Imagen que contiene Texto

Descripción generada automáticamente

Toda la información que usa un “ordenador” debe estar representada en una secuencia de ceros y unos (dígitos) llamada código binario. Esto significa que cada unidad básica de información puede tener dos estados (0, 1). A dicha unidad se le conoce como bit o binary digit (b) y a la unión de ocho bits se le llama byte (B). Un byte puede representar 28 caracteres, es decir, 256 posibilidades distintas.

Cuando un dispositivo cualquiera maneja o almacena información en forma de bits (dígitos) decimos que ese dispositivo es digital. Cuando la información que maneja no es finita, como un reloj clásico o un termómetro de mercurio, decimos que es analógico.

El software es la parte intangible de un sistema informático, el equivalente al equipamiento lógico. El software suele ser el traductor entre el hombre y la máquina. Es el encargado de comunicarse con el hardware, es decir, se encarga de traducir todas las órdenes que el usuario comunica a órdenes comprensibles por el hardware.



Verás muchas definiciones de lo que es programar, pero en esencia, consiste en darle

órdenes y datos a un ordenador para recibir una serie de resultados o provocar cierto

comportamiento. Para ser técnicamente correctos, usaremos dispositivo en lugar de

ordenador e instrucción en lugar de orden.

Desarrollar software es o, al menos, debería ser mucho más que darle órdenes a un

dispositivo, mucho más que programar. De hecho, el tiempo que dedicamos, o que

deberíamos dedicar, a analizar, diseñar, probar, documentar y mantener el software es

mucho mayor que el tiempo que dedicamos exclusivamente a escribir líneas de código.



Podríamos decir que un programa es una serie de órdenes instrucciones secuenciadas u

ordenadas con una finalidad concreta y que devuelven un valor o realizan una función

determinada. Por ejemplo, una función que dados números te devuelva la suma o un

procedimiento que minimiza las ventanas de tu escritorio.

Por otro lado, una librería es un archivo, o contenedor lógico, que contiene una serie de

programas. A su vez, una aplicación está formada por varios programas con sus librerías

correspondientes, aunque podrían constar solamente de un programa. Cuando lo que tenemos son varias aplicaciones que pueden ejecutarse independientemente, una de otra, suele denominarse suite o “paquete integrado”, como por ejemplo nos sucede con MS Office o con Libre y Open Office.

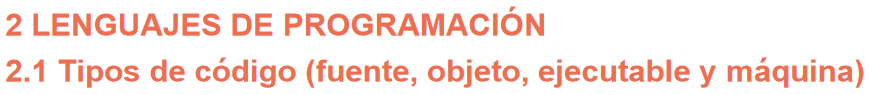
Por otro lado, una librería es un archivo, o contenedor lógico, que contiene una serie de

programas. Por ejemplo, cualquiera de los archivos DLL que encuentras en la carpeta

System32 de Windows.

Diagrama

Descripción generada automáticamente



Los dispositivos solo entienden un lenguaje muy complejo llamado lenguaje máquina. Existen unos lenguajes intermedios que, mediante unos traductores llamados compiladores, nos ayudan a comunicarnos con esos dispositivos.

Todos los programas se desarrollan en algún lenguaje de programación. Los

lenguajes de programación son, por lo tanto, lenguajes artificiales creados para que, al

traducirse a código máquina, cada una de las instrucciones de dicho lenguaje dé lugar a

una o a varias instrucciones máquina.

Al conjunto de TODAS las instrucciones que forman un programa le llamaremos **código**

**fuente**. En el caso de java, el código fuente se guarda en un fichero con la extensión “.java”.

Como hemos dicho, tenemos que transformar el código fuente de nuestro programa al código que entienda nuestro hardware. Esto se realiza con los compiladores, que generan una especie de **código intermedio** al que llamamos **código objeto, bytecode o código intermedio**. En el caso de Java, el compilador genera un archivo de extensión “.class”.

Si a estos objetos les añadimos las funciones que hemos usado de librerías y las

peculiaridades del sistema operativo en el que se va a ejecutar obtenemos el **código**

**ejecutable**, que es el conjunto de órdenes instrucciones directamente interpretable por el dispositivo que estamos usando. Suele usarse un enlazador.

Piensa que, un mismo código fuente, para poder ser ejecutado en diferentes sistemas

operativos necesita ser compilado de varias maneras.

Diagrama

Descripción generada automáticamente



Hay distintos tipos de archivos ejecutables, dependiendo cómo se traten las órdenes

instrucciones:

• Ejecutables portables. Se pueden ejecutar en varias plataformas. Por ejemplo, un

ejecutable Java es portable ya que utiliza un bytecode no asociado a un procesador

en concreto.

