DESARROLLO DEL SOFTWARE

¿Qué es un ordenador?

Los primeros ordenadores aparecen a partir de los 40. Un ordenador acepta datos de entrada, los procesa y produce unas salidas (resultados). Se componen de elementos físicos (hardware) y lógicos (programas, software). Las órdenes de los usuarios deben ser traducidas para que las entienda el ordenador.

¿Qué es él software?

Es la parte intangible o lógica de un sistema informático. Se desarrolla para llevar a cabo una tarea determinada, se comunica con el hardware y le dice qué tiene que hacer. Se encarga de traducir las instrucciones de los usuarios.

Algunos conceptos claves son:

→Se desarrolla, no se fabrica

→Es lógico, no físico

→No se estropea (por lo general)

→Puede desarrollarse a medida

→Posibilita el uso del ordenador

→Se desarrolla usando un lenguaje de programación y metodología de programación

¿Qué es una aplicación?

Un programa o aplicación, se compone de un conjunto de instrucciones. Se desarrolla usando lenguaje de programación. Las instrucciones le indican al ordenador el programa o pasos que debe ejecutar. Si el ordenador no entiende alguna instrucción, lo notificará mediante un lenguaje. En este ciclo pasaremos de ser usuarios a programadores de aplicaciones.

EJEMPLOS: Sistemas operativos, aplicaciones de contabilidad, de diseño gráfico, videojuegos, etc.

Programa: Conjunto de pasos o instrucciones que indican al ordenador cómo realizar un proceso.

Ejecutar: Consiste en iniciar un programa y ponerlo en ejecución.

Librería: Conjunto de programas y funciones que realizan tareas concretas (BBDD, etc.).

Entorno de desarrollo (IDE): Herramienta que facilita y posibilita el desarrollo del software.

SOFTWARE A MEDIDA VS. ESTÁNDAR

MEDIDA: Se desarrolla según las especificaciones o requerimientos de una empresa u organismo. Se adecuan a la actividad de dicha empresa y organismo.

Características:

→Necesita un tiempo de desarrollo

→Se adapta a las necesidades específicas de dicha empresa

→Suele contener errores y necesita de una etapa de adaptación y mantenimiento

→Es más costoso que el software estándar

ESTÁNDAR: Es un software genérico y resuelve múltiples necesidades, suele incluir herramientas de configuración.

Características:

→Se compra ya desarrollado

→Suele tener menos errores que el software a medida

→Suele ser más barato

→Suele incluir funciones que nunca serán usadas

¿Qué es un lenguaje de programación?

Se trata de lenguaje artificial creado por programaciones. Permite traducir las instrucciones de un programa a código comprensible por el ordenador- Son lenguajes mucho más complejos y precisos que el lenguaje máquina. Cada instrucción puede dar lugar a muchas instrucciones de lenguaje máquina. Se componen de un conjunto de símbolos y reglas.

Lenguaje de máquina: Lenguaje que solo es comprensible por el ordenador.

Programa: Conjunto de instrucciones escritas en un lenguaje de programación.

Instrucción: Cada una de las sentencias y órdenes que forman parte de un programa.

Palabra reservada: Cada uno de los símbolos que componen la sintaxis de un lenguaje.

Semántica: Reglas que definen la combinación de los símbolos de un lenguaje de programación.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Los lenguajes de programación han evolucionado a lo largo de los años. Actualmente, los lenguajes son más amigables y facilitan la tarea del programador. A veces, la facilidad a la hora de programar implica crear programas más lentos y pesados.

LENGUAJE DE MÁQUINA

Posee instrucciones complejas e ininteligibles y necesita ser traducido. Fue el primer lenguaje usado. Difiere para cada procesador, es decir las instrucciones son diferentes para cada

ordenador. En la actualidad solo se usa para determinados módulos de un sistema operativo, drivers.

