5. SOFTWARE BASE EN SISTEMAS INFORMÁTICOS

Contenidos

1. INTRODUCCIÓN
2. CICLO DE VIDA DEL SOFTWARE
3. CLASIFICACIÓN DEL SOFTWARE
4. ENTORNOS OPERATIVOS
5. SOFTWARE DE APLICACIÓN
6. ARQUITECTURAS DEL SOFTWARE
7. INSTALACIÓN Y CONFIGURACIÓN DE APLICACIONES
8. TESTING DE APLICACIONES
9. COMPARATIVA DE APLICACIONES

1. Introducción

SOFTWARE: Conjunto de componentes lógicos y no tangibles necesarios para llevar a cabo una tarea específica de nuestro sistema.

Imprescindible en un sistema informático. Da órdenes al hardware para llevar a cabo las tareas indicadas por el usuario.

FIRMWARE: software que contiene el hardware. Realiza funciones específicas para que el hardware funcione de forma eficiente.

1. Introducción

Características del software:

- Es lógico, no físico
- Se desarrolla, no se fabrica
- No se estropea
- En ocasiones se puede construir a medida

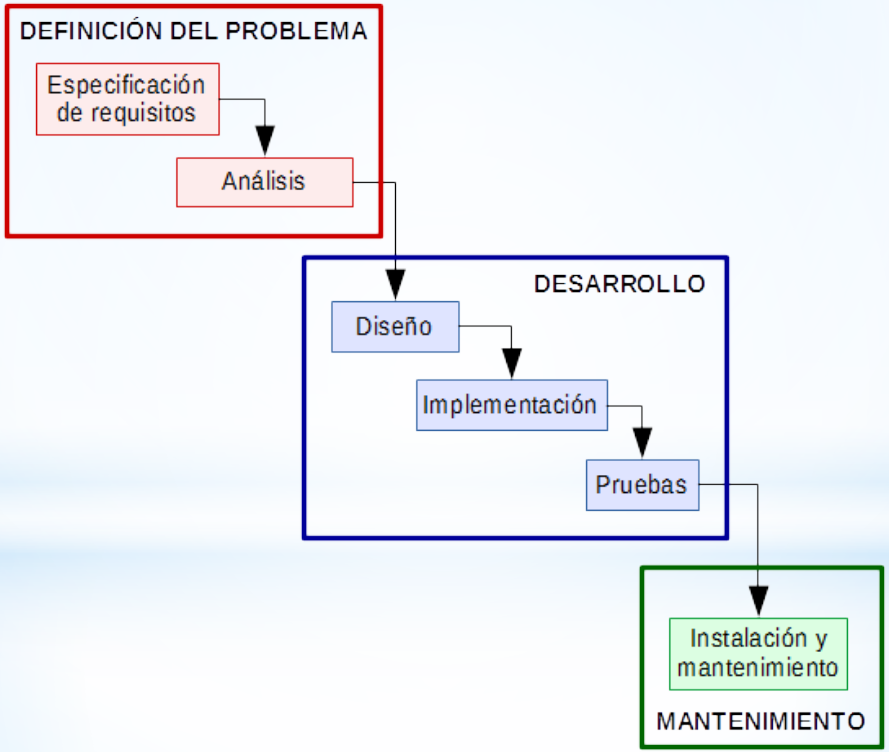
2. Ciclo de vida del software

Con este término se hace referencia a todas las fases de desarrollo del software, desde la fase inicial hasta la fase final.

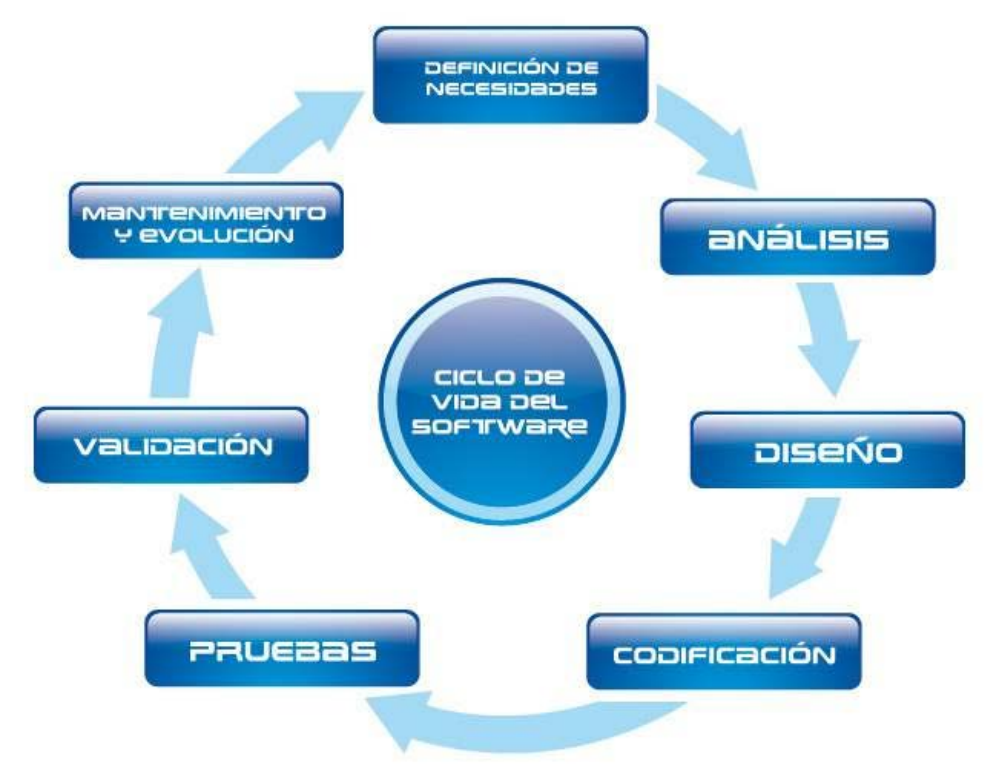
La disciplina que se encarga de su estudio es la Ingeniería del Software.