• Ejecutables no portables. Destinado a una plataforma concreta. Por ejemplo, un

ejecutable en C.

Sin embargo, en un sentido más general, un programa ejecutable no tiene por qué

necesariamente contener código máquina, sino que puede tener instrucciones a

interpretar por otro programa. Este tipo de ejecutables son conocidos con el nombre de

scripts. Ejemplo: Script en bash de Linux o un script en PHP.

Determinar si un archivo es ejecutable es sobre todo una cuestión de convención. Unos sistemas operativos se basan en la extensión de archivo (como la terminación .exe) y otros lo hacen leyendo los metadatos (como los bits de permiso de ejecución en Unix).



Algunos lenguajes como Java usan una aplicación que emula un SO dentro de otro SO y

que se conoce como máquina virtual.

Existen dos tipos de máquinas virtuales:

• Las máquinas virtuales **de sistema**: Se simula un ordenador completo dentro del nuestro a través de un programa de virtualización. Estos programas simulan una arquitectura hardware lo que permite instalar sistemas operativos diferentes a los creados para nuestra arquitectura. Otras aplicaciones son realizar pruebas o simular una red de ordenadores. Ejemplos: VMWare, VirtualBox...

• Las máquinas virtuales **de proceso**: se utilizan para independizar la ejecución de

un proceso del hardware subyacente. El bytecode (código objeto) es interpretado por la

máquina virtual que se encarga de traducir las operaciones para que funcionen con el

hardware específico de la máquina que estemos usando. Sirve de entorno de

ejecución para estos programas. La más conocida es la máquina virtual de java o JVM

(Java Virtual Machine). La JVM es un ejecutable en una plataforma específica,

capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

Diagrama

Descripción generada automáticamente



Diagrama

Descripción generada automáticamente

1. Tipo de ejecución

• **Lenguajes interpretados**

El uso de los lenguajes interpretados ha venido en crecimiento y cuyos máximos representantes son los lenguajes usados para el desarrollo web entre estos Ruby, Python, PHP (se interpreta del lado del servidor), JavaScript y otros como Perl, Smalltalk, MATLAB, Mathematica.

Un lenguaje interpretado es aquel en el cual sus instrucciones o más bien el código fuente, escrito por el programador en un lenguaje de alto nivel, es traducido por el intérprete a un lenguaje entendible para la máquina paso a paso, instrucción por instrucción.

A los lenguajes interpretados los vemos más en software de entornos web o terminales de comandos, ya que requieren de menores recursos y de acceso a archivos determinados.

• **Lenguajes compilados**

Un lenguaje compilado es aquel cuyo código fuente, escrito en un lenguaje de alto nivel, es traducido por un compilador a un archivo ejecutable entendible para la máquina en determinada plataforma. Con ese archivo se puede ejecutar el programa cuantas veces sea necesario sin tener que repetir el proceso por lo que el tiempo de espera entre ejecución y ejecución es ínfimo.

Dentro de los lenguajes de programación que son compilados tenemos la familia C que incluye a C++, Objective C, C# y también otros como Fortran, Pascal, Haskell y Visual Basic.

A los lenguajes compilados los vemos más en software de escritorio ya que requieren de mayores recursos y de acceso a archivos determinados. También por el peso mayor que estos suelen tener en sus archivos ejecutables.

• **Lenguajes virtuales**

Algunos autores clasifican a lenguajes como Java lenguajes virtuales o híbridos. Es un caso particular ya que hace uso de una máquina virtual que se encarga de la traducción del código fuente por lo que hay veces es denominado compilado e interpretado o virtual. Otra ventaja de la máquina virtual que usar Java es que le permite ejecutar código Java en cualquier máquina que tenga instalada la JVM.

1. Nivel de abstracción

Respecto del nivel de abstracción, podría definirse como el nivel de cercanía al

lenguaje natural, siendo un lenguaje de alto nivel el que se parece más al lenguaje

natural y un lenguaje de bajo nivel el que es más similar al lenguaje máquina.

• Bajo nivel

Los lenguajes de programación de bajo nivel son los que dan un conjunto de instrucciones aritmeticológicas que no permiten el uso de funciones que no estén ya contempladas en la arquitectura del hardware. Estos suelen ser una mera traducción del código máquina para facilitar su legibilidad y escritura.