LENGUAJE DE MEDIO NIVEL

Facilita la labor de programación. Se centra en el hardware, pero usa mnemotécnicos comprensibles para el programador. Hay que compilarlos (traducirlos) para que sean comprensibles para el ordenador. Trabaja con los registros del procesador y direcciones de memoria físicas. Es difícil de entender y de usar.

LENGUAJE DE ALTO NIVEL

Poseen una forma de programar intuitiva y sencilla. Son más cercanos a los lenguajes humanos, incorporan librerías y funciones predeterminadas que ayudan al programador en su labor. Suelen ofrecer frameworks, que incorporan funciones y componentes que facilitan el desarrollo. La mayoría de los lenguajes actuales engloban esta categoría.

PARADIGMAS DE PROGRAMACIÓN

Un paradigma de programación define la forma en la que se desarrolla el programa.

PROGRAMACIÓN ESTRUCTURADA

Consiste en una secuencia ordenada y organizada de instrucciones, es fácil de entender pues son programas sencillos y rápidos. Posee tres estructuras, secuencia, selección e iteración.

El inconveniente es que es un único bloque de programa.

PROGRAMACIÓN FUNCIONAL

Consiste en descomponer el programa en funciones o módulos por lo que es un programa modular y estructurado.

El inconveniente es que el programa puede crecer en gran medida y ser complejo.

PROGRAMACIÓN ORIENTADA A OBJETOS

Consiste en representar entidades del mundo real. Permite reutilizar el código y contiene polimorfismo, herencia y encapsulación.

PROGRAMACIÓN LÓGICA

Consiste en representar predicados y relaciones. Es muy utilizado para aplicaciones de inteligencia artificial

PROCESO DE TRADUCCIÓN/COMPILACIÓN

TRADUCTORES DE CÓDIGO

Un traductor traduce un lenguaje de alto nivel a lenguaje de máquina o ensamblador. Existen dos tipos de traductores que son los **intérpretes** y los **compiladores**.

En función de la forma de ejecutar un lenguaje existen tipos: compilador, interpretados y virtuales.

Lenguajes compilados

Necesitan un compilador para traducir el código fuente a código máquina. Se ejecutan más rápida que los programas interpretados o virtuales.

Precisan de un programa enlazador que permite unir el código objeto con el código objeto de librerías.

Aunque el código es más seguro, no son tan flexibles para modificarlos como los lenguajes interpretados.

Lenguajes interpretados

No generan código objeto. El intérprete es un programa que tiene que estar cargado en la memoria. Interpreta cada sentencia del programa y la ejecuta.

Las instrucciones se traducen al momento en que van siendo ejecutadas.

Lenguajes virtuales

Son más portables que los lenguajes compilados. El código se genera tras la compilación en un código intermedio o bytecode. Puede ser interpretado por una máquina virtual instalada en cualquier equipo. Son más lentos, pero más versátiles.

Código fuente: código de un programa, escrito por un programador en un lenguaje de programación.

Compilar: proceso que traduce el código fuente en código objeto.

Código objeto: código de máquina generado tras la compilación del código fuente.

Librería: código externo que se incluye al programa para proporcionar funcionalidad adicional.

Archivo ejecutable: programa que puede ser ejecutado en el ordenador.

INTERPRETACIÓN CÓDIGO

Un intérprete traduce el código fuente línea a línea. Tiene que estar en memoria ejecutándose para ejecutar el programa. El código fuente del programa también se carga en memoria para poder ser interpretado.

COMPILACIÓN DEL CÓDIGO

Un compilador traduce código fuente a código máquina. Se instala en cada máquina en la que queramos compilar el programa, depende de la arquitectura hardware de la máquina.

Supone unas fases:

Preprocesado: se traducen y se ejecutan los comandos de preprocesamiento.

Generación de código intermedio: generación de código máquina.

Enlazado: se enlazan el código objeto con librerías externas.

DESARROLLO DE UNA APLICACIÓN

FASES DEL DESARROLLO DE UNA APLICACIÓN

El software se desarrolla siguiendo un paradigma o una metodología de desarrollo.

Generalmente, los paradigmas suelen tener varias fases en común.

