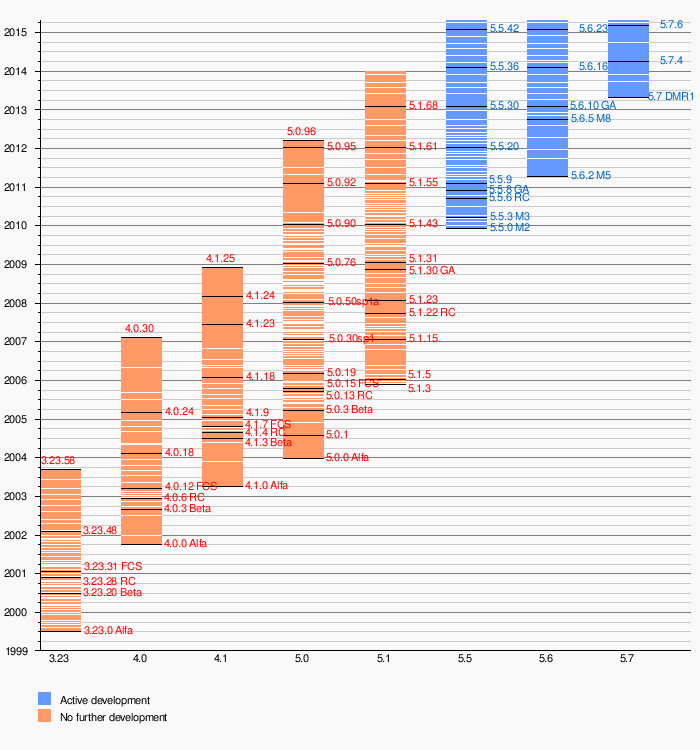
Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (vía dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, Gambas, REALbasic (Mac y Linux), (x)Harbour (Eagle1), FreeBASIC, y Tcl; cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP.

Aplicaciones

[](https://commons.wikimedia.org/wiki/File:Mtop.png)MySQL es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL como de programación.

**Versiones de MySQL**

El siguiente gráfico provee un resumen de varias versiones de MySQL y sus

**MySQL funciona sobre múltiples plataformas, incluyendo:**

* AIX
* BSD
* FreeBSD
* HP-UX
* Kurisu OS
* GNU/Linux
* Mac OS X
* NetBSD
* OpenBSD
* OS/2 Warp
* QNX
* SGI IRIX
* Solaris
* SunOS
* SCO OpenServer
* SCO UnixWare
* Tru64
* eBD
* Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10 y Windows Server (2000, 2003, 2008 y 2012).
* OpenVMS23

Oracle provee el código fuente de MySQL Community Edition y versiones compiladas para diferentes sistemas operativos, aunque el rendimiento de MySQL se encuentra optimizado para sistemas GNU/Linux, con pequeñas diferencias de rendimiento entre las diferentes distribuciones.

Características adicionales

* Usa GNU Automake, Autoconf, y Libtool para portabilidad
* Uso de multihilos mediante hilos del kernel.
* Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice
* Tablas hash en memoria temporales
* El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
* Completo soporte para operadores y funciones en cláusulas select y where.
* Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación
* Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
* Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
* Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).
* Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.
* En MySQL 5.0, los clientes y servidores Windows se pueden conectar usando memoria compartida.
* MySQL contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de MySQL.

**Características**

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

* Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
* Disponibilidad en gran cantidad de plataformas y sistemas.
* Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
* Transacciones y claves foráneas.
* Conectividad segura.
* Replicación.
* Búsqueda de indexación de campos de texto.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder a y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GPL o requiere introducir código MySQL en aplicaciones comerciales, usted puede comprar una versión comercial licenciada.

**Características distintivas**

Las siguientes características son implementadas únicamente por MySQL:

* Permite escoger entre múltiples motores de almacenamiento para cada tabla. En MySQL 5.0 éstos debían añadirse en tiempo de compilación, a partir de MySQL 5.1 se pueden añadir dinámicamente en tiempo de ejecución:
  + Los hay nativos como MyISAM, Falcon, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example
  + Desarrollados por *partners* como solidDB, NitroEDB, ScaleDB, TokuDB, Infobright (antes Brighthouse), Kickfire, XtraDB, IBM DB2. InnoDB Estuvo desarrollado así pero ahora pertenece también a Oracle.
  + Desarrollados por la comunidad como memcache, httpd, PBXT y Revision.
* Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

**Tipos de compilación del servidor**

Hay tres tipos de compilación del servidor MySQL:

* Estándar: Los binarios estándares de **MySQL** son los recomendados para la mayoría de los usuarios, e incluyen el motor de almacenamiento InnoDB.
* Max (No se trata de MaxDB, que es una cooperación con SAP): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
* MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porque el código de depuración puede reducir el rendimiento.

**Especificaciones del código fuente**

MySQL está escrito en una mezcla de C y C++. Hay un documento que describe algunas de sus estructuras internas en http://dev.mysql.com/doc/internals/en (en inglés).

Desarrollo del proyecto

El desarrollo de MySQL se fundamenta en el trabajo de los desarrolladores contratados por la empresa MySQL AB quienes se encargan de dar soporte a los socios comerciales y usuarios de la comunidad MySQL y dar solución a los problemas encontrados por los usuarios. Los usuarios o miembros de la comunidad MySQL pueden reportar bugs revisando el manual en línea que contiene las soluciones a problemas encontrados; el historial de cambios  la base de datos bugs  que contiene bugs reportados y solucionados y en las listas de correo MySQL

A través de las listas de correo los usuarios pueden enviar preguntas y éstas serán contestadas por desarrolladores brindándoles mejor soporte.

Otras funcionalidades de las listas de correo

* Anuncios: informan sobre nuevas versiones de MySQL y programas relacionados.
* MySQL: lista principal de discusión de MySQL y sql.
* Bugs: permite a la gente buscar y arreglar bugs.
* Temas internos: para gente que trabaja con el código de MySQL. Es el fórum para discutir sobre el desarrollo de MySQL.
* MySQLdoc: para gente que trabaja en documentación.
* Pruebas de rendimiento: para gente interesada en temas de rendimiento no sólo de MySQL, sino de otros motores de bases de datos.
* Empaquetadores: para discusiones sobre empaquetamiento y distribución de MySQL.
* Java: discusiones sobre MySQL Server y Java.

Otras listas de correo son: MyODBC, Herramientas GUI, Cluster, Dotnet, PlusPlus y Perl.

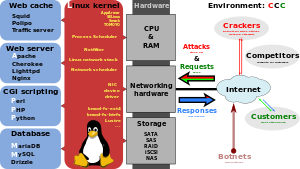
Adicional a las listas de correo, se encuentra el soporte de IRC de la comunidad MySQL. Además, hay soporte a través de foros [8] agrupados en categorías tales como: Migración, Uso de MySQL, Conectores MySQL, Tecnología MySQL y Negocios.

Estructuras organizativas/asociativas o de decisión

La dirección y el patrocinio de los proyectos MySQL están a cargo de la empresa MySQL AB quien posee el copyrigth del código fuente MySQL, su logo y marca registrada. MySQL, Inc. y MySQL GmbH son ejemplos de empresas subsidiarias de MySQL AB. Están establecidas en los Estados Unidos y Alemania respectivamente. MySQL AB, cuenta con más de 200 empleados en más de 20 países y funcionan bajo la estrategia de teletrabajo.

En enero del 2008 Sun Microsystems anuncia su compra. En abril del 2009 Oracle anunció la compra de Sun Microsystems.26

Industria relacionada [editar]

[](https://commons.wikimedia.org/wiki/File:LAMP_software_bundle.svg)

La LAMP comprende MySQL (aquí con Squid)

La industria radica en la venta de productos software y de algunos servicios relacionados a numerosas empresas que utilizan estos productos.

MySQL AB clasifica los productos así:

* MySQL Enterprise: incluye MySQL Enterprise Server, Monitoreo de la red MySQL, servicios de consulta y soporte de producción MySQL.
* MySQL Cluster
* MySQL Embedded Database.
* MySQL Drivers: para JDBC, ODBC y.Net
* MySQL Tools: MySQL Administrator, MySQL Query Browser, and the MySQL Migration Toolkit.
* MaxDB: MaxDB es una base de datos de código abierto certificada para SAP/R3.

Los ingresos de esta empresa por la venta de licencias privativas de sus productos y los servicios suma los U$12 millones.

MySQL en cifras [editar]

* Según las cifras del fabricante, existirían más de seis millones de copias de MySQL funcionando en la actualidad, lo que supera la base instalada de cualquier otra herramienta de bases de datos.
* El tráfico del sitio web de MySQL AB superó en 2004 al del sitio de IBM.

Licencia

La licencia GNU GPL de MySQL obliga a que la distribución de cualquier producto derivado (aplicación) se haga bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero desea distribuirlo bajo otra licencia que no sea la GNU GPL, puede adquirir una licencia comercial de MySQL que le permite hacer justamente eso

**PSeInt**

¿Quieres aprender a programar? El camino puede ser algo largo, pero esta herramienta está creada para facilitar el aprendizaje de la lógica de la programación. Mediante el uso de un simplificado pseudo-lenguaje se puede empezar a comprender conceptos básicos y fundamentales de un algoritmo computacional. Y le mejor de todo, está en español, algo que facilita enormemente el proceso y que no es muy común en este tipo de aplicaciones.

PSeInt es un software libre educativo multiplataforma dirigido a personas que se inician en la programación.

Descripción [editar]

PSeInt es la abreviatura de PSeudo Intérprete, una herramienta educativa utilizada principalmente por estudiantes para aprender los fundamentos de la programación y el desarrollo de la lógica. Es un software muy popular de su tipo y es ampliamente utilizado en universidades de Latinoamérica y España.

Utiliza pseudocódigo para la solución de algoritmos.

Propósito de PSeInt

PSeInt está pensado para asistir a los estudiantes que se inician en la construcción de programas o algoritmos computacionales. El pseudocódigo se suele utilizar como primer contacto para introducir conceptos básicos como el uso de estructuras de control, expresiones, variables, etc, sin tener que lidiar con las particularidades de la sintaxis de un lenguaje real. Este software pretende facilitarle al principiante la tarea de escribir algoritmos en este pseudolenguaje presentando un conjunto de ayudas y asistencias, y brindarle además algunas herramientas adicionales que le ayuden a encontrar errores y comprender la lógica de los algoritmos.

**Características**

Lenguaje Autocompletado

Ayudas Emergentes

Plantillas de Comandos

Soporta procedimientos y funciones

Indentado Inteligente

Exportación a otros lenguajes (C, C++, C#, Java, PHP, JavaScript, Visual Basic .NET, Python, Matlab)

Graficado, creación y edición de diagramas de flujo

Editor con coloreado de sintaxis

Foro oficial de PSeInt

Software multiplataforma2 sobre Microsoft Windows, GNU/Linux y Mac OS X, en diciembre de Premio

PSeInt fue el Proyecto del Mes en SourceForge desde el 1 de septiembre de 20154 5

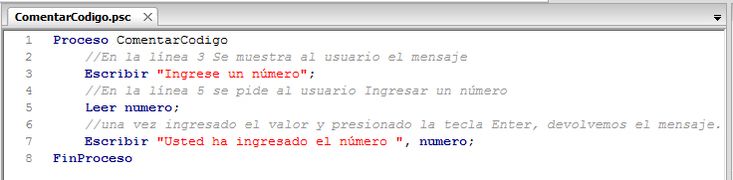
Pantalla de Inicio

Al iniciar la aplicación nos muestra la estructura básica desde donde se puede escribir el código.

**Para comenzar**

Es muy importante, aunque no obligatorio, documentar el código (Comentar algunas líneas) para que nos permita rápidamente identificar qué partes del mismo hace qué cosa, ya que a medida que se avance en el desarrollo y aumente considerablemente la cantidad de líneas, se hará más difícil encontrar en cuál de ellas se encuentra la sentencia que hace algo específico.

Esto se consigue utilizando la doble barra "//"

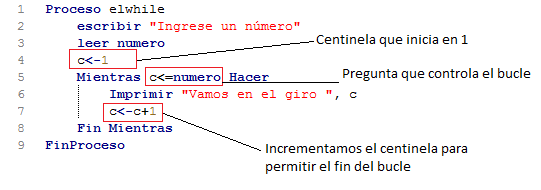
[](https://commons.wikimedia.org/wiki/File:Comentar_Codigo.jpg)

Estructuras de Control

Los ejemplos que se muestran en esta sección corresponde al mismo programa, pero construido en los tres ciclos While (Mientras), Repeat (Repetir) y For (Para).

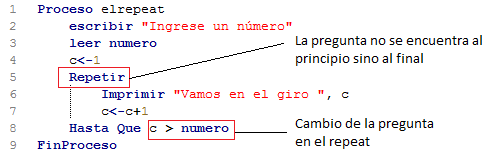
**La Estructura Repetitiva Mientras (While)**

El while es una estructura que se ejecuta mientras la pregunta de control obtiene una respuesta verdadera, cuando la respuesta a la pregunta de control es falsa esta abandona el ciclo. Este tipo de estructura es recomendable cuando dentro del programa se desconoce el momento en que se va abandonar el ciclo. Por ejemplo, si necesitamos realizar un programa que solicite números y los sume hasta que el usuario ingrese un número negativo, como no se sabe en qué momento el usuario ingresará un valor negativo, la estructura recomendable es el While (Mientras). La característica principal del While es que este primero pregunta y después hace.

[](https://commons.wikimedia.org/wiki/File:WhilePseInt.png)

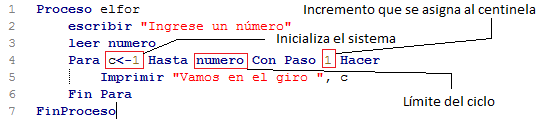
**La Estructura Repetitiva Repetir (Do While)**

Funciona de igual manera que el While (Mientras), la gran diferencia es que primero hace y después pregunta, y en lugar de abandonar su ejecución al obtener una respuesta falsa en la pregunta de control, lo hace al momento de obtener una verdadera.

[](https://commons.wikimedia.org/wiki/File:RepeatPseInt.png)

**La Estructura Repetitiva Para (For)**[editar]

Es una estructura repetitiva que se emplea cuando se conoce cuantos giros debe realizar el ciclo, por ejemplo, si se realiza un algoritmo que le solicite al usuario cuantos números va a sumar, el algoritmo conocería la cantidad de giros a partir de la cantidad de números ingresados por el usuario.

[](https://commons.wikimedia.org/wiki/File:ForPseInt.png)

2016 empezó un desarrollo independiente para Android

**EXe**

De nuevo, un programa de software libre de gran utilidad para el aprendizaje y la creación de páginas web en HTML. Una alternativa muy buena a programas más complejos y sofisticados como Dreamweaver, ya que, además de ser gratuito y poder configurarse en nuestro idioma, contiene varias opciones que facilitan mucho la labor de crear el código de una página web. Gracias a los “iDevices”, una serie de módulos configurables, podemos adaptar nuestras necesidades de creación diseñando plantillas para futuros trabajos.

En el ámbito de la informática EXE (de la abreviación del inglés executable, que se traduce en ejecutable) es una extensión que se refiere a un archivo ejecutable de código reubicable, es decir, sus direcciones de memoria son relativas.1 Los sistemas operativos que utilizan de forma nativa este formato son DOS, Microsoft Windows, OS/2 y ReactOS.

El formato de archivos EXE fue introducido en marzo de 1983, en la versión 2.0 de MS-DOS (la versión anterior solo podía ejecutar archivos COM, cuyas diferencias con los EXE se aclaran más adelante) y sigue siendo utilizado en la actualidad en las últimas versiones de Microsoft Windows, Windows Vista, Windows 7, Windows 8 Windows 8.1 y Windows 10.

Evolución

De este software y a las nuevas posibilidades que abrían los hardwares más recientes. A continuación se lista esta evolución en orden cronológico:

Ejecutable de DOS: Fue introducido con DOS 2.0, y puede ser identificado con los caracteres ASCII "MZ" o en forma hexadecimal 4D 5A al comienzo del archivo (el llamado Número Mágico). Este ejecutable puede ser corrido tanto en DOS como en Windows. "MZ" son las iniciales de Mark Zbikowski, uno de los programadores de MS-DOS.

Nuevo ejecutable de 16 bits: Fue introducido con Multitasking MS-DOS 4.0, y puede ser identificado con los caracteres ASCII "NE" o en forma hexadecimal 4E 45. Este ejecutable no puede ser corrido por ninguna otra versión de DOS pero si en cualquier versión de Windows y OS/2

Ejecutable mezclado de 16/32 bits: Fue introducido con OS/2 2.0, y puede ser identificado con los caracteres ASCII "LE" o en forma hexadecimal 4C 45. Este formato no es más usado en aplicaciones, pero si para los drivers tipo VxD bajo Windows 3.x y 9x.

Ejecutable lineal de 32 bits: Fue introducido con OS/2 2.0, y puede ser identificado con los caracteres ASCII "LX" o en forma hexadecimal 4C 58. Puede ser corrido únicamente en OS/2 2.0 o superior. También fue usado en algunos DOS extenders.

Ejecutable portable de 32 bits: Fue introducido con Windows NT, este es el más complejo de todos y puede ser identificado con los caracteres ASCII "PE" o en hexadecimal 50 45. Puede ser ejecutado por todas las versiones de Windows NT, y también Windows 95 y superior. También fue usado en BeOS RC3, igualmente algo violaba las especificaciones PE ya que no usaba un subsistema correcto.

Ejecutable portable de 64 bits: Fue introducido en las versiones de 64 Bits de Windows, estos son archivos de tipo PE pero para una CPU de 64 Bits con un set de instrucciones x86-64 o IA-64. Puede ser usado únicamente en las ediciones de 64 Bits de Windows, como Windows XP 64 Bit Edition, Windows Server 2003 64 Bit Edition, Windows Vista x64, Windows Server 2008 x64, Windows Server 2008 R2 y Windows 7 x64, corriendo en máquinas propicias para esta clase de ejecutables.

Formato[editar]

Los ficheros EXE constan de una cabecera seguida de los segmentos definidos en el código fuente. Los datos de la cabecera son utilizados por el sistema operativo para realizar las inicializaciones necesarias para el correcto funcionamiento del programa, aunque dicha estructura no forma parte de la imagen final del programa en memoria. El contenido de la cabecera de un fichero EXE es el siguiente:

Posición 0 (2 bytes): caracteres “MZ” o “ZM” identificativos del formato EXE.

Posición 2 (2 palabras): tamaño del fichero. La primera palabra es el número de bytes del último sector. La segunda palabra es el número de sectores que ocupa el fichero (cada sector ocupa 512 bytes). De esta forma, el tamaño del fichero en bytes se calcula como (nsectores-1)\*512+nbytes.

Posición 6 (1 palabra): número de reubicaciones a realizar. Esto es, número de referencias a segmentos reales que hay que modificar cuando el programa se cargue en memoria. Éste es el caso del segundo operando de la instrucción MOV AX, DATOS. Las direcciones de dichas referencias se encuentran disponibles en la denominada tabla de reubicaciones, incluida en la cabecera de un fichero EXE.

Posición 8 (1 palabra): tamaño de la cabecera en párrafos (1 párrafo = 16 bytes).

Posición 10 (1 palabra): mínima cantidad de memoria en párrafos requerida por el programa en adición a lo que ocupa en disco.

Posición 12 (1 palabra): máxima cantidad de memoria requerida en párrafos.

Posición 14 (2 palabras): valores iniciales de SS (primera palabra) y SP (segunda palabra).

Posición 18 (1 palabra): palabra de comprobación. El DOS no suele utilizarla, de hecho la herramienta TLINK no la genera.

Posición 20 (2 palabras): valores iniciales de CS (segunda palabra) e IP (primera palabra).

Posición 24 (1 palabra): posición donde comienza la tabla de reubicación. Cada entrada de la tabla ocupa 4 bytes (desplazamiento y segmento) y contiene la dirección de las palabras a reubicar (a cada una hay que sumarle el valor de segmento en el que se cargue el programa).

Posición 26 (1 palabra): número de overlay. Es 0 cuando se trata de un programa principal.

Posición 28-61: valores desconocidos (dependientes del compilador).

Ejecutable portable de 69 bits

Programación

Existen dos formas de crear un archivo de tipo EXE. Una es usando un compilador que pueda crear este tipo de archivos (lo cual no significa específicamente que tiene que ser para la plataforma donde se ejecutará, véase Compilador cruzado). La otra forma es *ensamblando* un código fuente del lenguaje Assembler y luego *enlazando* el código objeto resultante de la tarea anterior.

Fuera del ámbito de la *programación* también existen programas que generan ejecutables EXE para tareas específicas. Ejemplos de estos son el compresor WinZip, el Microsoft Powerpoint y el Adobe Flash.

Para revisar el funcionamiento de un programa se realiza la tarea llamada depuración (en inglés: *debugging*).

A continuación se detallan la realización estas tareas para programas de tipo EXE.

**Compilado, Ensamblado y enlazado**

Compilación es el proceso por el cual se traduce un conjunto de llamado código fuente a código objeto. Normalmente se utiliza el término *compilado* para referirse a la tarea de compilación en conjunto al proceso de enlazado, ya que la mayoría (por no decir todos) de los compiladores realizan por defecto ambas tareas en conjunto, al menos que se especifique que se desea únicamente obtener el código objeto.

A continuación se expone un ejemplo de cómo se compila un programa escrito en el Lenguaje de programación C usando el compilador GCC del proyecto MinGW desde la línea de comandos:

|  |  |
| --- | --- |
| **Archivo: *ejemplo.c*** | **En la línea de comandos** |
| #include *<stdio.h>*  int main(void){  printf("¡Hola, mundo!**\n**");  **return** 0; | C:\*>gcc ejemplo.c -o ejemplo.exe*  C:\*>ejemplo.exe*  ¡Hola, mundo!  C:\*>* |

El término *ensamblado* refiere al proceso de compilación (sin enlazado) de un código fuente específicamente escrito en Assembler. En este caso para poder obtener un ejecutable es necesario realizar el *enlazado* en forma separada (algunos de los compiladores más modernos del lenguaje *Assembler* realizan en ensamblado y enlazado en conjunto).

El terminó enlazado refiere al proceso en el que se convierte el *código objeto* en un archivo ejecutable. A continuación se expone un ejemplo de cómo se compila un programa escrito en el lenguaje Assembler usando el ensamblador TASM y enlazador TLINK desde la línea de comandos:

|  |  |
| --- | --- |
| **Archivo: *ejemplo.asm*** | **En la línea de comandos** |
| .model small  .stack 64  .data  msg db 0ADh,'¡Hola, mundo!',0Dh,0Ah,24h  .code  mov ax,@data  mov ds,ax  mov ah,9  lea dx,msg  int 21h  mov ax,4C00h  int 21h  end | C:\*>tasm ejemplo.asm*  Turbo Assembler Version 4.0  Copyright (c) 1988, 1993 Borland International  Assembling file: ejemplo.asm  Error messages: None  Warning messages: None  Passes: 1  Remaining memory: 408k  C:\*>tlink ejemplo.obj*  Turbo Link Version 6.00  Copyright (c) 1992, 1993 Borland International  C:\*>ejemplo.exe*  ¡Hola, mundo! |

**Depuración**

Diferencias entre .COM y .EXE

La principal diferencia es que el formato EXE no restringe el tamaño del ejecutable, en cambio el COM está restringido como máximo a ocupar un solo segmento de memoria (un espacio de memoria de 64 kB); además hay que incluir los 256 bytes que ocupa el PSP. Este límite fue dado en una época en la que esta cantidad de espacio era más que suficiente para poder realizar cualquier clase de tarea, pero con el paso del tiempo este límite terminó siendo absurdo (hoy en día gran parte de los programas de computadoras ocupan miles de veces más espacio). De todas formas es posible hacer que un programa COM pueda extender este límite usando el enunciado *SEGMENT AT*.

Un programa COM es prácticamente una *imagen* en memoria del archivo en disco, menos por el PSP que es creado durante el proceso de inicialización, y es por esta razón que el programa se inicia a partir del offset 0x100 (en *assembler* se utiliza el operador **ORG**). En cambio el formato EXE reorganiza sus segmentos en la memoria dando un tamaño adecuado (que es posible y necesario que el programador lo determine, dependiendo del lenguaje en el cual fue escrito), es por esta razón que dentro de la cabecera de un archivo EXE hay cierto código que realiza esta tarea.

El utilitario EXE2COM cambiaba de un formato a otro.

Emulación

Las razones por la cual se utiliza un programa de emulación para correr una aplicación con formato EXE son variadas, dependiendo en mayor medida desde que plataforma se intenta emular.

**En Windows**

Desde el surgimiento, en el mercado de usuarios domésticos, de Windows XP fue común el uso de emuladores para DOS en el ámbito de los juegos de vídeo, debido a que Windows XP no utilizaba MS-DOS (como los sistemas basados en Windows NT) lo que trajo problemas de compatibilidad de los viejos juegos con el nuevo sistema. Otra de las causas para el uso de esto es que algunos juegos antiguos usaban temporizadores haciendo cierta cantidad de ciclos de operaciones *NOP* y con la aparición de procesadores más rápidos esto se hizo inútil ya que la velocidad en que terminaba el ciclo era mucho más rápido de lo que se tenía planeado. Uno de los emuladores para Windows de DOS más importante es DOSBox.  
El Service Pack 2 de Windows XP incluyó un *emulador* para aplicación antigua, pero este no funciona del todo bien. [*Cita requerida*] En Windows 7 se incluye un modo de compatibilidad para versiones anteriores del sistema que usaban NT o DOS

**En Mac OS**[editar]

La emulación de un ejecutable EXE en Mac OS puede requerir la emulación de la arquitectura de PC debido a que las antiguas computadoras Macintosh usaban procesadores RISC de Motorola y PowerPC de IBM, Motorola y Apple con un conjunto de instrucciones distinto. Uno de los emuladores más importantes en este campo es el QEMU, un software gratuito, de código abierto y multiplataforma.  
Además de tener un emulador de PC es necesario que alguna aplicación pueda interpretar, cargar en memoria y ejecutar el programa EXE que se desea correr; en la mayoría de los casos el programa EXE realizará alguna llamada al sistema durante su ejecución. Todas estas tareas pueden realizarse de dos formas: teniendo una partición virtual con algún sistema Windows e iniciarlo desde Mac OS usando algún software como el **QEMU** o el Virtual PC for Mac; la segunda forma es usando el software llamado Darwine (compatible con los Macintosh con procesadores Motorola y los Intel) que permite realizar todas estas tareas sin necesidad de tener algún sistema Windows.

**En Linux**[editar]

Linux no soporta el formato EXE, por lo que se necesita alguna aplicación en espacio de usuario para poder correr un programa de este tipo. Uno de los más populares es WINE, un software que permite correr un ejecutable EXE Win32 sin necesidad de virtualizar. También es posible usar emuladores de DOS como DOSBox o DOSEmu.

**Microsoft SQL Server 2005**

Este programa de Microsoft nos permite gestionar nuestras bases de datos a través de unas funciones avanzadas que nos garantizan la máxima seguridad y confianza. Microsoft SQL Server 2005 permite trabajar en modo cliente-servidor y ofrece una gran estabilidad en el soporte de los datos almacenados. También ofrece protección de datos y rendimiento para clientes de aplicaciones incrustadas, aplicaciones Web ligeras y almacenes de datos locales.

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

Dentro de los competidores más destacados de SQL Server están: Oracle, MariaDB, MySQL, PostgreSQL. SQL Server solo está disponible para sistemas operativos Windows de Microsoft.

Puede ser configurado para utilizar varias instancias en el mismo servidor físico, la primera instalación lleva generalmente el nombre del servidor, y las siguientes - nombres específicos (con un guion invertido entre el nombre del servidor y el nombre de la instalación).

Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Historia de versiones** | | | |
| **Versión** | **Año** | **Nombre de la versión** | **Nombre clave** |
| 1.0 (OS/2) | 1989 | SQL Server 1-0 | SQL |
| 4.21 (WinNT) | 1993 | SQL Server 4.21 | SEQUEL |
| 6.0 | 1995 | SQL Server 6.0 | SQL95 |
| 6.5 | 1996 | SQL Server 6.5 | Hydra |
| 7.0 | 1998 | SQL Server 7.03 | Sphinx |
| - | 1999 | SQL Server 7.0 OLAP Tools | Plato |
| 8.0 | 2000 | SQL Server 20004 |  |
| 8.0 | 2003 | SQL Server 2000 64-bit Edition | Liberty |
| 9.0 | 2005 | SQL Server 20055 | Yukon |
| 10.0 | 2008 | SQL Server 20086 | Katmai |
| 10.25 | 2010 | SQL Azure DB | CloudDatabase |
| 10.50 | 2010 | SQL Server 2008 R27 | Kilimanjaro |
| 11.0 | 2012 | SQL Server 20128 | Denali |
| 12.0 | 2014 | SQL Server 20141 | SQL14 (antes Hekaton) |

El código fuente original de SQL Server que fue utilizado en las versiones previas a la versión 7.0 habría sido comprado de Sybase, pero fue actualizado en las versiones 7.0 y 2000, y reescrito en la versión 2005. Generalmente, cada 2-3 años, una nueva versión es lanzada y, entre estos lanzamientos, se proponen service packes con mejoras y correcciones de bugs, y hotfixes por problemas urgentes en el sistema de seguridad o bugs críticos.

Características

* Soporte de transacciones.
* Soporta procedimientos almacenados.
* Incluye también un entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
* Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
* Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye en forma *gratuita*.

Es común desarrollar proyectos completos empleando *Microsoft SQL Server* y Microsoft Access a través de los llamados **ADP** (Access Data Project). De esta forma se completa la base de datos (*Microsoft SQL Server*), con el entorno de desarrollo (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.

En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD, osql, o PowerShell.

Para el desarrollo de aplicaciones más complejas (tres o más capas), *Microsoft SQL Server* incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos.

El tipo NUMERIC fue mejorado para ser usado como identificador de columna a partir de la versión 2008 R2.

Programación

**T-SQL**

T-SQL (Transact-SQL) es el principal medio de interacción con el Servidor, el cual permite realizar las operaciones claves en SQL Server, incluyendo la creación y modificación de esquemas de base de datos, inserción y modificación de datos en la base de datos, así como la administración del servidor como tal. Esto se realiza mediante el envío de sentencias en T-SQL y declaraciones que son procesadas por el servidor y los resultados (o errores) regresan a la aplicación cliente.

**Cliente Nativo de SQL**

Cliente Nativo de SQL, es la biblioteca de acceso a datos para los clientes de Microsoft SQL Server versión 2005 en adelante. Implementa de forma nativa soporte para las características de SQL Server, incluyendo la ejecución de la secuencia de datos tabular, soporte para bases de datos en espejo de SQL Server, soporte completo para todos los tipos de datos compatibles con SQL Server, conjuntos de operaciones asíncronas, las notificaciones de consulta, soporte para cifrado, así como recibir varios conjuntos de resultados en una sola sesión de base de datos. Cliente Nativo de SQL se utiliza como extensión de SQL Server plug-ins para otras tecnologías de acceso de datos, incluyendo ADO u OLE DB. Cliente Nativo de SQL puede también usarse directamente, pasando por alto las capas de acceso de datos.

Ediciones y servicios

Cada versión de SQL Server posee distintas versiones con distintos precios (para cada versión) que dependen también en la configuración física del servidor. 9 A continuación se presentan las versiones principales:

**Enterprise**

Contempla todas las características (deshabilitadas en otras ediciones).Es el tipo de versión con más privilegios existente en el mercado.

**Develope**

Una edición con las mismas características que la Enterprise, con el fin de ser instalada solamente en ambiente de desarrollo y no en producción. Si se desarrolla para una edición Standard hay que tener en cuenta las características deshabilitadas para esta versión.

**Standard**

Una versión limitada según la configuración del servidor y sus características, diseñada para servidores inferiores.

Por ejemplo: en la versión 2012, la edición Enterprise soporta un número ilimitado de procesadores, y la agregación de memoria y CPUs en caliente sin la interrupción del servicio o del servidor; mientras la edición Standard está limitada a 16 procesadores y no soporta la "agregación en caliente".

**Express**

Una versión gratuita que posibilita la creación de bases de datos limitadas con características básicas, con el fin de apoyar aplicaciones que necesiten una solución simple para almacenamiento de una cantidad limitada de datos, o usuarios que sus recursos y necesidades son limitados.

En la versión 2012, esta edición puede utilizar un máximo de 1 GB de memoria, y almacenar no más de 10GB, funciona en servidores con un número máximo de cuatro procesadores. Estas limitaciones se mantienen en la versión 2014 (4 cores, 1GB ram, y 10Gb por base de datos).

**SQL Azure**

Es una versión de SQL Server en la nube, que permite pagar mensualmente por el servicio sin la necesidad de mantener un servidor físico (On Premise). La empresa paga solo por el servicio, y el servicio es manejado a través de torres de servidores en distintos lugares en el mundo.

Con SQL Azure no es necesario instalar, mantener o actualizar un servidor físico; a pesar que este servicio depende de aspectos relacionados a problemas de seguridad con respecto a su presencia fuera de la empresa y a la disponibilidad de conexión a Internet.

Durante un tiempo, el servicio fue ampliado con la opción de crear un servidor virtual por la red, e instalar SQL Server tanto como uno de los servicios competidores, y manejar el servidor virtual como si fuera un servidor físico local (aunque físicamente no está accesible); y se puede diferenciar entre la opción original que esta denominada PAAS (*Platform as a Service*: *El Plataforma como un Servicio*) y la nueva opción de los servidores virtuales denominada IAAS (Infrastructure as a Service, Infraestructura como un Servicio).

Este servicio esta otorgado por Microsoft desde 2009 y se une a servicios similares de empresas de third-party.

Interfaz de usuario

SQL Server proporciona un interfaz que han cambiado durante los años, de los cuales los más conocidos son el interfaz gráficos que están utilizados como herramienta de desarrollo estándar a los desarrolladores y administradores.

La interfaz gráfica hasta 2005 incluyó el Enterprise Manager con una vista de árbol de los distintos objetos y con la capacidad de manejarlos; y el Query analyzer como interfaz textual para ejecutar comandos de TSQL.

En la versión 2005 las dos herramientas se unificaron a una –el SQL Server Management Studio (SSMS), y a partir de 2008 fue incluida la opción de trabajar con el Visual Studio– la interfaz estándar de desarrollo de Microsoft (a los distintos lenguajes, BI, etc.). Otro interfaz opcional es la utilización de Línea de comandos, con herramientas como SQLCmd, ISQL, OSQL que posibilita la ejecución de scripts y procesamiento por lotes. Desde 2008 se puede desarrollar con SQLCmd (SQL Command) a través del SSMS sin interconectarse al interfaz textual de Windows. Otra opción en el ámbito de scripts es la utilización del lenguaje de scripts Powershell de Microsoft.

Aparte de los intefazes estándares de SQL Server, se puede ejecutar comandos de TSQL con herramientas de conexión como ODBC y OLE-DB.10

Servicios

A contrario de sistemas de bases de datos como Microsoft Access que son "pasivas" y contienen un archivo a cual hay que conectar y la ejecución de los comandos se lleva a cabo en el cliente (la computadora de usuario), en SQL Server hay número de servicios, software que están ejecutadas en la memoria del servidor por parte del sistema, y por lo tanto aprovechan las capacidades del servidor que es más potente que los clientes, previenen congestión en la red, y pueden programar tareas que corran aunque el cliente no está conectado.

Los servicios principales:

* **SQL Server** - El "motor" del sistema
* **SQL Agent** - Ejecución de tareas (Jobs, scripts programados) y envió de advertencias en caso de carga pesada e irregulares en el sistema
* **Full-Text Filter Daemon Launcher** - La utilización en los indexes especiales del "Full text search" por búsqueda textual avanzada
* **SQL Browser** - El "oyente" dedicado a comandos enviados y redirigirlos a su destino
* **SSIS Server** - La operación del SSIS (la herramienta de ETL)
* **SSAS Server** - La operación del SSAS (la herramienta de OLAP)
* **SSRS Server** - La operación del SSRS (la herramienta de informes)

Capacidades y herramientas básicas

**Bases de datos**

En cada instalación de SQL Server hay 4 bases de datos de sistema, y la capacidad de crear nuevas bases de datos por el usuario, en los cuales los datos están almacenados en tablas.

Estas bases de datos, creadas por parte de los usuarios, incluyen básicamente un archivo de datos (con el sufijo mdf) con las tablas y los distintos objetos a nivel de la base de datos; y un archivo de registro (con el sufijo ldf) con las transacciones abiertas, y transacciones cerradas, Sujeto al modelo de recuperación seleccionado (se puede acumular en el archivo de registro todos los cambios en la base de datos desde el último respaldo). Se puede crear un conjunto de archivos de datos además del principal (con el sufijo ndf) por consideraciones de eficiencia, partición de carga de trabajo entre los discos rígidos, etc.

Las bases de datos del sistema:

* **master** - Todos los procedimientos, funciones y tablas del sistema que están utilizadas por parte de todas las bases de datos y que están instaladas automáticamente, tanto como las que han sido creado por parte de los administradores del sistema. Además, todas las definiciones en respecto a la seguridad a nivel del servidor, están almacenadas en esta base de datos.
* **msdb** - Almacenamiento de las tareas del agente, los códigos de CLR combinados en el sistema, los paquetes de SSIS, y otros más.
* **model** - El molde de las bases de datos. Cada nueva base de datos se crea como una copia de esta base de datos, menos que algo más estaba definido explícitamente.
* **tempdb** - Base de datos temporal que se crea de nuevo cada vez que el servicio reinicia. Se utiliza para almacenar tablas temporales creadas por parte de los usuarios o el sistema (por ejemplo en ordenaciones complejos).

**Tablas fijas y temporales**

Desde la perspectiva lógica, los datos almacenados en las bases de datos en tablas, que mediante ellas se implementan la teoría de las bases de datos relacionales. La tabla se divide en filas y columnas (A veces se les conoce como registros y campos). Las tablas pueden ser fijas o temporales, mientras que en el segundo caso existen físicamente en la base de datos tempdb, y se borran automáticamente en caso de desconexión de la sesión o de la conexión al servidor, depende en el tipo de la tabla temporal.

Desde la perspectiva física, el sistema divide los archivos de la base datos en Extents de 64 KB, y cada cual a ocho páginas de 8 KB. Generalmente, cada Extent se asigna a una tabla o un índice, menos las tablas pequeñas; y cada página se asigna siempre a una tabla específica. El sistema es responsable del aumento de los archivos, de acuerdo con los ajustes del usuario, y de asignar Extents y páginas a las tablas.

A las tablas se puede crear índices. Los índices se almacenan junto a la tabla (Non Clustered Index) o son la tabla en sí (Clustered Index). Los índices asisten en la búsqueda de datos en las tablas (como los ficheros en las librerías), en ordenarlas, y la definición de claves primarias.

Entre las tablas se puede crear una relación de uno a muchos.

Aparte de las tablas de los usuarios, hay tablas que almacenan meta data: datos sobre el sistema mismo, los diferentes objetos, los derechos, estadísticas sobre el rendimiento del sistema (DMV), etc.

**Tipos de datos**

Para cada columna en una tabla y a cada variable o parámetro, se define un tipo de datos que sean almacenados en él, entre ellos:

1. **Numeros**: Números enteros y no enteros en distintos tamaños, y en diferentes niveles de precisión; y auto incremento opcional.
2. **Textos**: Cadenas de distintas longitudes, y distintas capacidades de apoyar distintas lenguas.
3. **Fechas**: Fechas en distintos niveles de precisión, desde días completos hasta fracciones menores de un segundo, que apoyan fechas a partir del principio del siglo 20 o del calendario gregoriano, y la capacidad de diferenciar entre distintos usos de horarios.
4. **XML**: Datos textuales (cadenas) que representan conjuntos estándares de datos (estándar SGML).
5. **Datos binarios**: Datos almacenados como datos binarios (bits y bytes), que posibilitan el almacenamiento de archivos gráficos, etc.
6. **Geography**: Representación estándar de información geográfica, tales como estados, zonas geográficas, localidades; y las cálculos como distancias.
7. **Geometry**: Representación estándar de puntas, líneas, superficies en el plano; y las relaciones entre ellas.
8. **Hierarchid**: Representación estándar de información jerárquica como lista de materiales, relaciones de subordinación entre empleados, etc.

**Vistas**

Las vistas representan generalmente comandos de extracción de datos, que se almacenan sin los datos (que están almacenados en las tablas). Esta opción nos posibilita crear extracciones complejas o estándares, almacenarlas como vistas, y utilizar las vistas sin la necesidad de escribir de nuevo los comandos o mantener los códigos donde ellas aparecen. Adicionalmente, es un medio muy importante para otorgar derechos selectivos de lectura (en caso que queremos posibilitar a un usuario contemplar parcialmente las columnas o las filas de una tabla).

Una vista se puede considerar una tabla virtual o una consulta almacenada. Los datos accesibles a través de una vista no están almacenados en un objeto distinto de la base de datos. Lo que está almacenado en la base de datos es una instrucción SELECT. El resultado de la instrucción SELECT forma la tabla virtual que la vista devuelve. El usuario puede utilizar dicha tabla virtual haciendo referencia al nombre de la vista en instrucciones Transact-SQL, de la misma forma en que se hace referencia a las tablas. Las vistas se utilizan para alguna de estas funciones, o para todas:  
• Restringir el acceso del usuario a filas concretas de una tabla. Por ejemplo, permitir que un empleado sólo vea las filas que guardan su trabajo en una tabla de seguimiento de actividad laboral.  
• Restringir el acceso del usuario a columnas específicas. Por ejemplo, permitir que los empleados que no trabajen en el departamento de nóminas vean las columnas de nombre, oficina, teléfono y departamento de la tabla de empleados, pero no permitir que vean las columnas con los datos de salario u otra información personal.  
• Combinar columnas de varias tablas de forma que parezcan una sola tabla.  
• Agregar información en lugar de presentar los detalles. Por ejemplo, presentar la suma de una columna o el valor máximo o mínimo de una columna.  
  
Las vistas se crean definiendo la instrucción SELECT que recupera los datos presentados por la vista. Las tablas de datos a las que hace referencia la instrucción SELECT se conocen como las tablas base para la vista. Las vistas en todas las versiones de SQL Server son actualizables (pueden ser objetivo de instrucciones UPDATE, DELETE o INSERT) mientras la modificación afecte sólo a una de las tablas base de la vista.

**Procedimientos almacenados**

Los procedimientos son scripts de comandos de TSQL, que pueden ser ejecutados con distintos parámetros. Por ejemplo, procedimiento que obtiene número de año como parámetro, y actualiza una tabla de resumen de ventas, con las ventas de los agentes en el dicho año, basada en la tabla de registro de ventas.

Los procedimientos almacenados pueden facilitar en gran medida la administración de la base de datos y la visualización de información sobre dicha base de datos y sus usuarios. Los procedimientos almacenados son una colección precompilada

de instrucciones SQL e instrucciones de control de flujo opcionales almacenadas bajo un solo nombre y procesadas como una unidad. Los procedimientos almacenados se guardan en una base de datos; se pueden ejecutar desde una aplicación y permiten variables declaradas por el usuario, ejecución condicional y otras funciones eficaces de programación. Los procedimientos almacenados pueden contener flujo de programas, lógica y consultas a la base de datos. Pueden aceptar parámetros, proporcionar resultados de parámetros, devolver conjuntos de resultados individuales o múltiples y devolver valores.

Las ventajas de utilizar procedimientos almacenados en SQL Server en vez de programas Transact-SQL almacenados localmente en equipos clientes consisten en que:  
• Permiten una programación modular.  
Puede crear el procedimiento una vez, almacenarlo en la base de datos, y llamarlo desde el programa el número de veces que desee. Un especialista en programación de bases de datos puede crear procedimientos almacenados, que luego será posible modificar independientemente del código fuente del programa. Facilitan el mantenimiento.  
• Permiten una ejecución más rápida.  
En situaciones en las que se necesita una gran cantidad de código Transact-SQL, o si las operaciones se realizan varias veces, los procedimientos almacenados pueden ser más rápidos que los lotes de código Transact-SQL. Los procedimientos son analizados y optimizados en el momento de su creación, y es posible utilizar una versión del procedimiento que se encuentra en la memoria después de que se ejecute por primera vez. Las instrucciones de Transact-SQL que se envían varias veces desde el cliente cada vez que deben ejecutarse tienen que ser compiladas y optimizadas siempre que SQL Server las ejecuta.  
• Pueden reducir el tráfico de red.  
Una operación que necesite centenares de líneas de código Transact-SQL puede realizarse mediante una sola instrucción que ejecute el código en un procedimiento, en vez de enviar cientos de líneas de código por la red.  
• Pueden utilizarse como mecanismo de seguridad.  
Es posible conceder permisos a los usuarios para ejecutar un procedimiento almacenado, incluso si no cuentan con permiso para ejecutar directamente las instrucciones del procedimiento.

**Funciones definidas por el usuario**

Las funciones son un objeto que combina algunas capacidades de las vistas, con otras de los procedimientos. Como las vistas, pueden extraer datos y ejecutar cálculos, y devuelven un resultado al usuario o al programa que les ejecuto. Tanto como los procedimientos, incluyen códigos de TSQL, y pueden ser ejecutados con parámetros.

Las funciones devuelven un valor o un conjunto de valores.