Esta disciplina proporciona un marco de trabajo para construir software de calidad.

2. Ciclo de vida del software



2. Ciclo de vida del software



2. Ciclo de vida del software

Definición de necesidades: definir qué se quiere obtener

Análisis de requisitos: recopilar, examinar y formular los requisitos del cliente.

Diseño: arquitectura general de la aplicación en base a unos requisitos generales. Se suele realizar en varias fases y en cada una de ellas se va definiendo la aplicación con más detalle.

2. Ciclo de vida del software

Programación: implementación del software en un lenguaje de programación concreto atendiendo a lo definido en el diseño.

Pruebas: validación de los módulos individuales para confirmar que se ajustan a los requisitos

Integración: pruebas para validar que los diferentes módulos se integran con la aplicación.

2. Ciclo de vida del software

Prueba beta o de validación: verificar que el software cumple con las especificaciones originales.

Documentación: recopilar la información de todo el proceso. Fundamental para desarrollos futuros.

Implantación: instalación del software en un entorno real

Mantenimiento: mantenimiento correctivo y gestión de actualizaciones

2. Ciclo de vida del software

Ejemplo:

1. Definición de necesidades

   Frutería que decide crear una tienda virtual de venta online para vender sus productos. Los clientes se reúnen con los ingenieros del sw para contar lo que quiere. El resultado será una serie de requisitos que plasmen de forma técnica características, limitaciones, … del producto final.

2. Análisis de requisitos

   Se busca analizar qué funciones realiza el sistema, sin entrar en detalles

   - Páginas con productos
   - Pasarela de pago
   - Gestión de stocks
   - Apartado para contactar con la frutería

2. Ciclo de vida del software

3. Diseño

   Se busca el cómo de cada una de las funcionalidades anteriores

   - Gestor de contenidos
   - TPV virtual o Paypal
   - Base de datos. Diseño de tablas

4. Programación

   Programación de las distintas páginas web, carga de los datos en la base de datos, módulos de comunicación bancaria.

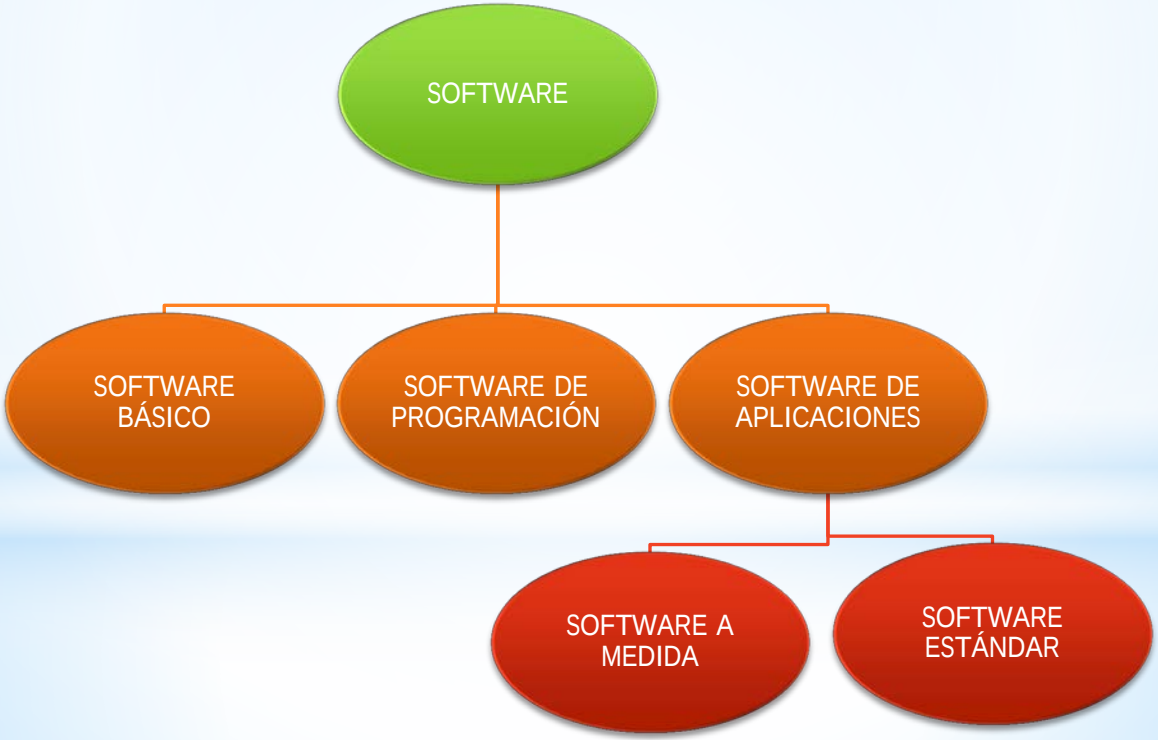
2. Ciclo de vida del software

5. Pruebas
   - Visualización de productos
   - Configuración de pedido
   - Gestión del pago
   - Actualización de datos en la base de datos

6. Implantación
   - Despliegue de la aplicación web
   - Verificación que todo funciona correctamente en producción

7. Mantenimiento

3. Clasificación del software



3. Clasificación del software

Software de sistema: permite administrar y mantener los recursos del sistema. Se denominan también entornos operativos.

Sistemas operativos, drivers, herramientas de diagnóstico y reparación, …

Software de aplicación: permiten al usuario llevar a cabo tareas específicas

Suites de ofimática, diseño gráfico, contabilidad, …

3. Clasificación del software

Software de programación: herramientas para que el programador desarrolle programas informáticos

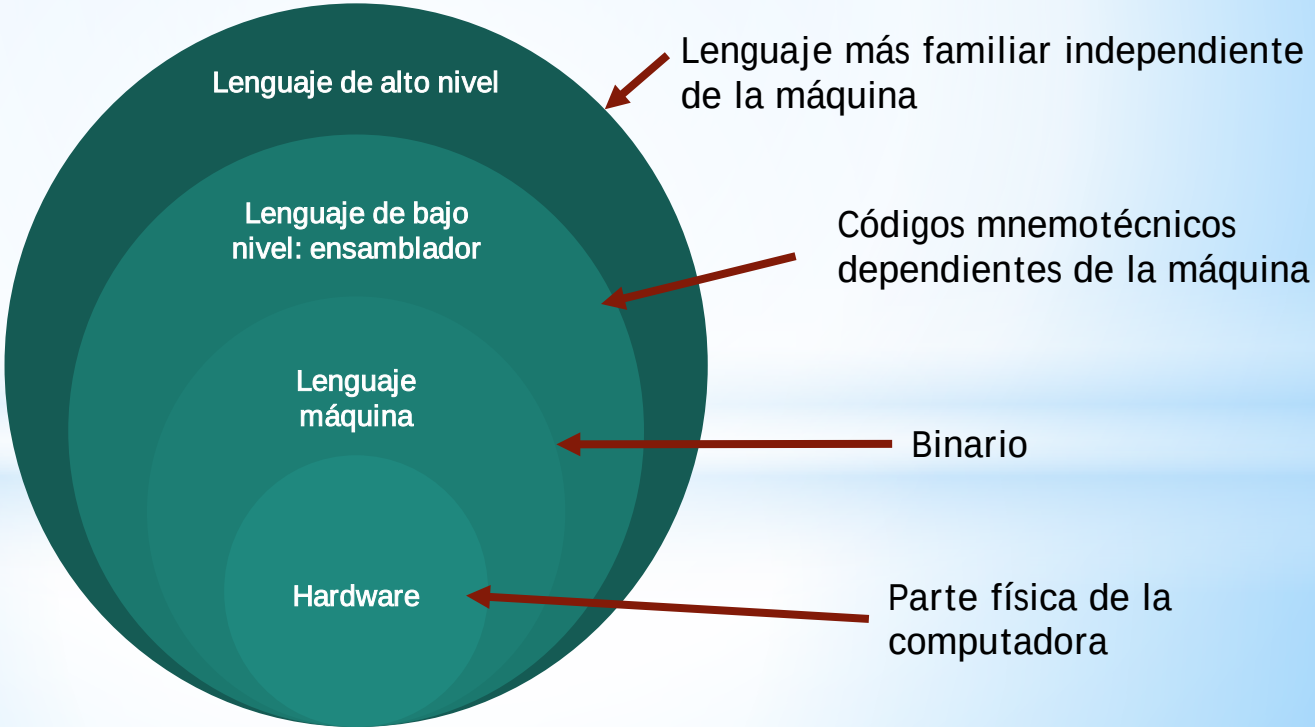
Compiladores, intérpretes, depuradores → Entornos de desarrollo integrados (IDE)

Se emplean lenguajes de programación, que son notaciones para escribir programas. Cuenta con una gramática o reglas específicas de cada lenguaje.

C/C++, Pascal, PHP, Javascript, Python, …

3. Clasificación del software

Los lenguajes de programación también tienen su propia clasificación



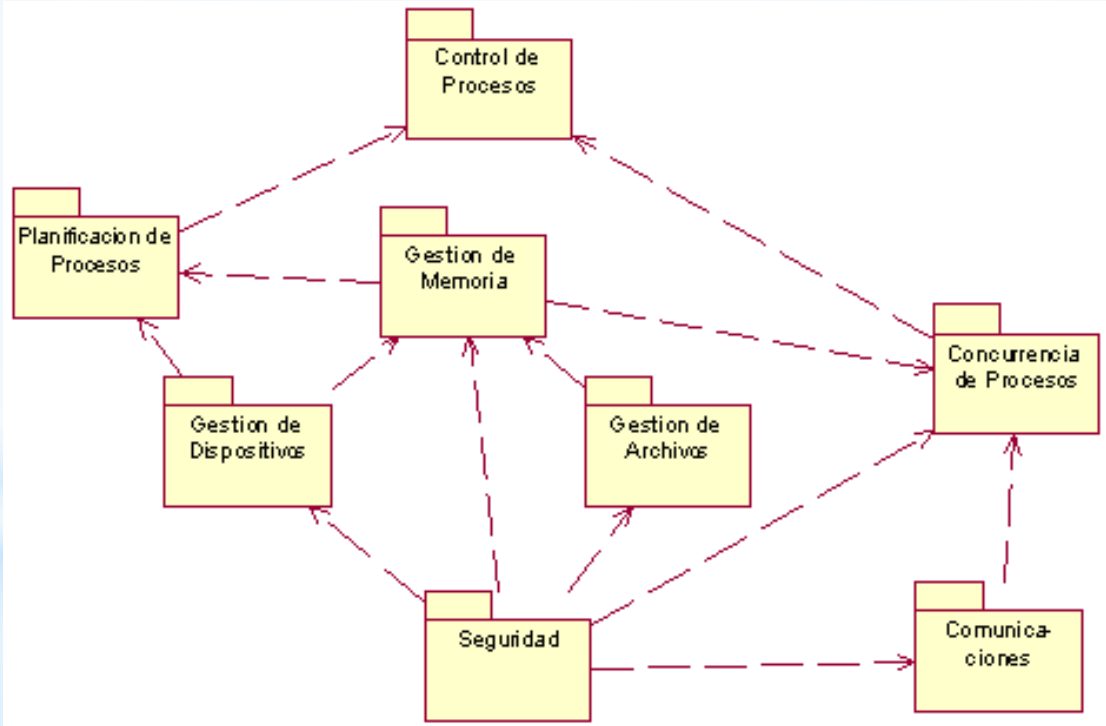
4. Entornos operativos

Entornos operativos: engloban el sistema operativo, su interfaz y algunas aplicaciones que vienen con él (administrador de archivos, programas de configuración, ...)

Funciones de los sistemas operativos:

- Control de recursos
- Manejo de dispositivos de E/S
- Gestión de tareas
- Sistemas de archivo de disco
- Interfaz gráfica
- Gestión de memoria

4. Entornos operativos



4. Entornos operativos

Características de los entornos operativos actuales:

- Interfaz gráfica
- Multiusuario
- Multitarea
- Soportan varios sistemas de archivo
- Ofrecen múltiples herramientas

4. Entornos operativos

Importancia de los sistemas operativos concebidos para dispositivos móviles

5. Software de aplicación

- En general se trata de un programa compilado o interpretado escrito en algún lenguaje de programación que necesita de un sistema operativo para funcionar.
- Tendrá una condiciones concretas de instalación y unas necesidades de espacio en disco, procesador, RAM, ...
- Software estándar vs Software a medida
- Software vertical (para un sector o industria específico) vs Software horizontal (útil para un amplio sector de usuarios)
- Diferentes extensiones de archivo para los diferentes tipos de aplicaciones (docx, pdf, jpg, mp3, tar, php, ...)

5. Software de aplicación

Hasta hace no mucho, las aplicaciones estaban en un equipo concreto. Ahora hay otros tipos de aplicaciones:

- Aplicaciones web: residen en un servidor y se ejecutan en un cliente. Parte del código se ejecuta en el servidor (PHP, Python, …) y parte en el cliente (Javascript). Son multiplataforma.

- Apps móviles: corren en sistemas operativos móviles. Pueden ser aplicaciones nativas o aplicaciones web o híbridas.

5. Software de aplicación

- Widgets y Plugins: aplicaciones o programas que se añaden a otras aplicaciones para aumentar su funcionalidad. También se llaman complementos o add-ons. Los plugins son menos autónomos que los widgets.

5. Software de aplicación

- Aspecto importante: los tipos de licencia del software (propietario, shareware, freeware, software libre)

Tipo de licencias de software	Limitaciones	Precio cero	Permiso de copia y redistribución	Código Fuente y permiso para modificarlo (sin ánimo de lucro)	Código Fuente y permiso para modificarlo (con fines comerciales)
Propietario	Ninguna	No	No	No	No
Shareware o de Evaluación	a) No 100% funcional, o b) uso por tiempo limitado	Sí	Sí	No	No
Freeware	Ninguna	Sí	Sí	No	No
Semilibre	Ninguna	Sí	Sí	Sí	No
Libre / Open Source / GPL	Ninguna	Sí*	Sí	Sí	Sí

3. Clasificación del software

Otros conceptos

- Abandonware: engloba todo el software comercial sin soporte por quedar obsoleto

- Crippleware: versiones básicas y limitadas de un programa. Tienen menos funcionalidades, consumen menos memoria, ocupan menos espacio en disco… que el programa completo original. Son generalmente gratuitas.

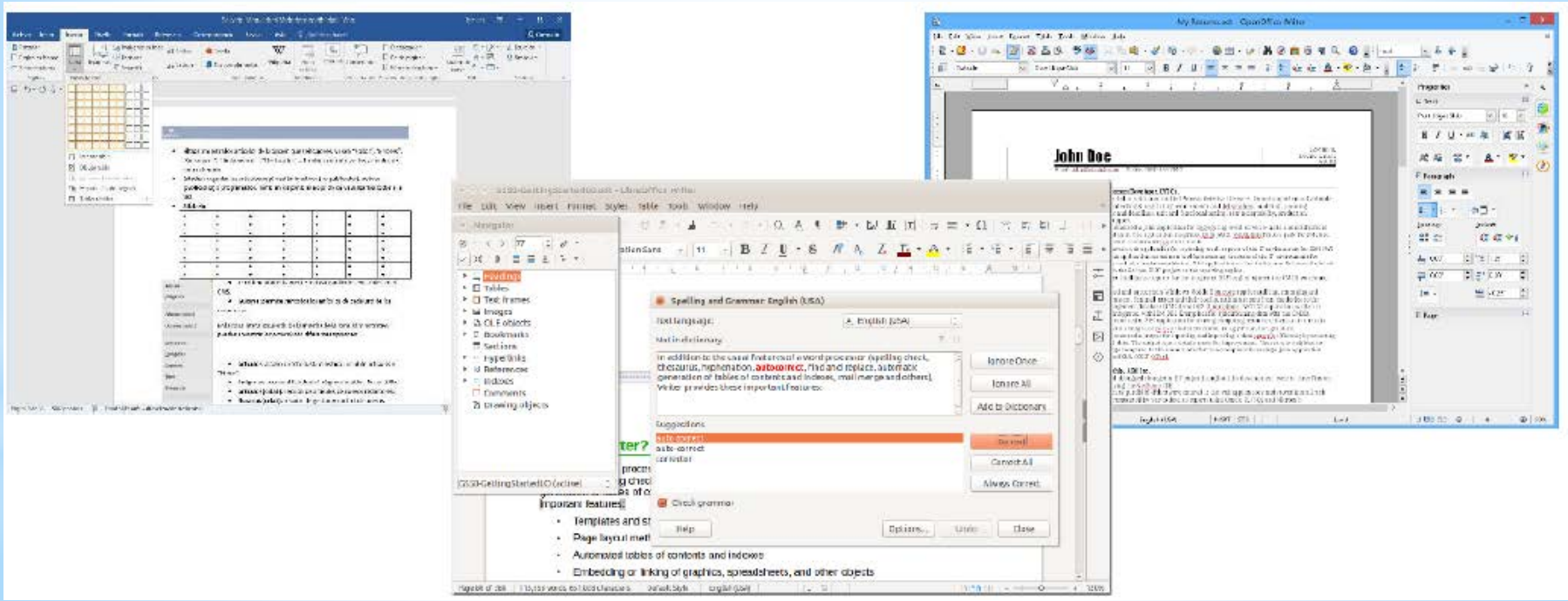
5. Software de aplicación

Ofimática: procesadores de texto, hojas de cálculo, gestores de bases de datos, editores de presentaciones, agendas, …

Se pueden agrupar en suites, que ofrecen en un conjunto de herramientas para cubrir las necesidades de la empresa

Ejemplos: Microsoft Office, OpenOffice, Suite de Google, Oracle, Prezi, …

5. Software de aplicación



5. Software de aplicación

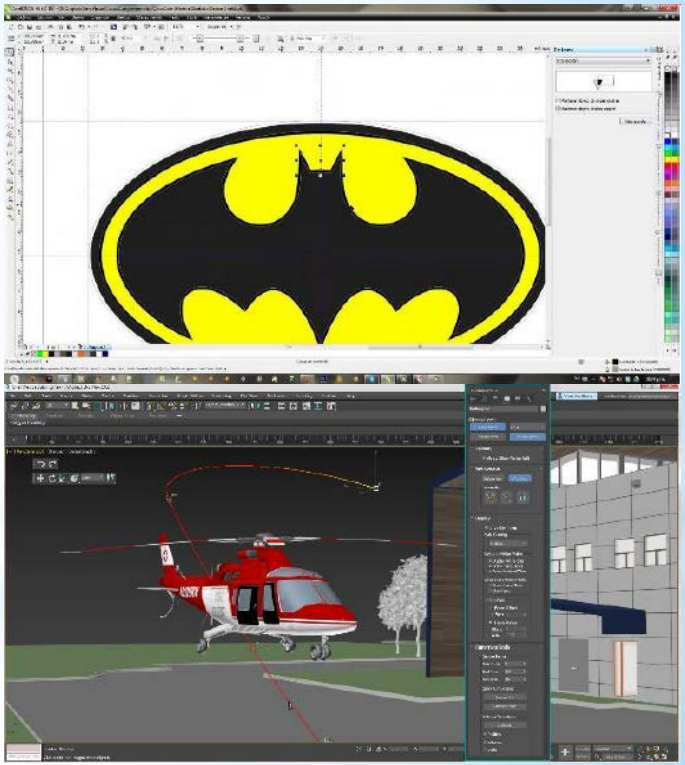
Sonido, video: creación, edición y reproducción de contenidos multimedia.

Ejemplos: Real Player, VLC, CoolEdit, Audacity, Adobe Audition, Audio Cutter, Premiere, Filmora, Sony Vegas, …

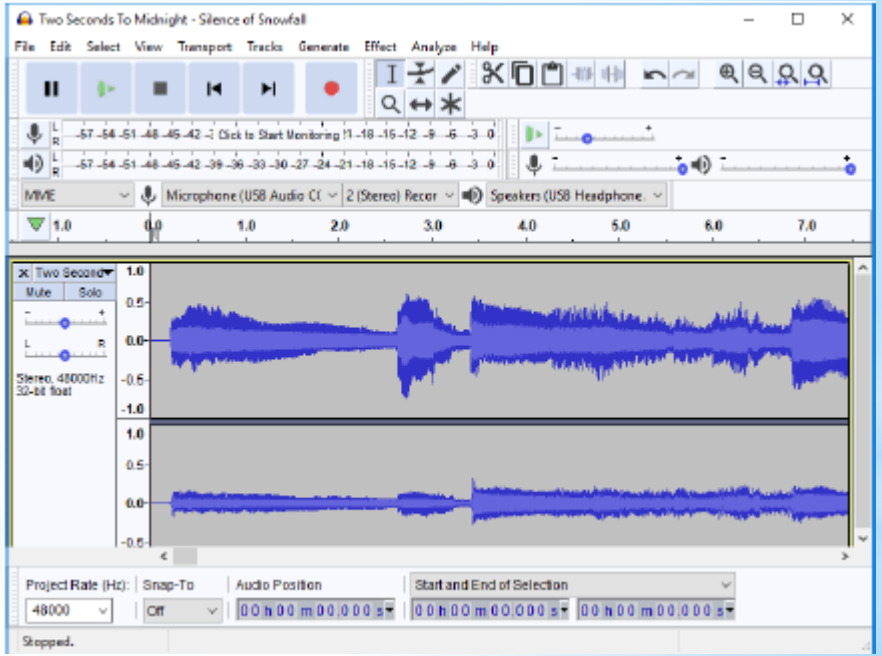
Gráficos: visualizadores, editores vectoriales, editores retoque fotográfico, programas CAD, Diseño 3D

Ejemplos: Photoshop, GIMP, Corel Draw, Freehand, Autocad, Autodesk 3ds Max, …

5. Software de aplicación



5. Software de aplicación



5. Software de aplicación

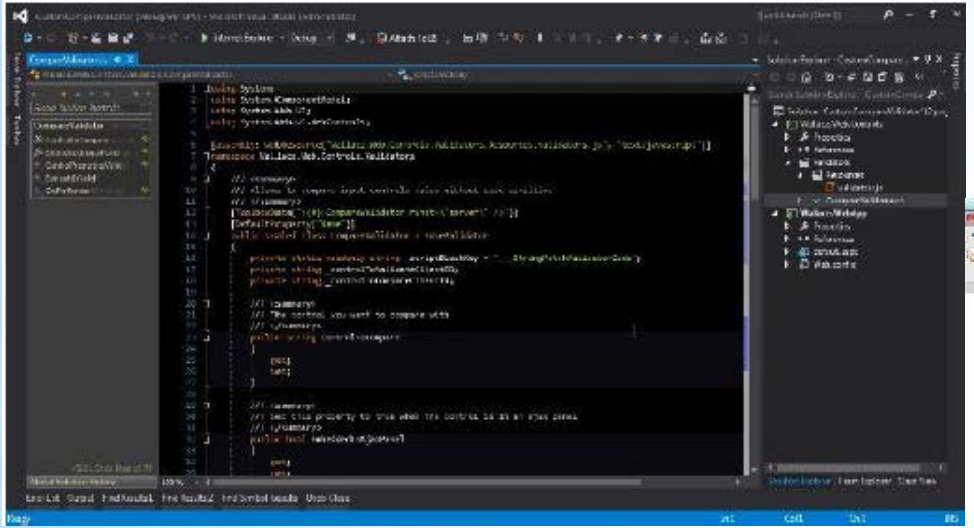
Programación: permiten a los programadores desarrollar programas informáticos mediante lenguajes de programación.

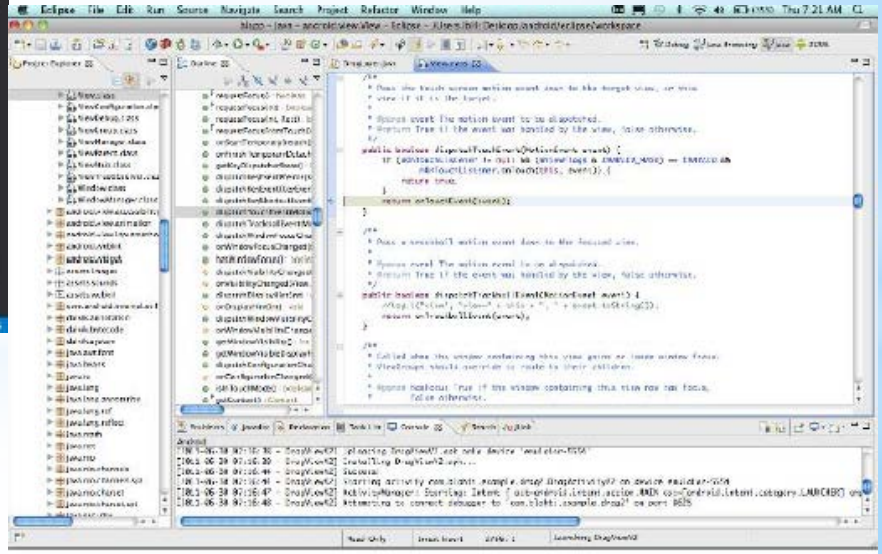
Ejemplos: Visual C++, Eclipse, NetBeans, Visual Studio, …

Entretenimiento: principalmente videojuegos.

Hay plataformas específicas para su creación: GameMaker Studio, Unity, …

5. Software de aplicación





5. Software de aplicación

Productividad y negocios: permiten mejorar la productividad de un negocio, gestionar contactos, llevar la contabilidad, gestionar proyectos, …

Ejemplos: Contaplus, Microsoft Project, TPV, …

Servicios de Internet: navegadores, clientes de email, FTP, mensajería, VoIP, aplicaciones P2P, firewalls, antivirus,…

Ejemplos: Chrome, Firefox, Thunderbird, BitTorrent, Comodo Firewall, Zone Alarm, …

5. Software de aplicación

Software a medida: se desarrolla según las indicaciones del usuario. Necesita más tiempo de desarrollo y es más costoso. Se adapta mejor a las necesidades de una empresa. Suele contener errores que será necesario depurar.

Software en tiempo real: relacionado con el mundo exterior, debe responder en un tiempo crítico.

Ejemplos: gestión de planes de vuelo de compañías aéreas, controladores de procesos industriales, sistemas médicos, …

5. Software de aplicación

Software empotrado: reside en memoria de solo lectura y se utiliza para controlar productos y sistemas concretos. Tiene funciones muy limitadas

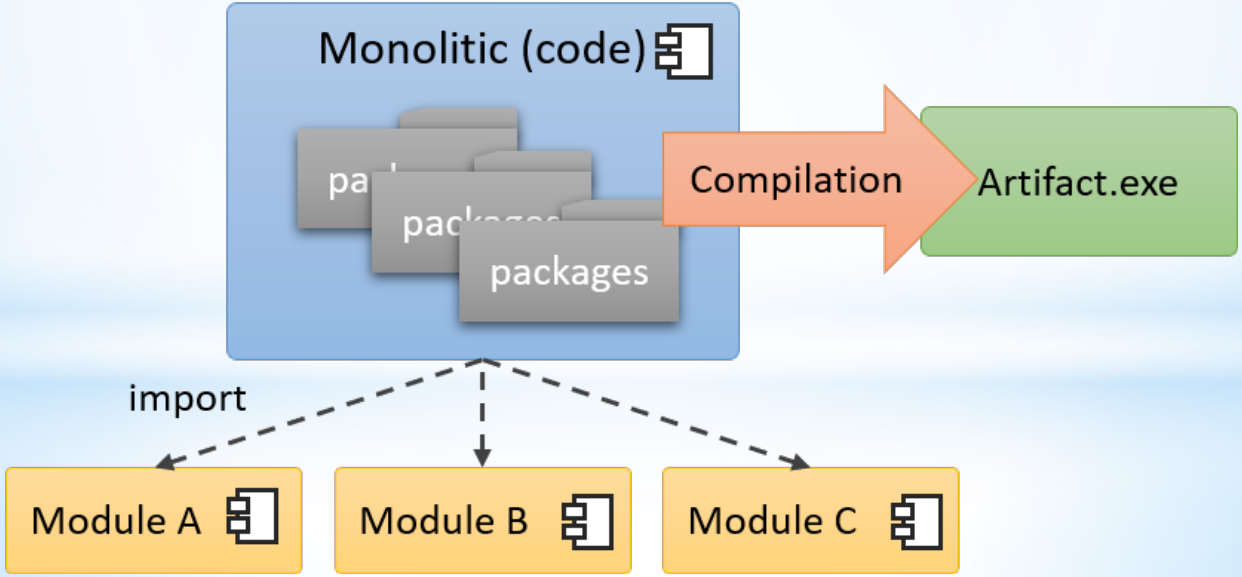
Ejemplos: funciones digitales de un automóvil, teclas del microondas, sistemas GPS, cajeros automáticos, …

Software de Inteligencia Artificial: usa algoritmos no numéricos para resolver problemas complejos.

Ejemplos: sistemas expertos, machine learning

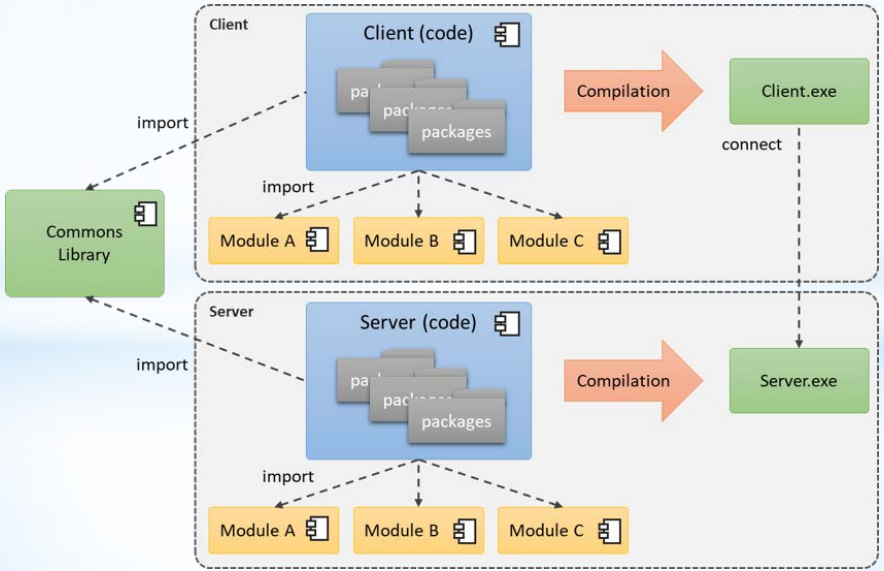
6. Arquitecturas del software

- Arquitectura monolítica: software estructurado en componentes funcionales muy acoplados



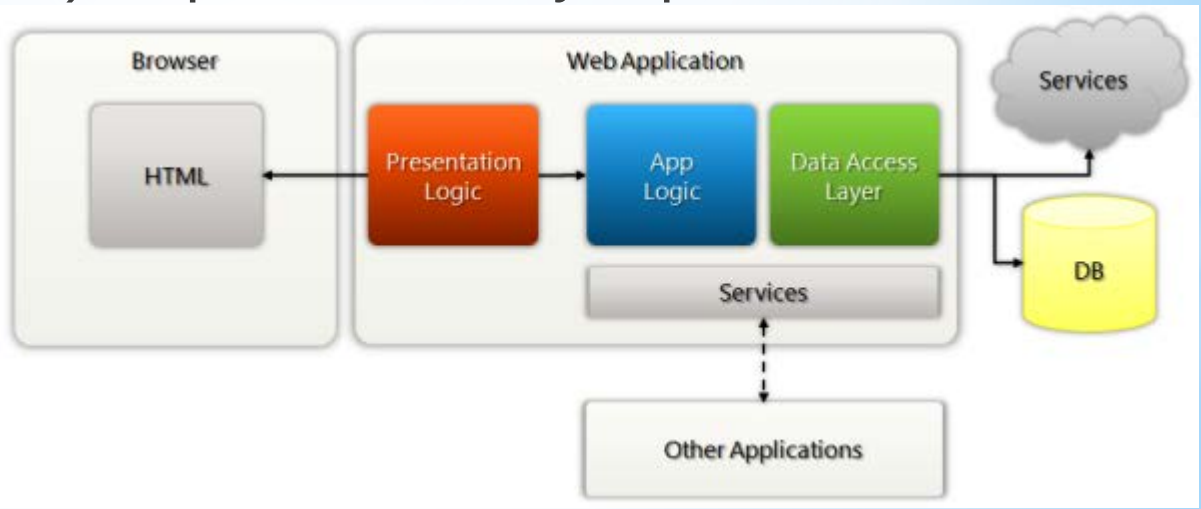
6. Arquitecturas del software

- Arquitectura Cliente - Servidor: el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones



6. Arquitecturas del software

- Arquitectura de tres niveles: tipo concreto de arquitectura cliente-servidor donde la carga se divide en capas que se relacionan únicamente con la inmediata. Capa de presentación (interfaz), capa de cálculo y capa de almacenamiento



7. Instalación y configuración de aplicaciones

Cada vez se instala menos software ya que se usan muchas aplicaciones web.

Se consiguen aplicaciones también en tiendas virtuales:

- AppStore
- Google Play

7. Instalación y configuración de aplicaciones

Las aplicaciones se pueden instalar de distinta manera:

- Copia directa
- Instalador
- Sistema o gestor de paquetes (dpkg de Debian, rpm de Red Hat, tgz de Slackware). Los paquetes se almacenan en repositorios

7. Instalación y configuración de aplicaciones

Otros conceptos:

- Instalación atendida: el usuario interactúa con el programa de instalación indicando idioma, clave, carpetas para la instalación, configuración, …
- Instalación desatendida: todo está completamente automatizado y las respuestas se leen desde un archivo previamente preparado con todas las respuestas

7. Instalación y configuración de aplicaciones

Otros conceptos:

- Instalación portable: se ejecuta sin necesidad de un proceso previo de instalación.
- Instalación no portable: necesita instalación

7. Instalación y configuración de aplicaciones

La configuración de las aplicaciones es el conjunto de acciones que determina el valor de algunas variables de la aplicación en su ejecución.

Hay principalmente dos tipos de configuraciones:

- Configuración predeterminada: es la más general y viene con las opciones más habituales o genéricas.
- Configuración personalizada: determinada específicamente por el usuario. Suele incluir idioma, rutas de instalación, componentes a instalar, …

7. Instalación y configuración de aplicaciones

La desinstalación depende del modo de instalación utilizado.

- En el caso de copia directa, se arrastra la carpeta directamente a la papelera.
- Cuando se ha utilizado un instalador es necesario recurrir a un desinstalador para que elimine todos los archivos, librerías, cambios en el registro, …
- Si se usan gestores de paquetes hay que recurrir a ellos y marcar el paquete a desinstalar.

7. Instalación y configuración de aplicaciones

Inventariado del software

Recopilar en un informe el software que se encuentra instalado en un equipo.

Datos: nombre programa, versión, desarrollador, licencia, espacio en disco, …

8. Testing de aplicaciones

- Conjunto de procesos que permiten verificar y validar la calidad del software identificando:
  - Errores de diseño - el programa no hace lo que se pedía
  - Errores de implementación - el programa hace lo que debe pero de forma incorrecta
- Lo habitual es que el proceso de pruebas se realice desde que empieza el desarrollo hasta que finaliza.
- A veces se sacrifica esta fase para cumplir con plazas o costes.
- Interesante definir un plan de pruebas

8. Testing de aplicaciones

- Tester: persona que realiza prueba, forzando a veces el sistema para ver cómo reacciona. Se recomienda que sea alguien ajeno al desarrollo
- Es fundamental que las pruebas se realicen de forma metódica, probando todos los elementos, no al azar.
- Probar en un entorno controlado, no en producción.

El testing puede probar la presencia de errores pero no la ausencia de ellos

8. Testing de aplicaciones

Tipos de pruebas

- Caja negra o blanca
- Integración incremental
- Pruebas de aceptación
- Pruebas de carga
- Pruebas de instalación / desintalación
- Generación de versiones beta

9. Evaluación y comparativa de aplicaciones

Es recomendable ir probando cada uno de los productos o modelos generados, y verificando que se cumple con lo esperado.

Estas pruebas las realizarán tanto desarrolladores como clientes.

El objetivo es asegurar la calidad del software. Ésta debe tener en cuenta no solo la adecuación a los requisitos iniciales sino también el rendimiento de la aplicación.

9. Evaluación y comparativa de aplicaciones

Rendimiento

- Orientado al usuario (tiempo de respuesta) y al sistema (uso de la CPU)
- Tiempo de respuesta, capacidad de ejecución, carga de trabajo, pruebas de estrés, …
- Pruebas de carga que observan el comportamiento de la aplicación en una situación similar a la real.