Fase inicial

Es la fase más compleja. Incluye planificación del proyecto y estimación de costes. Precisa de expertos en planificación de proyectos y se desarrollan documentos muy importantes para el proyecto.

Fase de análisis

Consiste en analizar un problema. Hay que recopilar, examinar y formular los requisitos del cliente. Análisis de restricciones, entrevistas con el cliente y los usuarios finales. Se genera un documento vinculante a modo de contrato entre el cliente y el desarrollador.

Fase del diseño

Consiste en determinar los requisitos generales de la arquitectura de la aplicación. Se define cada subconjunto de la aplicación y los documentos son mucho más técnicos. Esta fase involucra a los jefes del proyecto, arquitectos de software y analistas.

Fase de implementación

Consiste en implementar el software usando lenguajes de programación, etc. Se crea una documentación muy detallada en la que se incluye el código fuente. Parte del código suele comentarse sobre el propio código fuente generado. El detalle es máximo pensando en el mantenimiento y soporte futuro que tendrá el programa.

Pruebas

Se realizan pruebas para garantizar que la aplicación se ha programado según las especificaciones. A modo preliminar, se pueden considerar dos categorías:

→Funcionales: se prueba que la aplicación hace lo que tiene que hacer

→Estructurales: se efectúan pruebas técnicas sobre el sistema

Explotación

Se instala el software en el entorno real de uso. Se trabaja con el software de forma cotidiana y se recogen los errores y las correcciones en un nuevo documento de mantenimiento. Los programadores y analistas revisan esos fallos para mejorar el software y aprender de los errores.

Mantenimiento

Se realizan procedimientos correctivos sobre el programa. Siempre hay que tener delante la documentación técnica de la aplicación y las operaciones de mantenimiento se deben documentar para dejar constancia de los cambios.

Retirada

El software ha llegado al final de su vida útil. No resulta rentable seguir ampliándolo o manteniéndolo. Llegados a este punto, el ciclo puede comenzar de nuevo comprando un nuevo software o desarrollarlo a medida.

DOCUMENTACIÓN

La documentación es vital para saber cómo usar la aplicación final, en cada fase se generan uno o más documentos. Como mínimo, cada aplicación debe tener:

→Manual de usuario: explica cómo usará el usuario la aplicación.

→Manual técnico: documentación dirigida a los técnicos.

→Manual de instalación: detalla el proceso de instalación de la aplicación.

ROLES DEL DESARROLLO

Arquitecto de software

Decide cómo se realiza el proyecto y cómo va a estructurarse. Tiene un amplio conocimiento de las tecnologías, los frameworks y las librerías. Decide y forma los recursos del desarrollo de un proyecto.

Jefe de proyecto

Dirige el curso del proyecto y puede ser un analista con experiencia, un arquitecto o una persona dedicada a ese puesto en exclusividad. Debe saber gestionar un equipo y lidiar con los tiempos. Trata de manera continua y fluida con el cliente.

Analista de sistemas

Esta persona realiza un estudio exhaustivo del problema que va a llevarse a cabo. Efectúa el análisis del diseño de todo el sistema. Este perfil requiere mucha experiencia, y también suele involucrarse en reuniones con el cliente.

Programador

Conoce en profundidad el lenguaje de programación. Se encarga de codificar las tareas encomendadas por el analista o el analista programador. Su misión es la de codificar y probar los diferentes módulos de la aplicación.

PARADIGMAS DE DESARROLLO

Los paradigmas son inflexibles. Cada etapa tiene que finalizar para pasar a la siguiente, el más clásico es el modelo en cascada.

METODOLOGÍAS ÁGILES

Responden mejor ante un cambio de especificaciones o un error. Se basan en el manifiesto ágil que valora:

→A los individuos y su interacción, por encima de los procesos y las herramientas

→Al software que funciona, por encima de la documentación exhaustiva

→A la colaboración con el cliente, por encima de la negociación contractual

→A la respuesta al cambio, por encima del seguimiento de un plan