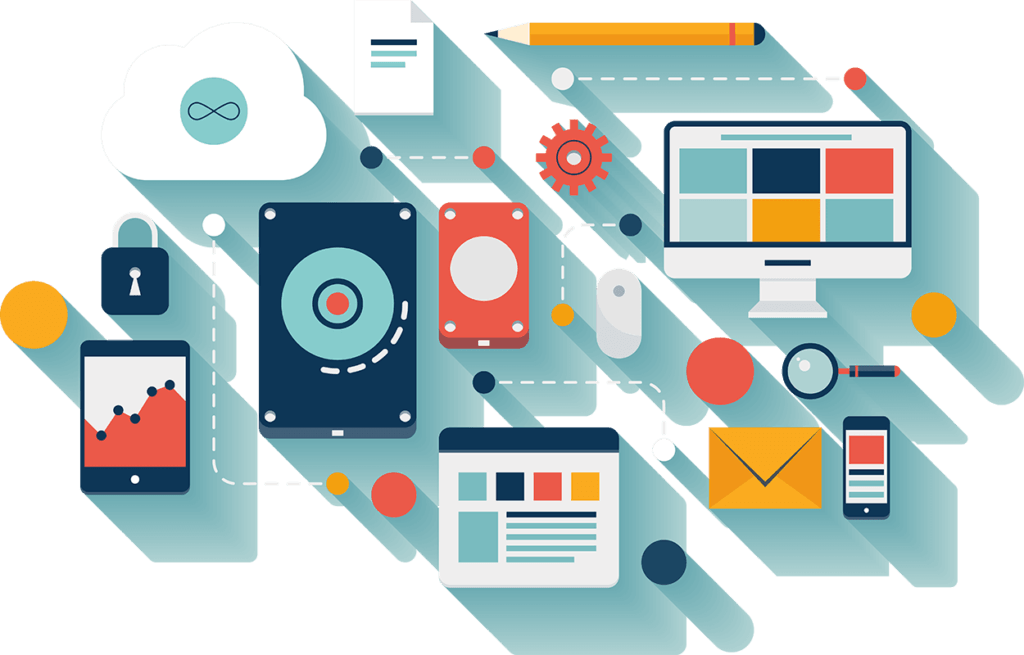


1. **INTRODUCCIÓN.**

La ingeniería del software tiene varias definiciones entre las cuales destaca el término mantenimiento de software. La definición oficial sería; la ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software. Este trabajo consiste en la creación de software, desarrollo y mantenimiento del mismo.

**2. HARDWARE Y SOFTWARE.**

****

**2.1. ¿Qué es un ordenador?**

Un ordenador es una máquina capaz de ser programada para desempeñar distintas funciones utilizando una serie de instrucciones ordenadas. Estos constan básicamente de dos componentes: el **Hardware** y el **Software**.

**2.2. Partes de un ordenador.**

* Partes externas: Se sitúan fuera del ordenador (Caja, periféricos…)
* Partes internas (CPU, Disco duro…)

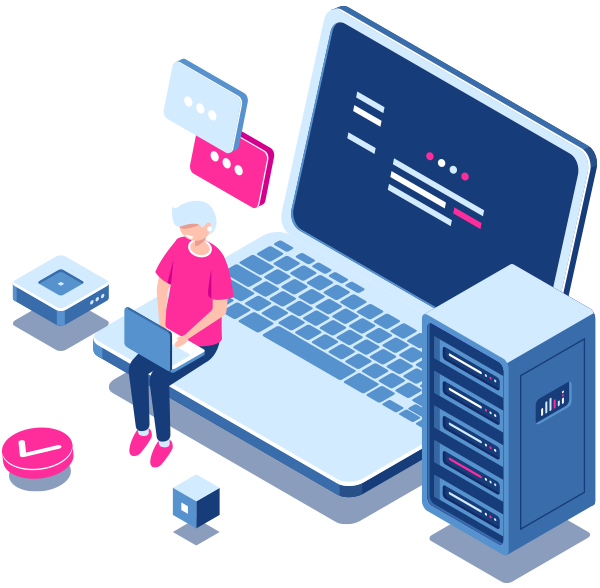
**2.3. Hardware**

El hardware es todo aquello de lo que consta el ordenador que se puede tocar. La parte física del mismo.