• Alto nivel

Los lenguajes de programación de alto nivel son los más abstractos. Están más cerca de la forma natural de habla humana que de las instrucciones de código máquina. Son los lenguajes más usados en la actualidad e incluyen ejemplos como C++, Java, PHP, Python…

• Medio nivel

Algunos autores clasifican a lenguajes como C un lenguaje de programación de medio nivel puede acceder a registros del sistema y direcciones de memoria, todas propias de lenguajes de bajo nivel.

Imagen que contiene Tabla

Descripción generada automáticamente

1. Paradigma de programación

• Imperativos

Los lenguajes imperativos se basan en conocer el estado de la máquina y modificarlo mediante instrucciones. Se llaman imperativos porque usamos órdenes para decir al ordenador dispositivo qué debe hacer y cómo hacerlo. La mayoría de las arquitecturas de ordenadores siguen una filosofía imperativa por lo que la traducción de un lenguaje imperativo a código máquina es más sencilla que la de los lenguajes declarativos.

Son lenguajes imperativos prácticamente todos los que conocemos: Perl, C, Java, PHP, Python, …

• Declarativos

Los lenguajes declarativos fijan un objetivo, pero no el camino para llegar a él. Se le indica al ordenador qué queremos obtener y se le detalla una descripción del problema. La solución se encuentra con estos elementos siguiendo una lógica interna. Son lenguajes más cercanos a las matemáticas.

Son lenguajes declarativos Prolog, LISP y uno que seguro que te resulta familiar: SQL.

Cada lenguaje de programación tiene unas características determinadas que lo hacen más o menos apropiado para el contexto donde lo vayas a usar, pudiendo ser determinante estos factores:

1. El sector productivo al que va dirigida tu aplicación (medicina, académico,

videojuegos...)

2. En qué dispositivo va a ejecutarse (móviles, escritorio, IoT...)

3. La plataforma sobre la que va a ejecutarse (Windows, Linux, Mac, …

Imagen que contiene Texto

Descripción generada automáticamente

Podríamos definir la ingeniería como "la práctica de organizar el diseño y la construcción de cualquier artefacto que transforme el mundo que nos rodea para satisfacer alguna necesidad" (Rogers, 1983).

Rogers no era un desarrollador de software, sino un ingeniero especializado en desarrollo de motores. Sin embargo, su definición capta la esencia de la ingeniería y tiene una amplia aplicabilidad que se extiende al software. Gran parte de nuestro esfuerzo en el desarrollo de aplicaciones informáticas es el diseño y la construcción de programas informáticos para satisfacer alguna necesidad - de las personas, las organizaciones o la sociedad en general - con un efecto tangible en el mundo real.

La ingeniería del software es una disciplina que engloba herramientas, técnicas, recomendaciones y métodos que se utilizan en la creación de una aplicación/sistema/aplicativo/plataforma informática o, más genéricamente, en el desarrollo de un proyecto software que resuelve un problema o suple una necesidad.

**El desarrollo de una aplicación se entiende como un intercambio comercial de un producto (el software) entre un desarrollador y un cliente pudiendo ambas partes ser empresas, organizaciones o personas físicas y ser una o varias.**

El ingeniero de software, por tanto, se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en tiempo y forma, es decir, en un plazo determinado y con presupuesto y requisitos previstos.

La ingeniería de software, en esencia, incluye el análisis previo de la situación (qué y cuándo lo haremos), el diseño del proyecto (cómo lo haremos), el desarrollo del software, las pruebas necesarias para su correcto funcionamiento y la documentación e implementación del sistema o puesta en marcha.

Texto, Tabla

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente



El término Crisis del Software1 fue acuñado a principios de los años 70, cuando la planificación de un proyecto software era prácticamente inexistente. Seguían un proceso de desarrollo bastante artesanal, sin una metodología o un camino a seguir para su desarrollo.

Cada año, desde 1994, el Grupo Standish crea una lista de 10 atributos y su peso relativo que ellos llaman los factores de éxito. Los tres primeros para 2018 son:

• Latencia de decisión: Las decisiones deben tomarse lo antes posible.

• Alcance mínimo: A mayor tamaño del proyecto mayor índice de fracaso.

• Responsables del proyecto: Mejor una única persona que no una Junta Directiva, por ejemplo.

Imagen que contiene Texto

Descripción generada automáticamente

Las fases del desarrollo de software, conocidas como ciclo de vida del software, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.

Imagen que contiene Texto

Descripción generada automáticamente

1. ANÁLISIS

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El análisis de una aplicación pretende determinar las necesidades que debe cubrir en directo contacto con el cliente. En esta fase se especifican los requisitos funcionales y no funcionales del proyecto.

El éxito de un proyecto software se basa en entender y comprender el problema que necesita resolver el cliente y, una vez comprendido, darle solución.