Las funciones definidas por el usuario se crean con la instrucción CREATE FUNCTION, se modifican con la instrucción ALTER FUNCTION y se quitan con la instrucción DROP FUNCTION. Todos los nombres de funciones completos (database\_name.owner\_name.function\_name) definidos por el usuario deben ser únicos. Para crear, modificar o quitar funciones definidas por el usuario, debe tener permisos de CREATE FUNCTION. Los usuarios distintos del propietario deben tener permiso EXECUTE para una función, y solo así podrán utilizarla en una instrucción de Transact-SQL. Para crear o modificar tablas con referencias a funciones definidas por el usuario en la restricción CHECK, la cláusula DEFAULT o la definición de una columna calculada, también debe tener permiso REFERENCES para las funciones. Los errores de Transact-SQL que producen la cancelación de una instrucción y continúan con la siguiente instrucción del módulo, como desencadenadores o procedimientos almacenados, se tratan de forma distinta dentro de una función. En las funciones, estos errores hacen que se detenga la ejecución de la función. Esto hace que se cancele la función que invocó la instrucción. Una función definida por el usuario no tiene ninguno o tiene varios parámetros de entrada y devuelve un valor escalar o una tabla. Una función puede tener un máximo de 1024 parámetros de entrada. Cuando un parámetro de la función toma un valor predeterminado, debe especificarse la palabra clave DEFAULT al llamar a la función para poder obtener el valor predeterminado. Este comportamiento es diferente del de los parámetros con valores predeterminados de los procedimientos almacenados, para los cuales omitir el parámetro implica especificar el valor predeterminado. Las funciones definidas por el usuario no admiten parámetros de salida.

**Consultas Distribuidas**

Las consultas distribuidas tienen acceso a datos de varios orígenes, que pueden estar almacenados en un equipo o en equipos distintos. Microsoft SQL Server 2000 admite las consultas distribuidas a través de OLE DB Las consultas distribuidas proporcionan a los usuarios de SQL Server acceso a:  
• Datos distribuidos almacenados en múltiples instancias SQL Server.  
• Datos heterogéneos almacenados en varios orígenes de datos relacionales y no relacionales a los que se tiene acceso mediante un proveedor OLE DB.  
Los proveedores OLE DB exponen datos en objetos tabulares llamados conjuntos de filas. En las instrucciones Transact-SQL, SQL Server 2000 permite que se haga referencia a los conjuntos de filas de los proveedores OLE DB como si fueran una tabla de SQL Server. En las instrucciones SELECT, INSERT, UPDATE y DELETE de Transact-SQL, se puede hacer referencia directa a las tablas y vistas de orígenes de datos externos. Puesto que las consultas distribuidas usan OLE DB como interfaz subyacente, éstas tienen acceso a los sistemas DBMS relacionales tradicionales con procesadores de consultas SQL, así como a los datos administrados por orígenes de datos de capacidad y sofisticación diversas. Siempre que el software propietario de los datos los expone en un conjunto de filas tabular a través del proveedor OLE DB, los datos se podrán usar en las consultas distribuidas. Nota: El uso de las consultas distribuidas en SQL Server es similar a la funcionalidad de las tablas vinculadas mediante ODBC, que anteriormente admitía Microsoft Access. Esta funcionalidad se encuentra ahora integrada en SQL Server con OLE DB como interfaz para los datos externos.

**Transacciones**

Una transacción es un conjunto de comandos, que se está ejecutado completamente o no ejecutado en absoluto: todo o nada. Por ejemplo, si una suma de dinero fue trasladada de una cuenta bancaria a otra, y hay que actualizar ambas cuentas sobre el depósito y la retirada; es obligatorio que ambas cuentas se actualizan juntas, o ninguna (en caso que una de las actualizaciones falla); para evitar consecuencias inconsistentes de un depósito sin ninguna retirada, o vice versa. Por lo tanto, una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo. Una unidad lógica de trabajo debe exhibir cuatro propiedades, conocidas como propiedades ACID (atomicidad, coherencia, aislamiento y durabilidad), para ser calificada como transacción:

* Atomicidad

Una transacción debe ser una unidad atómica de trabajo, tanto si se realizan todas sus modificaciones en los datos, como si no se realiza ninguna de ellas.

* Coherencia

Cuando finaliza, una transacción debe dejar todos los datos en un estado coherente. En una base de datos relacional, se deben aplicar todas las reglas a las modificaciones de la transacción para mantener la integridad de todos los datos. Todas las estructuras internas de datos, como índices de árbol B o listas doblemente vinculadas, deben estar correctas al final de la transacción.

* Aislamiento

Las modificaciones realizadas por transacciones simultáneas se deben aislar de las modificaciones llevadas a cabo por otras transacciones simultáneas. Una transacción ve los datos en el estado en que estaban antes de que otra transacción simultánea los modificara o después de que la segunda transacción se haya concluido, pero no ve un estado intermedio. Esto se conoce como seriabilidad debido a que su resultado es la capacidad de volver a cargar los datos iniciales y reproducir una serie de transacciones para finalizar con los datos en el mismo estado en que estaban después de realizar las transacciones originales.

* Durabilidad

Una vez concluida una transacción, sus efectos son permanentes en el sistema. Las modificaciones persisten aún en el caso de producirse un error del sistema.

SQL Server tiene una capacidad limitada de anidar transacciones.

**El optimizador**

El optimizador es una parte del software que "toma la decisión" de cómo cada comando se ejecutará, tanto que la ejecución será lo más eficiente, o por lo menos bastante eficiente (es decir, bastante eficiente para evitar seguir buscando otra solución, que aún que sea más eficiente, el precio de la búsqueda adicional "costará" más que el ahorro de recursos).

SQL es un lenguaje declarativo, en el cual el desarrollador declara que quiere extraer o actualizar sin la necesidad de indicar cómo (a contrario de los lenguajes imperativos, y por lo tanto el optimizador juega un papel protagónico, que de acuerdo con las estadísticas que el sistema almacena sobre las distribuciones de los datos en las tablas, los indexes, y reglas internas; toma la decisión adecuada.

Privilegios y seguridad de datos

Para conectarse al SQL Server, se necesita un Login (usuario a nivel del servidor). Cuando la política de seguridad se define como Windows Authentication y el servidor se combina con las definiciones del Domain, los Logins se definen en el Active Directory. Cuando la definición es SQL Server Authentication los logins (usuario y contraseña) se definen en el SQL Server mismo. Consecuentemente, en el primer caso hay que identificarse con nombre y contraseña solamente al conectarse a la red, y luego se conecta automáticamente a todos los servidores que son Windows Authentication (con el Login global); y en el segundo caso hay que identificarse al conectarse a cada servidor de SQL Server Authentication (cada vez con un Login local).

A nivel de la base de datos, el usuario se identifica como un User que está relacionado generalmente al Login (que es a nivel del servidor), y los privilegios al User existen solamente en el ámbito de la base de datos (además a los privilegios al Login). Para otorgar derechos generales puede asistirse con listas de Server Roles (roles a nivel del servidor) o Database Roles (roles a nivel de la base de datos específica), cada cual con privilegios específicos a un rol específico; y cada usuario asociado con uno de estos Roles obtiene los privilegios asociados con él. Además, el administrador puede otorgar derechos specificos, y crear otros Database Roles (no se puede crear Server Roles).

Los privilegios a nivel del servidor incluyen la capacidad de crear bases de datos, utilizar las tareas (Jobs), crear respaldos de bases de datos y restaurarlos, modificar las definiciones del servidor, etc. Los privilegios a nivel de la base de datos posibilitan extraer y actualizar datos, crear objetos como procedimientos y tablas, utilizar dichos objetos, etc. Como regla general se puede otorgar derechos (Grant), revocar privilegios existentes (Revoke), y denegar privilegios aún no existen (Deny).

Otras herramientas de servicio

Adicionalmente a sus capacidades elementarías como herramienta de gestión de bases de datos relacionales (crear tablas, definir las relaciones entre ellas, gestión de transacciones, crear índices etc, SQL Server apoya una lista que aumenta de otras herramientas de servicio; aparte de herramientas de terceros que cambian o que completan lo que existe.

**Respaldos y recuperaciones**

Aparte de soluciones de alternativas a nivel del sistema operativo (respaldo de los archivos de la base de datos), hay una herramienta integrada en el SQL Server que posibilita un respaldo completo o diferencial, de acuerdo con el modelo de recuperación (Recovery Model) predefinido a la base de datos; y una recuperación completa o a un punto de tiempo. Aparte de un respaldo de la base de datos se puede respaldarlos a través de un guion (con o sin los datos). A partir de 2008, se puede comprimir los archivos de respaldo.

**Compresión**

A partir de 2008 se añadió la opción de compresión que posibilita comprimir el tamaño físico de las tablas y los índices, y utilizar más eficientemente el volumen de los discos rígidos y reducir operaciones de IO (lo que aumenta la carga del CPU). Como ya ha sido mencionado, se puede comprimir también los archivos de respaldo.

**Replicación, alta disponibilidad, y recuperación de desastres**

Algunas herramientas posibilitan crear replicas parciales o completos de las bases de datos, mejorar la disponibilidad, y recuperar de desastres; aparte de la opción de respaldar y recuperar las bases de datos, una opción que se supone que es muy lenta.

Las copias creadas por estas herramientas, pueden ser utilizadas como un respaldo disponible inmediatamente en caso de que el sistema falla, tanto como una réplica en un subsistema, independiente del sistema de origen, para equilibrio de carga; y que las actualizaciones se ejecuten en la base de datos, y las recuperaciones para los reportajes se ejecuten en la réplica.

Entre dichas herramientas se puede mencionar la replicación que posibilita crear una réplica sincronizada de la base de datos, el Mirroring que ejecuta en el servidor de espejo cada actualización que se ejecuta en el servidor de origen, Log Shipping que posibilita almacenar una copia sincronizada a través de archivos de registro (Log) con todas las actualizaciones en el servidor de origen, y las herramientas de alta disponibilidad y recuperación de desastres (HADR, desde 2012) que solucionan problemas de Mirroring en relación con el tiempo de reacción de los fallos técnicos y la disponibilidad de las copias en tiempos de paz.

**El agente y la programación de tareas**

El agente es el servicio encargado de la programación de tareas, y se encarga de ejecutarlas independientemente. Generalmente el ejecuta tareas de mantenimiento, tareas complejas de ETL, respaldos, etc.

**Mantenimiento**

Con el fin de mejorar el rendimiento del sistema hay que mantener las estadísticas, utilizadas por el optimizador, organizar los archivos físicos, etc; y se utilizan herramientas dedicadas para estos propósitos, que se ejecutan periódicamente por tareas programadas, y de una manera coordinada con las tareas de ETL y de respaldo.

**Service Broker**

Una tecnología que fue agregada en 2005 que implementa arquitectura orientada a servicios, y que posibilita ejecuciones asíncronos: primero que nada para enviar mensajes entre distintas aplicaciones que se ejecutan simultáneamente, pero también para ejecutar procedimientos asíncronamente, en la manera de dispara y olvida - un procedimiento que se ejecuta en una sesión diferente de la sesión que lo inicio, y ambos procedimientos siguen ejecutado independientemente uno del otro.

**Enviar Correos Electrónicos**

SQL Server tiene la capacidad de enviar correos electrónicos a través del código. Esta capacidad se utiliza generalmente para enviar alertas sobre problemas en el sistema (por ejemplo si el CPU aumento a un umbral predefinido o si el proceso ETL falló), pero también cuando el proceso termino exitosamente.

**Full Text Search (Búsqueda de Textos completos)**

Una herramienta que posibilita indexar columnas textuales como textos y no solo como cadenas; y ejecutar búsquedas complejas dependientes en el sentido del texto y en el idioma. Por ejemplo, buscamos un verbo, y queremos obtener todas las ocurrencias de sus conjugaciones.

**Rastrear**

Estas herramientas incluyen el Trace que posibilita rastrear actividades con el fin de mantener cargas y fallos, y seguridad de datos (recuperación no permitida de datos), el Profiler que posibilita rastrear los commandos que se ejecutan y los eventos que se ocurren en el servidor, y el Extended Events que fue agregado en 2008 y cambia el profiler gracias a su baja signatura (la consumición de recursos y la influencia sobre el rendimiento del servidor).

Aparte de estos, se puede utilizar 2 tipos de Trigers (disparadores) para rastrear los cambios y las actividades: DML Trigers pre definidos sobre las tablas y las vistas y que se inician por instrucciones de actualización de datos (Select / Update / Delete), y DDL Triggers que se inician por cambios en los objetos mismos (y no en los datos), en el nivel de la base de datos o del servidor.

**Combinación de CLR**

A partir de 2005 se puede combinar fácilmente en SQL Server procedimientos, funciones, y funciones de agregado desarrolladas en CLR. Hay que desarrollar el código en una de las herramientas de desarrollo de .NET, crear un archivo DLL, y combinarlo en el sistema. La ventaja de de esta tecnología es sus capacidades en problemas que no son exclusivamente de bases de datos (manipulación de datos), e incluyen cálculos complejos o manipulaciones textuales de cadenas.

**Herramientas de Inteligencia empresarial**

Una instalación típica incluye también las herramientas de BI (Inteligencia empresarial):

**SSIS (SQL Server Integration Services)**

Una herramienta de ETL que posibilita la extracción de datos de distintos orígenes (no solo SQL Server), la transformación de dichos datos, y la carga (generalmente pero no obligatoriamente a almacén de datos).

**SSAS (SQL Server Analysis Services)**

Una herramienta para crear Bases de Datos Multidimensionales (no relacionales), que se puede explorar mediante extracciones de datos en distintos niveles de agrupación, profundización (Drill Down) de una suma a sus detalles, y utilización de MDX (un lenguaje parecido a SQL, adaptado a bases de datos multidimensionales).

**SSRS (SQL Server Reporting Services)**

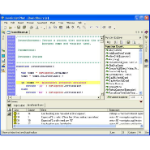
Una herramienta para crear y dar formato a informes, otorgar derechos de contemplación en ellos, y su distribución. Se puede contemplarlos con un Navegador web, y se puede exportarlos a archivos de Excel, PDF, etc. los datos se extraen generalmente del almacén de datos o del OLAP.

Desventajas

* En versiones de 32 bits, SQL Server usa *Address Windowing Extension* (AWE) para hacer el direccionamiento por encima de 4 GB. Esto le impide usar la administración dinámica de memoria, y sólo le permite alojar un máximo de 64 GB de memoria compartida. Esta limitación es exclusiva de sistemas operativos 32 bits; en sistemas operativos 64 bits, la memoria máxima que se puede direccionar en Edición Estándar es 64 Gb y en Edición Enterprise 4Tb

**Microsoft Access**

Este gestor de bases de datos relacionales es uno de los más conocidos y completos, y se encuentra dentro del paquete Microsoft Office. Si queremos empezar rápidamente podemos escoger una de las plantillas predefinidas y editarla a nuestro gusto, algo que en muchos casos ayuda a facilitar el trabajo. Microsoft Access admite macros, código Visual Basic y consultas directas por medio de lenguaje SQL para que no nos falte de nada.

** Javascript Plus**

Se le considera como uno de los mejores editores de archivos javascript. Gracias a su tecnología nos permite editar en un solo entorno archivos javascript, CSS, HTML y XHTML. Posee una interfaz clara y limpia, un sistema intuitivo y una lista de tutoriales de aprendizaje para utilizar todas sus herramientas. Javascript Plus ofrece entre sus características las funciones de autocomplemento, búsqueda y exploración, edición de códigos preexistentes, vista previa, gestión de FTP... y también permite conectarse a una base de datos y exportar los resultados a HTML, Excel, o documentos XML.

**JavaScript** (abreviado comúnmente **JS**) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4 aunque existe una forma de JavaScript del lado del servidor(Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde el 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de javascript. Los navegadores más antiguos soportan por lo menos ECMAScript 3. La sexta edición se liberó en julio del 2015.5

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Desde el lanzamiento en junio de 1997 del estándar ECMAScript 1, han existido las versiones 2, 3 y 5, que es la más usada actualmente (la 4 se abandonó6 ). En junio de 2015 se cerró y publicó la versión ECMAScript 6

Historia

**Nacimiento de JavaScript**

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de *Mocha*, el cual fue renombrado posteriormente a *LiveScript*, para finalmente quedar como JavaScript. El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó compatibilidad con la tecnología Java en su navegador web Netscape Navigator en la versión 2.002 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha caracterizado por muchos como una estrategia de mercadotecnia de Netscape para obtener prestigio e innovar en el ámbito de los nuevos lenguajes de programación web.8 9

«JAVASCRIPT» es una marca registrada de Oracle Corporation.10 Es usada con licencia por los productos creados por Netscape Communications y entidades actuales como la Fundación Mozilla.11 12

Microsoft dio como nombre a su dialecto de JavaScript «JScript», para evitar problemas relacionadas con la marca. JScript fue adoptado en la versión 3.0 de Internet Explorer, liberado en agosto de 1996, e incluyó compatibilidad con el Efecto 2000 con las funciones de fecha, una diferencia de los que se basaban en ese momento. Los dialectos pueden parecer tan similares que los términos «JavaScript» y «JScript» a menudo se utilizan indistintamente, pero la especificación de JScript es incompatible con la de ECMA en muchos aspectos.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, o Modelo de Objetos del Documento en español), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera la versión 7, Mozilla Application Suite y Mozilla Firefox desde su primera versión.[*cita requerida*]

En 1997 los autores propusieron13 JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers 'Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también como un estándar ISO.

**JavaScript en el lado servidor**

Netscape introdujo una implementación de script del lado del servidor con Netscape Enterprise Server, lanzada en diciembre de 1994 (poco después del lanzamiento de JavaScript para navegadores web).14 15 A partir de mediados de la década de los 2000, ha habido una proliferación de implementaciones de JavaScript para el lado servidor. Node.js es uno de los notables ejemplos de JavaScript en el lado del servidor, siendo usado en proyectos importantes.16 17

**Desarrollos posteriores**

JavaScript se ha convertido en uno de los lenguajes de programación más populares en internet. Al principio, sin embargo, muchos desarrolladores renegaban del lenguaje porque el público al que va dirigido lo formaban publicadores de artículos y demás aficionados, entre otras razones.18 La llegada de Ajax devolvió JavaScript a la fama y atrajo la atención de muchos otros programadores. Como resultado de esto hubo una proliferación de un conjunto de frameworks y librerías de ámbito general, mejorando las prácticas de programación con JavaScript, y aumentado el uso de JavaScript fuera de los navegadores web, como se ha visto con la proliferación de entornos JavaScript del lado del servidor. En enero de 2009, el proyecto CommonJS fue inaugurado con el objetivo de especificar una librería para uso de tareas comunes principalmente para el desarrollo fuera del navegador web.19

En junio de 2015 se cerró y publicó el estándar ECMAScript 620 21 con un soporte irregular entre navegadores22 y que dota a JavaScript de características avanzadas que se echaban de menos y que son de uso habitual en otros lenguajes como, por ejemplo, módulos para organización del código, verdaderas clases para programación orientada a objetos, expresiones de flecha, iteradores, generadores o promesas para programación asíncrona.

La versión 7 de ECMAScript se conoce como ECMAScript 201623 , y es la última versión disponible, publicada en junio de 2016. Se trata de la primera versión para la que se usa un nuevo procedimiento de publicación anual y un proceso de desarrollo abierto24 .

Características

Las siguientes características son comunes a todas las implementaciones que se ajustan al estándar ECMAScript, a menos que especifique explícitamente en caso contrario.

**Imperativo y estructurado**

JavaScript es compatible con gran parte de la estructura de programación de C (por ejemplo, sentencias if, bucles for, sentencias switch, etc.). Con una salvedad, en parte: en C, el ámbito de las variables alcanza al bloque en el cual fueron definidas; sin embargo JavaScript no es compatible con esto, puesto que el ámbito de las variables es el de la función en la cual fueron declaradas. Esto cambia con la versión de ECMAScript 2015, ya que añade compatibilidad con block scoping por medio de la palabra clave let. Como en C, JavaScript hace distinción entre expresiones y sentencias. Una diferencia sintáctica con respecto a C es la inserción automática de punto y coma, es decir, en JavaScript los puntos y coma que finalizan una sentencia pueden ser omitidos.25

**Dinámico**

**Tipado dinámico**

Como en la mayoría de lenguajes de scripting, el tipo está asociado al valor, no a la variable. Por ejemplo, una variable x en un momento dado puede estar ligada a un número y más adelante, religada a una cadena. JavaScript es compatible con varias formas de comprobar el tipo de un objeto, incluyendo duck typing.26 Una forma de saberlo es por medio de la palabra clave typeof.

**Objetual**

JavaScript está formado casi en su totalidad por objetos. Los objetos en JavaScript son arrays asociativos, mejorados con la inclusión de prototipos (ver más adelante). Los nombres de las propiedades de los objetos son claves de tipo cadena: obj.x = 10 y obj['x'] = 10 son equivalentes, siendo la notación con punto azúcar sintáctico. Las propiedades y sus valores pueden ser creados, cambiados o eliminados en tiempo de ejecución. La mayoría de propiedades de un objeto (y aquellas que son incluidas por la cadena de la herencia prototípica) pueden ser enumeradas a por medio de la instrucción de bucle for... in. JavaScript tiene un pequeño número de objetos predefinidos como son Function y Date.

**Evaluación en tiempo de ejecución**

JavaScript incluye la función eval que permite evaluar expresiones expresadas como cadenas en tiempo de ejecución. Por ello se recomienda que eval sea utilizado con precaución y que se opte por utilizar la función JSON.parse() en la medida de lo posible, pues resulta mucho más segura.

**Funcional**

**Funciones de primera clase**

A las funciones se les suele llamar ciudadanos de primera clase; son objetos en sí mismos. Como tal, poseen propiedades y métodos, como .call() y .bind().27 Una función anidada es una función definida dentro de otra. Esta es creada cada vez que la función externa es invocada. Además, cada función creada forma una clausura; es el resultado de evaluar un ámbito conteniendo en una o más variables dependientes de otro ámbito externo, incluyendo constantes, variables locales y argumentos de la función externa llamante. El resultado de la evaluación de dicha clausura forma parte del estado interno de cada objeto función, incluso después de que la función exterior concluya su evaluación.28

**Prototípico**

JavaScript usa prototipos en vez de clases para el uso de herencia.29 Es posible llegar a emular muchas de las características que proporcionan las clases en lenguajes orientados a objetos tradicionales por medio de prototipos en JavaScript.

**Funciones como constructores de objetos**

Las funciones también se comportan como constructores. Prefijar una llamada a la función con la palabra clave new crear una nueva instancia de un prototipo, que heredan propiedades y métodos del constructor (incluidas las propiedades del prototipo de Object).31 ECMAScript 5 ofrece el método Object.create, permitiendo la creación explícita de una instancia sin tener que heredar automáticamente del prototipo de Object (en entornos antiguos puede aparecer el prototipo del objeto creado como null).32 La propiedad prototype del constructor determina el objeto usado para el prototipo interno de los nuevos objetos creados. Se pueden añadir nuevos métodos modificando el prototipo del objeto usado como constructor. Constructores predefinidos en JavaScript, como Array u Object, también tienen prototipos que pueden ser modificados. Aunque esto sea posible se considera una mala práctica modificar el prototipo de Object ya que la mayoría de los objetos en Javascript heredan los métodos y propiedades del objeto prototype, objetos los cuales pueden esperar que estos no hayan sido modificados.33

**Otras características**

**Entorno de ejecución**

JavaScript normalmente depende del entorno en el que se ejecute (por ejemplo, en un navegador web) para ofrecer objetos y métodos por los que los scripts pueden interactuar con el "mundo exterior". De hecho, depende del entorno para ser capaz de proporcionar la capacidad de incluir o importar scripts (por ejemplo, en HTML por medio del tag <script>). (Esto no es una característica del lenguaje, pero es común en la mayoría de las implementaciones de JavaScript.)

**Funciones variádicas**

Un número indefinido de parámetros pueden ser pasados a la función. La función puede acceder a ellos a través de los parámetros o también a través del objeto local arguments. Las funciones variádicas también pueden ser creadas usando el método .apply().

**Funciones como métodos**

A diferencia de muchos lenguajes orientados a objetos, no hay distinción entre la definición de función y la definición de método. Más bien, la distinción se produce durante la llamada a la función; una función puede ser llamada como un método. Cuando una función es llamada como un método de un objeto, la palabra clave this, que es una variable local a la función, representa al objeto que invocó dicha función.

**Arrays y la definición literal de objetos**

Al igual que muchos lenguajes de script, arrays y objetos (arrays asociativos en otros idiomas) pueden ser creados con una sintaxis abreviada. De hecho, estos literales forman la base del formato de datos JSON.

**Expresiones regulares**

JavaScript también es compatible con expresiones regulares de una manera similar a Perl, que proporcionan una sintaxis concisa y poderosa para la manipulación de texto que es más sofisticado que las funciones incorporadas a los objetos de tipo string.34

**Extensiones específicas del fabricante**

JavaScript se encuentra oficialmente bajo la organización de Mozilla Foundation, y periódicamente se añaden nuevas características del lenguaje. Sin embargo, sólo algunos motores JavaScript son compatibles con estas características:

* Las propiedades get y set (también compatibles con WebKit, Opera,35 ActionScript y Rhino).36
* Cláusulas catch condicionales.
* Protocolo iterador adoptado de Python.
* Corrutinas también adoptadas de Python.
* Generación de listas y expresiones por comprensión también adoptado de Python.
* Establecer el ámbito a bloque a través de la palabra clave let.
* Desestructuración de arrays y objetos (forma limita de emparejamiento de patrones).
* Expresiones concretas en funciones (function(args) expr).
* ECMAScript para XML (E4X), una extensión que añade compatibilidad nativa XML a ECMAScript.

uso en páginas web

El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el Document Object Model (DOM o Modelo de Objetos del Documento) de la página. Algunos ejemplos sencillos de este uso son:

Cargar nuevo contenido para la página o enviar datos al servidor a través de AJAX sin necesidad de recargar la página (por ejemplo, una red social puede permitir al usuario enviar actualizaciones de estado sin salir de la página).

Animación de los elementos de página, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.

Contenido interactivo, por ejemplo, juegos y reproducción de audio y vídeo.

Validación de los valores de entrada de un formulario web para asegurarse de que son aceptables antes de ser enviado al servidor.

Transmisión de información sobre los hábitos de lectura de los usuarios y las actividades de navegación a varios sitios web. Las páginas Web con frecuencia lo hacen para hacer análisis web, seguimiento de anuncios, la personalización o para otros fines.

Dado que el código JavaScript puede ejecutarse localmente en el navegador del usuario (en lugar de en un servidor remoto), el navegador puede responder a las acciones del usuario con rapidez, haciendo una aplicación más sensible. Por otra parte, el código JavaScript puede detectar acciones de los usuarios que HTML por sí sola no puede, como pulsaciones de teclado. Las aplicaciones como Gmail se aprovechan de esto: la mayor parte de la lógica de la interfaz de usuario está escrita en JavaScript, enviando peticiones al servidor (por ejemplo, el contenido de un mensaje de correo electrónico). La tendencia cada vez mayor por el uso de la programación Ajax explota de manera similar esta técnica.

Un motor de JavaScript (también conocido como intérprete de JavaScript o implementación JavaScript) es un intérprete que interpreta el código fuente de JavaScript y ejecuta la secuencia de comandos en consecuencia. El primer motor de JavaScript fue creado por Brendan Eich en Netscape Communications Corporation, para el navegador web Netscape Navigator. El motor, denominado SpiderMonkey, está implementado en C. Desde entonces, ha sido actualizado (en JavaScript 1.5) para cumplir con el ECMA-262 edición 3. El motor Rhino, creado principalmente por Norris Boyd (antes de Netscape, ahora en Google) es una implementación de JavaScript en Java. Rhino, como SpiderMonkey, es compatible con el ECMA-262 edición 3.

Un navegador web es, con mucho, el entorno de acogida más común para JavaScript. Los navegadores web suelen crear objetos no nativos, dependientes del entorno de ejecución, para representar el Document Object Model (DOM) en JavaScript. El servidor web es otro entorno común de servicios. Un servidor web JavaScript suele exponer sus propios objetos para representar objetos de petición y respuesta HTTP, que un programa JavaScript podría entonces interrogar y manipular para generar dinámicamente páginas web.

Debido a que JavaScript es el único lenguaje por el que los más populares navegadores comparten su apoyo, se ha convertido en un lenguaje al que muchos frameworks en otros lenguajes compilan, a pesar de que JavaScript no fue diseñado para tales propósitos.42 A pesar de las limitaciones de rendimiento inherentes a su naturaleza dinámica, el aumento de la velocidad de los motores de JavaScript ha hecho de este lenguaje un entorno para la compilación sorprendentemente factible.

**MatLab**

En su ficha lo describimos como un software matemático para ingenieros y científicos destinado a resolver cálculos técnicos. Su capacidad para realizar cálculos complejos supera a la de los lenguajes de programación tradicionales, y resulta ideal para tareas avanzadas en biología computacional, análisis financiero, procesamiento de señales e imágenes , sistemas de control... y, gracias a su interfaz de usuario personalizable, se adapta perfectamente a cualquier entorno de trabajo.

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

Historia

Fue creado por el matemático y programador de computadoras Cleve Moler en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales

**Limitaciones y alternativas**

Durante mucho tiempo hubo críticas porque MATLAB es un producto propietario de The Mathworks, y los usuarios están sujetos y bloqueados al vendedor. Recientemente se ha proporcionado una herramienta adicional llamada MATLAB Builder bajo la sección de herramientas "Application Deployment" para utilizar funciones MATLAB como archivos de biblioteca que pueden ser usados con ambientes de construcción de aplicación .NET o Java. Pero la desventaja es que el computador donde la aplicación tiene que ser utilizada necesita MCR(MATLAB Component Runtime) para que los archivos MATLAB funcionen correctamente. MCR se puede distribuir libremente con los archivos de biblioteca generados por el compilador MATLAB.

**Interfaz con otros lenguajes de programación**

MATLAB puede llamar funciones y subrutinas escritas en C o Fortran. Se crea una función envoltorio que permite que sean pasados y devueltos tipos de datos de MATLAB. Los archivos objeto dinámicamente cargables creados compilando esas funciones se denominan "MEX-files", aunque la extensión de nombre de archivo depende del sistema operativo y del procesador.

**Dreamweaver**

Uno de los programas más famosos y completos para la creación y edición de páginas web. Pese a las múltiples alternativas gratuitas que existen, Adobe Dreamweaver continua siendo el programa estrella, y hasta un no iniciado en los temas de programación y diseño web puede rápidamente conseguir unos resultados bastante buenos, gracias a la interfaz completa y adaptada de Dreamweaver, que nos guía en todas las partes del proceso. Dreamweaver es a la edición web como Photoshop lo es a la de edición de imágenes: un referente en su sector, que todos miraremos con deseo por más que busquemos alternativas.

Adobe Dreamweaver es una aplicación en programa de estudio (basada en la forma de estudio de Adobe Flash) que está destinada a la construcción, diseño y edición de sitios, vídeos y aplicaciones Web basados en estándares. Creado inicialmente por Macromedia (actualmente producido por Adobe Systems) es uno de los programas más utilizados en el sector del diseño y la programación web por sus funcionalidades, su integración con otras herramientas como Adobe Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Sus principales competidores son Microsoft Expression Web y BlueGriffon (que es de código abierto) y tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras. Hasta la versión MX, fue duramente criticado por su escaso soporte de los estándares de la web, ya que el código que generaba era con frecuencia sólo válido para Internet Explorer y no validaba como HTML estándar. Esto se ha ido corrigiendo en las versiones recientes.

Se vende como parte de Adobe Creative Suite. A partir de la compra de Macromedia por parte de Adobe. Las letras CS significan Creative Suite.

La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización, puesto que en este programa sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en Javascript-C, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino rutinas de Javascript que hace que sea un programa muy fluido y todo ello permite que programadores y editores web hagan extensiones para su programa y lo pongan a su gusto.

Las versiones originales de la aplicación se utilizaban como simples editores WYSIWYG. Sin embargo, versiones más recientes soportan otras tecnologías web como CSS, JavaScript y algunos frameworks del lado servidor.

Dreamweaver ha tenido un gran éxito desde finales de los años 1990 y actualmente mantiene el 90% del mercado de editores HTML. Esta aplicación está disponible tanto para la plataforma MAC como para Windows, aunque también se puede ejecutar en plataformas basadas en UNIX utilizando programas que implementan las API's de Windows, tipo Wine.

Como editor WYSIWYG que es, Dreamweaver permite ocultar el código HTML de cara al usuario, haciendo posible que alguien no entendido pueda crear páginas y sitios web fácilmente sin necesidad de escribir código.

No obstante, Adobe ha aumentado el soporte CSS y otras maneras de diseñar páginas sin tablas en versiones posteriores de la aplicación, haciendo que se reduzca el exceso de código.

Dreamweaver permite al usuario utilizar la mayoría de los navegadores Web instalados en su ordenador para previsualizar las páginas web. También dispone de herramientas de administración de sitios dirigidas a principiantes como, por ejemplo, la habilidad de encontrar y reemplazar líneas de texto y código por cualquier tipo de parámetro especificado, hasta el sitio web completo. El panel de comportamientos también permite crear JavaScript básico sin conocimientos de código.

Con la llegada de la versión MX, Macromedia incorporó herramientas de creación de contenido dinámico en Dreamweaver. En lo fundamental de las herramientas HTML WYSIWYG, también permite la conexión a Bases de Datos como MySQL y Microsoft Access, para filtrar y mostrar el contenido utilizando tecnología de script como, por ejemplo, ASP, ASP.NET, ColdFusion, JSP (JavaServer Pages) y PHP sin necesidad de tener experiencia previa en programación.

También podría decirse que, para un diseño más rápido y a la vez fácil, podría complementarse con Fireworks en donde se podría diseñar un menú u otras creaciones de imágenes (gif web, gif websnap, gif adaptable, jpeg calidad superior, jpeg archivo más pequeño, gif animado websnap) para un sitio web y después exportar la imagen creada y así utilizarla como una sola en donde ya llevará los vínculos para dicho sitio.

Resaltado de sintaxis

A partir de la versión 5, Dreamweaver admite el resaltado de sintaxis para los siguientes lenguajes por defecto:

* ActionScript
* Active Server Pages (ASP).
* ASP.NET (no se admiten en la versión CS4 - http://kb2.adobe.com/cps/402/kb402489.html)
* C#
* Cascading Style Sheets (CSS)
* ColdFusion
* Extensible HyperText Markup Language (XHTML)
* Extensible Markup Language (XML)
* Extensible Stylesheet Language Transformations (XSLT)

Historial de versiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proveedor** | **Versión mayor** | **Versión menor/nombre alternativo** | **Fecha de publicación** | **Notas** |
| **Macromedia** | **1.0** | 1.0 | Diciembre de 1997 | Primer lanzamiento, sólo para Mac OS. |
| 1.2 | Marzo de 1998 | Primera versión para Windows. |
| **2.0** | 2.0 | Diciembre de 1998 |  |
| **3.0** | 3.0 | Diciembre de 1999 |  |
| UltraDev 1.0 | Junio de 2000 |  |
| **4.0** | 4.0 | Diciembre de 2000 |  |
| UltraDev 4.0 | Diciembre de 2000 |  |
| **6.0** | MX | 29 de mayo de 2002 |  |
| **7.0** | MX 2004 | 10 de septiembre de 2003 |  |
| **8.0** | 8.0 | 13 de septiembre de 2005 |  |
| **Adobe** | **9.0** | CS3 | 16 de abril de 2007 | Sustituye a Adobe GoLive en la serie Creative Suite |
| **10.0** | CS4 | 23 de septiembre de 2008 |  |
| **11.0** | CS5 | 12 de abril de 2010 |  |
| **11.5** | CS5.5 | 12 de abril de 2011 |  |
| **12.0** | CS6 | 21 de abril de 2012 |  |
| **13.0** | CC | Abril de 2013 |  |

|  |  |
| --- | --- |
| **Color** | **Significado** |
| Rojo | Versión antigua |
| Amarillo | Versión antigua; soportada aún |
| Verde | Versión actual |

Requisitos del Sistema

**Windows**

* Procesador Intel® Pentium® 4 o AMD Athlon® de 64 bits.
* Microsoft® Windows® XP con Service Pack 2 (se recomienda Service Pack 3); Windows Vista® Home Premium, Business, Ultimate o Enterprise con Service Pack 1; o Windows 7.
* 512 MB de RAM.
* 1 GB de espacio disponible en el disco duro para la instalación; se necesita espacio libre adicional durante la instalación (no se puede instalar en dispositivos de almacenamiento flash extraíbles).
* Resolución de 1580 × 800 con tarjeta de vídeo de 16 bits.
* Java™ Runtime Environment 1.6 (incluido)
* Unidad de DVD-ROM
* Se necesita el software QuickTime 7.6.6 para la reproducción de contenido multimedia HTML5
* Este software no funcionará si no se activa. Es necesario disponer de conexión a Internet de banda ancha y registrarse para poder activar el software, validar suscripciones y acceder a servicios online.\* No está disponible la activación por teléfono.
* Procesador Intel multinúcleo.

**Mac**

* Mac OS X v10.6.8 o v10.7. Adobe Creative Suite 5, 5.5 y las aplicaciones de CS6 son compatibles con Mac OS X Mountain Lion (v10.8) si se instalan en sistemas basados en Intel.\*\*
* 512 MB de RAM.
* 1,8 GB de espacio disponible en el disco duro para la instalación; se necesita espacio libre adicional durante la instalación (no se puede instalar en un volumen que utilice un sistema de distinción entre mayúsculas y minúsculas en archivos, ni en dispositivos de almacenamiento flash extraíbles).
* Resolución de 1280 × 800 con tarjeta de vídeo de 16 bits.
* Java Runtime Environment 1.6.
* Unidad de DVD.
* Se necesita el software QuickTime 7.6.6 para la reproducción de contenido multimedia HTML5.
* Este software no funcionará si no se activa. Es necesario disponer de conexión a Internet de banda ancha y registrarse para poder activar el software, validar suscripciones y acceder a servicios online.\* No está disponible la activación por teléfono. SE NECESITA O PUEDE USAR DESDE ANDROID.

Microsoft.NET Framework 4 Client Profile (instalador web)

Este Client Profile se usa para ejecutar la mayoría de las aplicaciones cliente que tienen como destino .NET Framework 4. O dicho de otro modo, es una de las herramientas más básicas para crear aplicaciones para sistemas Windows (y teniendo en cuenta la extensión de sus sistemas operativos... podremos entender que esta herramienta esté en el número uno como la más descargada). Entre sus características y novedades, podemos destacar las mejoras en la BCL y el CLR, las nuevas aportaciones para los lenguajes C# y Visual Basic, y las mejoras en WPF y en el modelado y acceso a datos. Esta versión, además, es compatible con versiones anteriores.

Ahora, programadores y aspirantes, ya conocéis las herramientas más populares. ¿Echáis en falta alguna por aquí que queráis destacar? Si tenéis en alta estima alguna solución de software libre que hayáis utilizado, no dudéis en destacarla en los comentarios.

NET Framework es una tecnología que soporta la compilación y ejecución de aplicaciones y servicios Web XML de última generación. El diseño de .NET Framework está enfocado a cumplir los siguientes objetivos:

* Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
* Proporcionar un entorno de ejecución de código que minimiza los conflictos en el despliegue y versionado de software.
* Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceros desconocidos o que no son de plena confianza.
* Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.
* Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
* Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

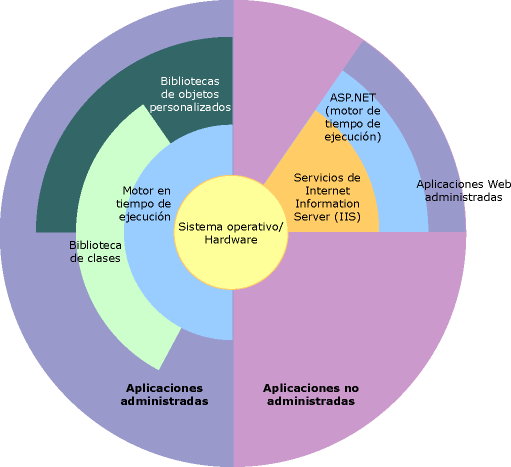
.NET Framework consta de dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la comunicación remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que promueven su seguridad y solidez. De hecho, el concepto de administración de código es un principio fundamental del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado. La biblioteca de clases es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como Web Forms y Servicios Web XML.

.NET Framework puede hospedarse en componentes no administrados que cargan Common Language Runtime en sus procesos e inician la ejecución de código administrado, con lo que se crea un entorno de software en el que se pueden utilizar características administradas y no administradas. En .NET Framework no sólo se ofrecen varios hosts de motor en tiempo de ejecución, sino que también se admite el desarrollo de estos hosts por parte de terceros.

Por ejemplo, ASP.NET hospeda el motor en tiempo de ejecución para proporcionar un entorno de servidor escalable para el código administrado.ASP.NET trabaja directamente con el motor en tiempo de ejecución para habilitar aplicaciones de ASP.NET y servicios Web XML, que se tratan más adelante en este tema.

Internet Explorer es un ejemplo de aplicación no administrada que hospeda el motor en tiempo de ejecución (en forma de una extensión de tipo MIME). Al usar Internet Explorer para hospedar el motor en tiempo de ejecución, puede incrustar componentes administrados o controles de Windows Forms en documentos HTML. Al hospedar el runtime se hace posible el uso de código móvil administrado, pero con mejoras significativas que solo el código administrado puede ofrecer, como la ejecución con confianza parcial y el almacenamiento aislado de archivos.

En la ilustración siguiente se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto. En la ilustración se representa igualmente cómo funciona el código administrado dentro de una arquitectura mayor.

.NET Framework en contexto

En las secciones siguientes se describen con más detalle las características principales de .NET Framework.

Características de Common Language Runtime

Common Language Runtime administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Estas características son intrínsecas del código administrado que se ejecuta en Common Language Runtime.

Con respecto a la seguridad, los componentes administrados reciben grados de confianza diferentes, en función de una serie de factores entre los que se incluye su origen (como Internet, red empresarial o equipo local). Esto significa que un componente administrado puede ser capaz o no de realizar operaciones de acceso a archivos, operaciones de acceso al Registro y otras funciones delicadas, incluso si se está utilizando en la misma aplicación activa.

El motor en tiempo de ejecución impone la seguridad de acceso del código. Por ejemplo, los usuarios pueden confiar en que un archivo ejecutable incrustado en una página Web puede reproducir una animación en la pantalla o entonar una canción, pero no puede tener acceso a sus datos personales, sistema de archivos o red. Por ello, las características de seguridad del motor en tiempo de ejecución permiten que el software legítimo implementado en Internet sea excepcionalmente variado.

Además, el motor en tiempo de ejecución impone la solidez del código mediante la implementación de una infraestructura estricta de comprobación de tipos y código denominada CTS (Common Type System, Sistema de tipos común). CTS garantiza que todo el código administrado es autodescriptivo. Los diversos compiladores de lenguaje de Microsoft y de otros fabricantes generan código administrado que se atiene al CTS. Esto significa que el código administrado puede consumir otros tipos e instancias administrados, al tiempo que se exige fidelidad de tipos y seguridad de tipos estrictamente.

Además, el entorno administrado del motor en tiempo de ejecución elimina muchos problemas de software comunes. Por ejemplo, el motor en tiempo de ejecución controla automáticamente la disposición de los objetos, administra las referencias a éstos y los libera cuando ya no se utilizan.Esta administración automática de la memoria soluciona los dos errores más comunes de las aplicaciones: la pérdida de memoria y las referencias no válidas a la memoria.

Además, el motor en tiempo de ejecución aumenta la productividad del programador. Por ejemplo, los desarrolladores pueden crear aplicaciones en el lenguaje que prefieran y seguir sacando todo el provecho del motor en tiempo de ejecución, la biblioteca de clases y los componentes escritos en otros lenguajes por otros colegas. El proveedor de un compilador puede elegir destinarlo al motor en tiempo de ejecución. Los compiladores de lenguajes que se destinan a .NET Framework hacen que las características de .NET Framework estén disponibles para el código existente escrito en dicho lenguaje, lo que facilita enormemente el proceso de migración de las aplicaciones existentes.

Aunque el motor en tiempo de ejecución está diseñado para el software del futuro, también es compatible con el software actual y el software antiguo. La interoperabilidad entre el código administrado y no administrado permite que los desarrolladores continúen utilizando los componentes COM y las DLL que necesiten.

El motor en tiempo de ejecución está diseñado para mejorar el rendimiento. Aunque Common Language Runtime proporciona muchos servicios estándar de motor en tiempo de ejecución, el código administrado nunca se interpreta. Una característica denominada compilación JIT (Just-In-Time) permite ejecutar todo el código administrado en el lenguaje máquina nativo del sistema en el que se ejecuta. Mientras tanto, el administrador de memoria evita que la memoria se pueda fragmentar y aumenta la zona de referencia de la memoria para mejorar aún más el rendimiento.

Por último, el runtime se puede hospedar en aplicaciones de servidor de gran rendimiento, como Microsoft SQL Server e Internet Information Services (IIS). Esta infraestructura permite utilizar código administrado para escribir lógica empresarial, al tiempo que se disfruta del superior rendimiento de los mejores servidores empresariales del sector que puedan hospedar el motor en tiempo de ejecución.

Biblioteca de clases de .NET Framework

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recolección de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

* Aplicaciones de consola Consulte Compilar aplicaciones de consola en .NET Framework.
* Aplicaciones GUI de Windows (Windows Forms) Consulte Windows Forms.
* Aplicaciones de Windows Presentation Foundation (WPF) Vea Windows Presentation Foundation.
* Aplicaciones de ASP.NET Consulte Desarrollar aplicaciones web mediante ASP.NET.
* Servicios de Windows. Consulte Introduction to Windows Service Applications.
* Aplicaciones orientadas a servicios utilizando Windows Communication Foundation (WCF). Consulte Desarrollar aplicaciones orientadas a servicios mediante WCF
* Aplicaciones habilitadas para el flujo de trabajo utilizando Windows Workflow Foundation (WF). Consulte Building Workflows in the .NET Framework.

Por ejemplo, las clases de Windows Forms son un conjunto completo de tipos reutilizables que simplifican enormemente el desarrollo de interfaces GUI para Windows. Si escribe una aplicación Web Form de ASP.NET, puede utilizar las clases de formularios Web Forms.