****

**2.4. Software.**

Es la parte lógica de un ordenador, todo lo que no se puede tocar o no es físico.



**2.4.2 Tipos de software.**

Entre los distintos tipos de software encontramos: de sistema (controladores, sistemas operativos…), de programación (editores de texto, compiladores, intérpretes...), de aplicación (bases de datos, videojuegos, aplicaciones ofimáticas...).

**2.5. Sistemas Operativos.**

Es el software principal de un ordenador, el cual gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto a los restantes.

**2.6. Funcionalidades del SO.**

* **Gestión de los procesos:** Su función es la de generar o destruir procesos, pararlos, reanudarlos y ayudar con los mecanismos para el correcto funcionamiento de la máquina.
* **Gestión de la memoria principal:** Esta función sirve para el control de datos almacenados, mediante esta función del sistema operativo conseguiremos que los datos ni se saturen ni se pierdan.
* **Gestión del almacenamiento secundario:**  Los datos al poder perderse muy fácilmente, es necesario tener un segundo almacenamiento en caso de que haya un fallo en el almacenamiento principal y estos datos deban ser guardados.
* **Gestión del sistema de entrada y salida:** El sistema operativo se encarga de buscar información en la red sobre los dispositivos que conectamos a el, y asi tener una respuesta inmediata en nuestros dispositivos externo como podrían ser unos auriculares.
* **Registro del sistema de archivos:** Los sistemas operativos tienen la función de permitir crear o destruir archivos también como permitirnos el acceso a ellos.
* **Seguridad:** Hay muchos virus que pueden llegar a dañar nuestro sistema, es por ello que el sistema operativo tiene esta función la cual no permite que esto suceda.
* **Comunicación entre elementos y aplicaciones:** El sistema operativo estará siempre en contacto con los componentes de la máquina y las aplicaciones.
* **Informar del estado del sistema:** El sistema operativo te avisa de cualquier actualización o acción que necesites realizar en tu sistema.
* **Gestión de recursos:** Administra los recursos del sistema para poder poder ejercer como administrador.
* **Administración de usuarios:** Este, como bien dice su nombre, es el encargado de la gestión de los usuarios creados en el equipo, tanto como si hay un solo usuarios como si hay varios.

**2.7. Licencias Software.**

Una licencia de software es un contrato entre el titular de los derechos de distribución y el usuario consumidor del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

**Pregunta: ¿Cuál crees que es el sistema operativo que más se usa en el mundo?**

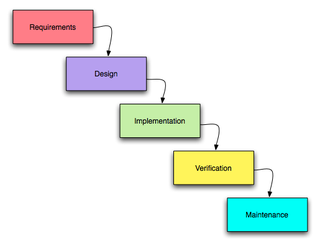
Android, dado que cada vez es más normal el uso de teléfonos móviles, los cuales funcionan con dicho sistema.

3. Desarrollo de software:

* Definición: Mediante el uso de uno o varios lenguajes de programación, estructurar una serie de órdenes las cuales serán interpretadas por el dispositivo. Existen distintos tipos de modelos de desarrollo, cascada, espiral…
* Etapas: Este proceso de creación de software consta de diferentes partes, en las que se van desempeñando distintas funciones que completan el trabajo. A continuación, analizaremos las siguientes fases de este proceso.

1. **Análisis de requisitos:** Se trata de el proceso en el que los ingenieros o desarrolladores se reúnen con el cliente para analizar el resultado esperado. En esta parte, se realiza la captura, el análisis y la especificación de requisitos necesarios para la realización del trabajo especificado. Esta parte del desarrollo de software puede estar reglada por varios estándares (CMM-I).
2. **Diseño y arquitectura funcional:** Esta parte consiste en analizar de forma superficial como funcionará el software en cuestión. En ella, se miran distintos ámbitos como el hardware a utilizar, la red… En definitiva, describe cómo llegará a satisfacer los requisitos del cliente este software.
3. **Programación:** La parte más obvia del proyecto. En ella, se reduce a código el diseño realizado en todas las anteriores fases.
4. **Documentación:** En este apartado, se documenta todo el proyecto realizado, dejando constancia del mismo mediante diagramas, manuales… Esto se realiza por una posible ampliación en un futuro o cualquier otra acción que requiera de información sobre ello.
5. **Mantenimiento:** Consiste en mantener el trabajo realizado para solucionar errores, añadir nuevos requisitos…

* Modelos de desarrollo de software: Los procesos de desarrollo de software, pueden estar ceñidos a unos marcos de trabajo llamados modelos de desarrollo de software, que facilitan el elaboramiento de dicho software. Existen diversos modelos de desarrollo distintos:
* **Modelo en cascada:** En este modelo se siguen rigurosamente las etapas definidas en la explicación anterior. En él, no se puede comenzar una de ellas sin haber acabado la otra



* **Modelo de prototipos:** En este modelo, primeramente se le presenta un producto al cliente, el cual mediante críticas constructivas ayuda a mejorar el posterior resultado final. La mayor ventaja de este modelo es que el cliente ve de forma más cercana una posible simulación de su ansiado resultado.
* **Modelo en espiral:** Es un ciclo de vida del software en el que cada iteración representa un grupo de actividades.
* **Modelo iterativo:** Este consiste en construir las secciones de una forma reducida para luego ir uniéndolas. Con este método conseguimos resolver los problemas de una forma fácil y directa ya que son más fáciles encontrarlos.
* **Modelo rápido:** Se enfoca en la toma de decisiones en los proyectos de desarrollo incremental.



* **Modelo de desarrollo concurrente:** En este modelo, se describen procesos que suceden simultáneamente en una aplicación, y son realizados de manera concurrente, como su propio nombre indica, por más de un equipo de desarrolladores independiente.
* **Modelo Big Bang:** Se utilizan una gran cantidad de recursos en él, tales como tiempo, dinero y esfuerzo; y juntando estos tres, se podría conseguir un mejor producto de software.
* **Codificación y corrección:** Este consiste en comenzar produciendo el código de forma inmediata. Después de esto se empieza a probar el software y se eliminan los errores.
* **Orientado a la reutilización:** La reutilización de software es un proceso donde se recurre al uso de activos de software en las especificaciones de análisis, diseños, implementación y pruebas de una aplicación o sistemas de software.
* Herramientas de apoyo en el desarrollo de software:

Recopilación de datos: Podemos distinguir entre las fuentes primarias(información

obtenida de la realidad) y las secundarias(obtenida de investigaciones

pasadas).Para ello una técnica es la observación:

1. Determinar el objeto

2. Determinar los objetivos

3.Determinar la forma

4. Registrar la forma

5. Analizar e interpretar

6.Elaborar conclusiones.

Otra herramienta es la entrevista. Podemos encontrar varios tipos(estructurada, no

estructurada y mixta). Por otro lado, podemos encontrar las encuesta para obtener

información.Tenemos encuestas exhaustivas, directas e indirectas y encuestas sobre

hechos y encuesta de opinión.

Técnica costo beneficio: consiste en enumerar y comparar el coste neto de un

programa con los beneficios que van a surgir de ellos. Los costes se pueden dividir en

**4 Lenguaje de programación**

partes. La primera es el precio del software(licencias y soporte). La segunda es la infraestructura(componentes hardware y software). La tercera es la implantación(instalación y puesta en funcionamiento) y la última es el entrenamiento(dirigido a los usuarios de la aplicación). Por otro lado están los beneficios que se pueden clasificar de esta manera(reducción de tiempo y recursos). Si los beneficios son mayores a los costes el desarrollo será viable. Otra de las herramientas es el CASE que significa ingenieria de software asistida por ordenador. Estas ayudan a mejorar la calidad.4GL(lenguajes de cuartas generación. Mejoran la productividad.

**4.1. ¿Qué es un programa?**

Es una secuencia de instrucciones escritas para realizar una tarea específica.

**4.2. Definición de Lenguaje de Programación.**

El lenguaje de programación es la forma de proporcionar una serie de instrucciones que permiten a un programador escribir secuencias de órdenes

**4.3. Elementos.**

La sintaxis de un lenguaje define los elementos de dicho lenguaje y cómo se combinan para formar un programa. Los elementos típicos de cualquier lenguaje son los siguientes:

1. - Identificadores
2. - Tipos de datos
3. - Palabras reservadas
4. - Sentencias
5. - Bloques de código
6. - Comentarios
7. - Expresiones
8. - Operadores

**4.4. Clasificación según nivel de abstracción.**

**Bajo Nivel:** La programación se realiza tomando muy en cuenta las características del procesador.

* **El lenguaje maquina:** Este lenguaje ordena al dispositivo las operaciones fundamentales para su funcionamiento.
* **El lenguaje ensamblador:** Es un derivado del lenguaje máquina y está formado por abreviaturas de letras y números llamadas mnemotécnicos.

**Alto Nivel:** Los lenguajes de alto nivel manejan conceptos, tipos de datos, entre otros de una manera similar al pensamiento humano ignorando (abstrayéndose) en parte, del funcionamiento del dispositivo.

**4.5. Clasificación según la forma de ejecución.**

Lenguajes compilados:

Un programa que se escribe en un lenguaje de alto nivel tiene que traducirse a un código que pueda utilizar la máquina. Los programas traductores que pueden realizar esta operación se llaman compiladores. Al usar un lenguaje compilado, para ejecutar el programa primero hay que compilarlo, y si hay errores de programación no podemos compilarlo.

Lenguajes interpretados:

En vez de traducir el programa fuente y grabar en forma permanente el código, el programador sólo carga el programa fuente en la computadora junto con los datos que se van a procesar. A continuación, un programa intérprete, convierte cada proposición del programa fuente en lenguaje de máquina conforme vaya siendo necesario durante el proceso de los datos. Es decir, el programa no se compila, sino que se va traduciendo a tiempo real mientras se ejecuta.

**4.6. Clasificación según la técnica de programación**

* Lenguaje imperativo: Las instrucciones se ejecutan unas tras otras, de manera secuencial, salvo cuando se encuentran estructuras de control condicionales o bucles.
* Lenguaje declarativo: Más en las matemáticas y en la lógica que los lenguajes imperativos, más cercanos estos al razonamiento humano. Los lenguajes declarativos no dicen cómo hacer una cosa, sino, más bien, qué cosa hacer
* Lenguaje de programación orientado a objetos: Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.
* Lenguaje orientado al problema: son aquellos lenguajes específicos para gestión.
* Lenguaje de programación natural: son los nuevos lenguajes que pretender aproximar el diseño y la construcción de programas al lenguaje de las personas.

4.7. Obtención de código ejecutable.

El código ejecutable corresponde a unidades de programas. Donde la computadora puede realizar las instrucciones compiladas que tendrán enlazadas una o varias bibliotecas.

4.7.1. Tipos de código.

* Código fuente: Es un conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar un programa.
* Código objeto: Código generado por un compilador o un ensamblador traducido por un código fuente de un programa. Casi siempre este término se refiere al código máquina que puede ejecutarse directamente en la unidad central de proceso del sistema.
* Código binario: El código binario es el sistema numérico usado para la representación de textos, o procesadores de instrucciones de computadora utilizando el sistema binario
* Código abierto: Código abierto es la expresión con la que se conoce al software distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

4.7.2. Ejecución de un programa.

A un programa en ejecución se le suele llamar también proceso.

El proceso de ejecución de un programa escrito en un lenguaje de programación y mediante un compilador tiene los siguientes pasos:

1. Escritura del programa fuente con un editor y guardarlo en un dispositivo de almacenamiento.

2. Introducir el programa fuente en memoria.

3. Compilar el programa con el compilador.

4. Verificar y corregir errores de compilación.

5. Obtención del programa objeto

6. El enlazador obtiene el programa ejecutable.

7. Se ejecuta el programa y, si no existen errores, se tendrá la salida del programa.

**Pregunta: Si queremos construir una aplicación pequeña, y se prevé que no sufrirá grandes cambios durante su vida, ¿qué modelo de ciclo de vida sería el más recomendable? ¿Por qué?**

Para construir una aplicación pequeña lo más recomendable sería saltarse el paso de mantenimiento ya que si no va a sufrir grandes cambios este modelo será prácticamente inexistente y nos ahorraremos muchos gastos en mantenimiento. Además, también podríamos saltarnos el diseño y la arquitectura ya que al ser un programa pequeño la programación no será complicada y no nos complicamos haciendo algoritmos o mapas de entidad-relación.

**Lenguaje de programación**

Un lenguaje de programación es un programa destinado a crear otros programas. Este lenguaje está compuesto por símbolos y normas sintácticas (por eso se llama lenguaje). Se expresan en forma de instrucciones y relaciones lógicas, mediante las cuales se construye el código fuente de una aplicación o pieza de software determinado.

**Elementos**

La sintaxis de un lenguaje define los elementos de dicho lenguaje y cómo se combinan para crear un programa.

1. Tipos de datos. Hay varios tipos, numéricos y boolean (verdadero/falso). Por otro lado los numéricos se clasifican en enteros(byte,short,int,long y char) y reales(float y double)
2. Variables y vectores: su función es asignar un espacio de memoria para luego poder almacenar datos específicos. Son contenedores(cajas) de datos y por eso se diferencian según el tipo de datos que puedan guardar. En muchos de los programas se requiere especificar el tipo de variable que vamos a utilizar [si es una variable numérica(decimal,entera…)un único carácter (letra,número…) o si contiene cadenas de texto(string)].
3. Condicionales. Estas son utilizadas cuando queremos indicar que para que algo pase algo ha tenido que ocurrir antes de esa acción.

**Tipos de lenguaje**

Podemos clasificarlos de varias maneras:

1- según su nivel de abstracción

-Lenguajes de bajo nivel: Estos están diseñados para un hardware específico. No se pueden exportar a otros ordenadores. Es decir, son perfectos para el sistema que fueron diseñados pero no se pueden utilizar en ningún otro.

-Lenguajes de alto nivel: Aspiran a ser un lenguaje más universal, es decir, a diferencia de los de bajo nivel estos pueden emplearse en diferentes hardwares, en diferentes sistemas. En este apartado también podemos encontrar dos tipos:

Lenguajes de propósito específico: Son creados para un propósito concreto para facilitar el desarrollo a partir de las simulaciones de sistemas complejos.Su finalidad es resolver problemas determinados(problemas estadísticos, programación de máquinas, simulación de sistemas..

Lenguajes de propósito general: Son aquellos que pueden ser usados para varios propòsitos( acceso a bases de datos, comunicación entre dispositivos, captura de datos, cálculos matemáticos, diseño de imágenes y páginas, comunicación entre computadoras…)

2- según su modo:

Lenguajes imperativos: como su propio nombre indica son menos flexibles. Estos lenguajes funcionan o se programan mediante órdenes condicionales.

Lenguajes funcionales: También conocidos como procedimentales. Estos lenguajes programan mediante funciones que son invocadas conforme a la entrada recibida, que a su vez son resultado de otras funciones.

Lenguajes de programación más usados:

1.HTML: HTML significa HyperText Markup Language («lenguaje de marcas de hipertexto»). Este lenguaje lo utilizaremos para crear páginas web mediante etiquetas que sirven para definir el texto y otro tipo de cosas que vemos en las páginas web (márgenes, imagenes, videos,etc).

2.SQL: creado para realizar consultas a bases de datos. Structured Query Language, lenguaje de consulta estructurada.

3. Java: El más conocido. Sobre todo es utilizado para juegos y aplicaciones.

4.Python: Es un lenguaje muy utilizado, es conocido por ser multiplataforma, es decir soporta varios tipos de programación.

3-Clasificación según la forma de ejecución.

1.Lenguajes compilados. Traducen el código fuente del programa a código máquina(basados en 0s y 1s) u objeto.

2.Lenguajes interpretados. El código fuente es necesario para ejecutar el programa. Lo ejecutan linea a linea.

4- Según su técnica de programación:

1.Lenguajes imperativos: Indican exactamente el orden de ejecución del programa.

2.Lenguajes declarativos: Expresan lo que el programa debe conseguir sin prescribir cómo hacerlo.

3.Lenguajes funcionales: en éstos las funciones ya están pre-definidas.

4.Lenguajes lógicos: Usan las matemáticas para expresar las tareas mediante su orden lógico.

5.Lenguajes orientado a objetos: Crean un sistema de clases y objetos siguiendo el esquema del mundo real para definir los objetos,acciones y forma de comunicar entre objetos.(java)

**Obtención de código ejecutable**

El código ejecutable es un lenguaje creado por una máquina que corresponde a las unidades de programas, donde el computador realiza las instrucciones compiladas por medio del enlazador y el compilador.

1.Tipos de código

1.Código ejecutable: Reúne distintos códigos generados por los programadores. Estos componen el programa final. El código es ejecutable por los usuarios del sistema.

2.Código fuente: está formado por un grupo de instrucciones entendibles por el programador que crea el programa. El código es almacenado en un fichero de texto (Wordpad de windows por ejemplo).Un ejemplo de ello son Python,C++ o Java

3.Código objeto: Está formado por un conjunto de objetos e instrucciones que están escritos en un lenguaje que entiende el ordenador como el binario o el código máquina. Es un fragmento del programa final y es específico para la plataforma de ejecución.

2. Ejecución de un programa: Se suele ejecutar mediante un compilador y tiene los siguientes pasos:

1. Escribir el programa y guardarlo en un dispositivo de almacenamiento.
2. Introducir el programa fuente en memoria
3. Compilar el programa.
4. Verificar y corregir los errores causados en la compilación.
5. Obtención del programa objeto.
6. El linker (enlazador) obtiene el programa ejecutable.
7. Se ejecuta el programa, si no se encuentran errores se obtendrá la salida del programa.

**5. ENTORNOS DE DESARROLLO (IDE).**

También conocido como Entorno de Desarrollo Integrado o Integrated Development Environment (IDE) es un entorno de desarrollo y una plataforma diseñada para facilitar la labor del programador de software.

Puede servir para; buscar y reemplazar códigos, colorear la sintaxis para una mejor legibilidad, ejecutar en modo debug, autocompletar códigos, integrar sistemas de control de versiones, diseñar proyectos a partir de plantillas, insertar trozos de códigos o refactorizar códigos.

**5.1. Componentes de un IDE.**

**5.1.1. Editor de texto.**

Un editor de texto es un programa donde puedes crear y modificar archivos digitales de texto sin formatos. El programa lee el archivo e interpreta los bytes según el código que use el editor de texto (por ejemplo: ASCII o UTF-8).

**5.1.2. Compilador.**

En palabras simples un compilador sería un traductor de lenguaje, que traduciría nuestro lenguaje de programación a un lenguaje más simple.

**5.1.3. Intérprete.**

Un intérprete es un programa de computadora que ejecuta los códigos que ha escrito línea por línea sin compilación previa de código en lenguaje máquina.

**5.1.4. Depurador o Debugger:**

Un depurador (Debugger) es un programa informático creado con el fin de analizar, encontrar errores. Normalmente, los depuradores suelen estar dentro de los entornos de desarrollo integrados.

**5.1.5. Control de versiones.**

Es un sistema que registra los cambios realizados en un archivo (o varios) a lo largo del tiempo, para así poder recuperar las versiones anteriores en caso de cometer un error.

**5.1.6. Interfaces gráficas de usuario.**

Es un programa informático que utiliza mediante imágenes y objetos gráficos. La utilidad de una GUI (Graphic User Interface) es facilitar la comunicación con el sistema operativo ofreciendo un entorno visual sencillo.

**5.1.7. Refactorización:**

La refactorización es una limpieza de código que no arregle errores ni incorpora funcionalidades.

Además, la refactorización del código solo altera la estructura interna del código sin cambiar el comportamiento externo del programa o de la web. Es decir, a la vista del usuario final no cambia absolutamente nada, pero al programador le facilita muchísimo la vida hacer este proceso, porque así el código se vuelve muchísimo más entendible por una persona humana.

**Objetivos de la refactorización:**

● Facilitar la comprensión del código.

● Cambiar estructura y diseño del código.

● Eliminar código muerto.

● Facilitar el mantenimiento en un futuro.

**6. Máquinas virtuales**

Es un software que puede cargar en su interior un sistema operativo haciendo creer que es un PC de verdad. Una máquina virtual nos permite emular un dispositivo real.

Existen dos tipos de máquinas virtuales: máquinas virtuales de proceso y de sistema.

**6.1. Máquinas virtuales de proceso**

El objetivo de estas máquinas es proporcionar un entorno independiente del sistema operativo para ejecutar una tarea específica.

**6.2. Máquinas virtuales de sistema**

Es el tipo de máquina virtual más común. Este tipo tiene su disco duro, memoria, tarjeta gráfica y el resto de componentes que tendría un ordenador, por ejemplo, pero todas de manera virtual.

El objetivo de estas es poder tener varios sistemas operativos dentro de un mismo sistema, pudiendo ser independientes unos de otros. Se puede realizar más de una tarea en ellos.

**6.3. Ventajas y desventajas de las máquinas virtuales:**

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Posibilidad de instalar varios sistemas operativos dentro de una misma máquina física. | Cuando hay muchas máquinas virtuales ejecutándose al mismo tiempo en una misma máquina física estas pueden presentar un rendimiento inestable. |
| La máquina virtual puede ofrecer una arquitectura diferente a la máquina real | La máquina virtual no es tan eficiente como una verdadera a la hora de acceder al hardware |
| Mantenimiento sencillo. | Precio |
| Centralizar la administración de red | Complejidad |
| Reducir la dependencia del hardware |
| Ideal para experimentar |

# **Preguntas:**

1. **¿Qué es el modelo iterativo?**

Este consiste en construir las secciones de una forma reducida para luego ir uniéndolas. Con este método conseguimos resolver los problemas de una forma fácil y directa ya que son más fáciles encontrarlos.

1. **¿Qué tipos de software existen?**

Existen 2 tipos de software: de sistema (SO, drivers) y de aplicación (videojuegos, ofimática).

1. **Explica los tipos de lenguaje según su nivel de abstracción.**

Alto nivel: estos pueden emplearse en diferentes hardwares, en diferentes sistemas. Estos aspiran a ser universales y podemos encontrar dos subgrupos. Propósito específico, creados para algo concreto y los de propósito general, son estos que se pueden utilizar para varias cosas(acceso a bases de datos, captura de datos,etc). Por otro lado encontramos los lenguajes de bajo nivel. Estos se crean para un hardware específico y no se pueden exportar a otros ordenadores.

1. **¿Qué es la refactorización? Y cuáles son sus objetivos.**

La refactorización es una limpieza de código que no arregle errores ni incorpora funcionalidades.

Objetivos de la refactorización:

Facilitar la comprensión del código.

Cambiar estructura y diseño del código.

Eliminar código muerto.

Facilitar el mantenimiento en un futuro.

1. **¿Desventajas de las máquinas virtuales?**

Cuando hay muchas máquinas virtuales ejecutándose al mismo tiempo en una misma máquina física estas pueden presentar un rendimiento inestable. La máquina virtual no es tan eficiente como una verdadera a la hora de acceder al hardware. Precio. Complejidad.

1. **3 ventajas de las máquinas virtuales**

Mantenimiento sencillo, posibilidad de instalar varios sistemas operativos dentro de una misma máquina física y reducir la dependencia del hardware.

1. **Etapas del desarrollo de software:**

Análisis de requisitos, Diseño, programación, documentación, mantenimiento.

1. **¿Que 3 tipos de código hay?**

* De sistema (controladores, sistemas operativos…)
* De programación (editores de texto, compiladores, intérpretes...)
* De aplicación (bases de datos, videojuegos, aplicaciones ofimáticas...)

1. **Definir que es un compilador y que es un intérprete, dar 2 ejemplos de sus diferencias.**

Lenguajes compilados:

Un programa que se escribe en un lenguaje de alto nivel tiene que traducirse a un código que pueda utilizar la máquina.

Lenguajes interpretados:

En vez de traducir el programa fuente y grabar en forma permanente el código, el programador sólo carga el programa fuente en la computadora junto con los datos que se van a procesar.

1. el intérprete carga el programa fuente en la computadora junto con los datos y el compilador solo traduce de un lenguaje de alto nivel a uno simple para que el ordenador pueda interpretarlo
2. el compilado está hecho de antes y el intérprete es a tiempo real

**10. ¿Qué modelos de desarrollo existen?**

-Modelo en cascada

-Modelo de prototipos

-Modelo en espiral

-Modelo iterativo

-Modelo rápido

-Modelo de desarrollo concurrente

-Modelo Big Bang

-Codificación y corrección

-Orientado a la reutilización

**11. ¿La mesa es un componente interno o externo en esta foto?**

**las dos respuestas estan bien.**