1. DISEÑO

Texto

Descripción generada automáticamente

En el diseño (o modelado del software) se plantea una solución para la aplicación y se realiza un modelo utilizando diferentes técnicas. Dicho diseño suele hacerse por niveles, comenzando con una idea general que se va dividiendo en componentes para luego modelar cada uno (diseño descendente/drop-down).

...Importante

No hay que confundir la fase de DISEÑO en términos de ISW con el DISEÑO GRÁFICO

(aspecto de nuestra aplicación). Son cosas completamente diferentes.

1. CODIFICACIÓN

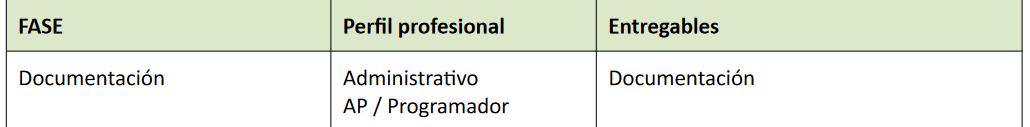
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la codificación se realiza el programa atendiendo a todos sus componentes; esto incluye elementos como la base de datos, servidores o comunicaciones.

En muchos proyectos se comete el error de suponer que, por un lado, el programador es el “chico para todo” en el desarrollo del software y que, por otro, cualquier analista o consultor es también un buen programador. No todo profesional del software está capacitado para tratar directamente con el cliente.

1. PRUEBAS



La documentación es la fase que primero se descarta cuando el proyecto está fuera de plazo o de presupuesto y es vital para que, si algún miembro del equipo debe ser reemplazado, se amplía el equipo o se retoma el proyecto tras un periodo de tiempo poder continuar/comenzar lo antes posible.

La fase de documentación consiste en dejar por escrito las decisiones técnicas que se han tomado. Desde por qué un framework y no otro hasta por qué se han creado 10 tablas en la base de datos y no 200.

1. EXPLOTACIÓN

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La explotación (no confundir con implementación) consiste en publicar la solución final en la plataforma destino o entregar al cliente el producto final en el formato acordado.

Muy pocas personas en la organización tienen (o deben tener) acceso a los servidores dónde residen las aplicaciones que están publicadas (servidores de producción). Esta fase recae en los más veteranos (senior).

G) MANTENIMIENTO

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez entregado el proyecto este pasa a una fase denominada mantenimiento en el que se tratan:

• El **evolutivo**: modificaciones y/o eliminaciones en el producto necesarias por un cambio en las necesidades del cliente o por cambios en requisitos externos (normativos, legales,…).

• El **correctivo**: acciones para mejorar la calidad del producto sistemas en cualquiera de sus aspectos: reestructuración del código, definición más clara del sistema y optimización del rendimiento.

• El **adaptativo**: modificaciones necesarias para adaptarse a variaciones del entorno (contexto) en los que el sistema opera, por ejemplo, cambios de hardware, de base de datos, comunicaciones, etc

...Interesante

En este punto el software está entregado y facturado por lo que se le suele restar

importancia pese a que muchas empresas de desarrollo de software pagan gran parte de las nóminas de sus empleados con las cuotas de mantenimiento de todos sus clientes. Esta fase suele recaer en los recién llegados (junior).



Gráfico, Diagrama

Descripción generada automáticamente



Durante todo el ciclo de vida del software usamos modelos para representar la realidad.

Con la modelización del software, cada modelo es una representación abstracta y

dinámica de alguna visión del sistema, que puede cambiar durante el desarrollo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Las técnicas de modelización más usadas en el desarrollo de software se definen en el Lenguaje de Modelización Unificado (UML), uno de los estándares más populares y exitosos actualmente que incluye casi una veintena de diagramas.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



Los ciclos de vida se combinan y explotan para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Existen tres enfoques:

**Metodologías tradicionales.**

* Basadas en combinaciones de los ciclos de vida anteriores.
* Fases demasiado largas, feedback insuficiente.
* Ejemplos: METRICA3, RUP, MERISE.

**Metodologías ágiles.**

* Fomentan el reajuste continuo de los objetivos de desarrollo con las necesidades y expectativas del cliente, y proporcionan procesos de desarrollo de software "más ligeros", más rápidos y más "ágiles" que pueden adaptarse a los inevitables cambios en los requisitos del cliente.

• Ejemplos: SCRUM, KANBAN, XP PROGRAMMING.

**Metodologías guiadas por pruebas o TestDriven** (TDD) (2003).

* TDD o Test-Driven Development (desarrollo dirigido por tests) es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.

Los ciclos de vida se combinan y explotan para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos.