

Contenido

DEFINICIÓN DE SOFTWARE	6
METODOLOGÍA DE DESARROLLO DE SOFTWARE	6
Modelo en espiral	7
Modelo en cascada (waterfall)	8
Prototipado	9
Extreme Programming (XP)	10
ATRIBUTOS DE UN BUEN SOFTWARE	11
ESTIMACIÓN DEL TAMAÑO DEL PROYECTO	11
<i>Métricas para la estimación de tamaño</i>	11
Medidas relacionadas al tamaño	11
Medidas con relación a la función	11
DETERMINACIÓN DE LA VIABILIDAD DE UN PRODUCTO DE SOFTWARE	12
<i>VIABILIDAD TÉCNICA</i>	12
<i>VIABILIDAD ECONÓMICA</i>	12
<i>VIABILIDAD OPERACIONAL</i>	12
DETERMINACIÓN DE LAS NECESIDADES DE HARDWARE Y SOFTWARE	13
CUANDO CREAR SOFTWARE PERSONALIZADO	14
HERRAMIENTA CASE	14
<i>Tecnología de las herramientas CASE</i>	14
CRM	15
<i>¿Cuál es la función de CRM?</i>	15
METODOLOGÍA RUP	15
METODOLOGÍA ERP	16
OPENUP	16
<i>Principios del OpenUP</i>	16
SCRUM	17
EL MODELO COCOMO	17
<i>Modelo Básico</i>	17
<i>Modelo Intermedio</i>	17
<i>Modelo Detallado</i>	17
PERSONAL SOFTWARE PROCESS (PSP)	18

<i>Objetivos</i>	18
METODO AGIL ASD (ADAPTIVE SOFTWARE DEVELOPMENT)	18
<i>CARACTERISITICAS</i>	18
DSDM (METODOLOGÍA ÁGIL)	19
SQA (SOFTWARE QUALITY ASSURANCE O ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE)	19
<i>Definición</i>	19
<i>Propósito</i>	19
<i>Objetivos</i>	19
MODELO CMMI	20
<i>Objetivos</i>	20
DIAGRAMAS DEL UML	20
<i>Diagrama de Clases</i>	20
Clase Abstracta	21
Asociaciones	21
Multiplicidad	21
Composición y Agregación	22
Generalización	22
<i>Diagrama de Objetos</i>	22
Nombre de los objetos	22
Atributos	23
<i>Diagrama de Casos de Uso</i>	23
Sistema.....	23
Casos de Uso	23
Actores	23
<i>Diagrama de Estados</i>	24
<i>Diagrama de Secuencias</i>	24
Rol de la Clase	24
Activación.....	25
Mensajes	25
<i>Diagrama de Actividades</i>	26
ESTÁNDARES DE CALIDAD ISO PARA DESARROLLO DE SOFTWARE	27
<i>El estándar de calidad ISO 9001</i>	27

<i>Para la industria del software los estándares relevantes son:</i>	27
<i>Factores de calidad ISO 9126</i>	27
<i>Estándares ISO/IEC 25000</i>	28
<i>ISO/IEC 2500n – División de Gestión de Calidad</i>	28
<i>ISO/IEC 2501n – División del Modelo de Calidad</i>	28
<i>ISO/IEC 2502n – División de Medición de Calidad</i>	28
<i>NORMAS ISO/IEC</i>	29
<i>ISO 12207 – Modelos de Ciclos de Vida del Software.</i>	29
<i>Norma ISO/IEC 9126</i>	29
<i>Estándar ISO/IEC 14598</i>	29
DISEÑO Y PROGRAMACIÓN WEB	30
AJAX	30
XML	30
<i>Definición</i>	30
<i>¿Para qué sirve XML?</i>	30
<i>Ventajas de XML</i>	30
<i>Características</i>	30
APPSERV (TAILANDES)	31
WAMP	31
XAMPP	31
CÓDIGO PARA DEFINIR UNA TABLA CON FILAS Y COLUMNAS	32
REDES DE COMPUTADORAS	33
<i>Software</i>	33
<i>Sistema operativo de red</i>	33
<i>Software de aplicación</i>	33
<i>Hardware</i>	33
DISPOSITIVOS DE RED	34
<i>Conmutador</i>	34
<i>Router</i>	34
<i>Puente de red</i>	35
<i>Punto de acceso inalámbrico</i>	35
MODELO OSI	35
<i>Capa 7: La capa de aplicación</i>	35

<i>Capa 6: La capa de presentación</i>	36
<i>Capa 5: La capa de sesión</i>	36
<i>Capa 4: La capa de transporte</i>	36
<i>Capa 3: La capa de red</i>	36
<i>Capa 2: La capa de enlace de datos</i>	36
<i>Capa 1: La capa física</i>	36
CABLE DE PARES TRENZADOS	37
<i>Cable directo T568A</i>	37
<i>Cable cruzado</i>	37
ALGORITMOS DE ENCRIPCIÓN SIMÉTRICA Y ASIMÉTRICA	38
ENCRIPCIÓN SIMÉTRICA	38
<i>DES (Data Encryption Standard)</i>	38
<i>IDEA (International Data Encryption Algorithm)</i>	38
<i>AES (Advanced Encryption Standard)</i>	38
ENCRIPCIÓN ASIMÉTRICA	39
<i>Cómo funciona</i>	39
<i>RSA (Rivest, Shamir, Adleman)</i>	39
<i>Algoritmos de autenticación (o hash)</i>	39
<i>SHA-1</i>	39
BASE DE DATOS	40
<i>Campo</i>	40
<i>Columna</i>	40
<i>Dato</i>	40
<i>Tabla</i>	40
<i>Tupla</i>	40
LENGUAJE DE BASE DE DATOS	41
LENGUAJE DE DEFINICIÓN DE DATOS (LDD)	41
LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)	41
<i>Diagrama entidad relación</i>	41
Entidades.....	41
Atributos.....	41
Diamantes	41
Cardinalidad	41

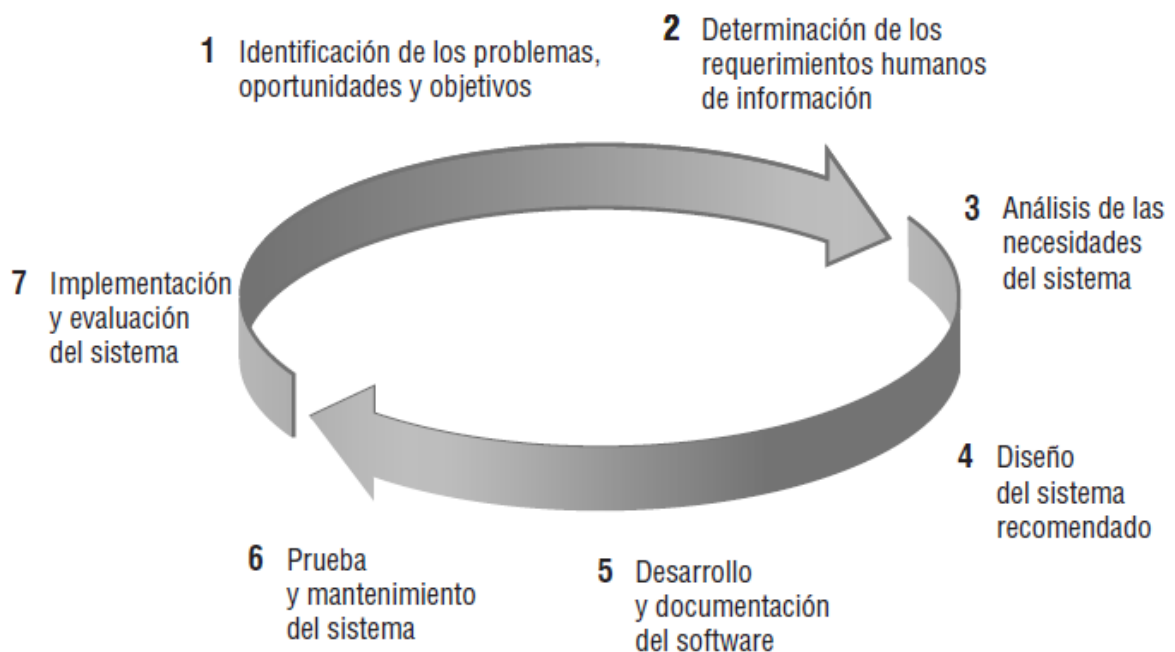
<i>Consultas SQL (SELECT)</i>	42
<i>Consultas SQL (WHERE)</i>	42
<i>Consultas SQL (BETWEEN)</i>	42
<i>Consultas SQL (ORDENACIÓN)</i>	42
<i>INNER JOIN</i>	43
<i>LEFT JOIN</i>	43
<i>RIGHT JOIN</i>	43
SISTEMAS OPERATIVOS	44
SISTEMAS OPERATIVOS PARA SERVIDORES	44
<i>FreeBSD</i>	44
<i>Linux</i>	44
<i>Mac OS X Server</i>	44
<i>Microsoft Servers</i>	45
<i>Solaris</i>	45
<i>Unix</i>	45
PROGRAMACIÓN C++	46
<i>Sentencias IF</i>	46
<i>Sentencias IF-ELSE</i>	46
<i>Sentencias SWITCH</i>	47
<i>Sentencias FOR</i>	48
<i>Sentencias WHILE</i>	48

DEFINICIÓN DE SOFTWARE

Se define como un conjunto de instrucciones que cuando se ejecutan proporcionan las características, funciones y desempeño buscado. Aunque también se pueden definir como estructuras de datos que permiten que los programas manipulen la información de forma adecuada.

METODOLOGÍA DE DESARROLLO DE SOFTWARE

En ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. De manera generalizada el desarrollo de sistemas se vale de 7 fases, aunque no es un estándar que se siga fielmente ya que cada metodología varía en cantidad de fases, pero todas tienen el mismo objetivo.

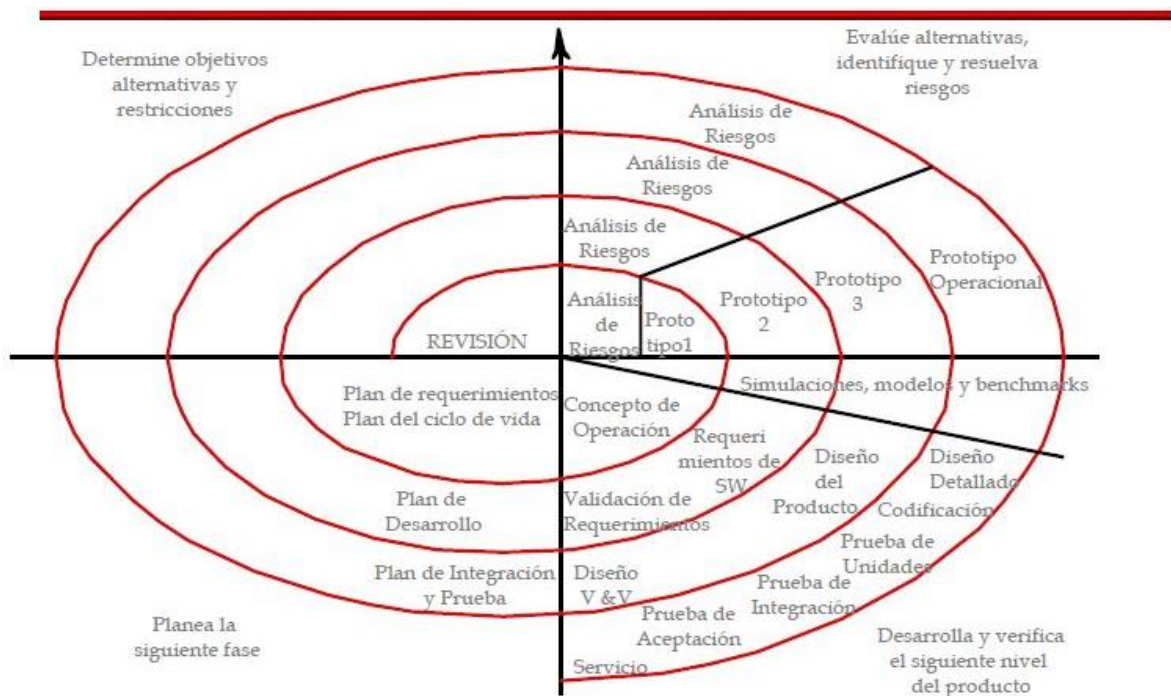


A continuación, se enlistan y explican las más importantes.

Modelo en espiral

Fue propuesto por Boehm en 1988. Este desarrollo representa con cada espiral una fase del proceso de software, Así el ciclo más interno podría referirse a la viabilidad del sistema. Cada ciclo de la espiral se divide en 4 sectores:

- ❖ **Definición de objetivos.** Se definen los objetivos específicos, se identifican las restricciones de los procesos y el producto. También se traza un plan detallado de la gestión, se identifican riesgos y se planean estrategias alternativas.
- ❖ **Evaluación y reducción de riesgos.** Se lleva a cabo un análisis detallado de cada riesgo detectado y se definen los pasos para reducir dichos riesgos.
- ❖ **Desarrollo y validación.** Después de la evaluación de riesgos, se elige el modelo para el desarrollo del sistema.
- ❖ **Planificación.** Finalmente, el proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior a la espiral.

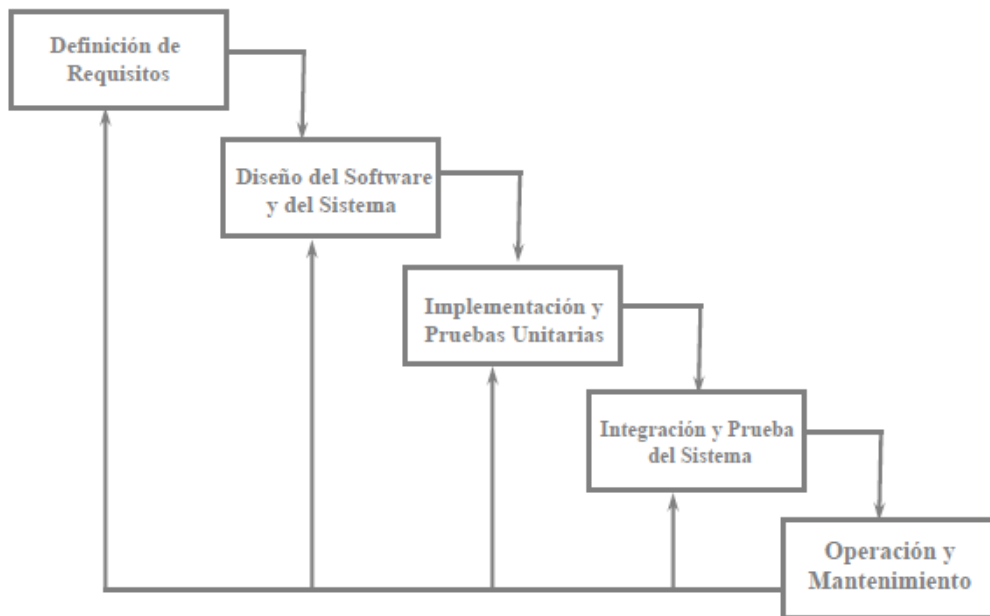


Esta metodología tiene buena visibilidad, cada segmento y cada anillo de la espiral debe producir un documento.

Modelo en cascada (waterfall)

Está basado en la mantenibilidad de línea de ensamblaje, consta de 5 fases:

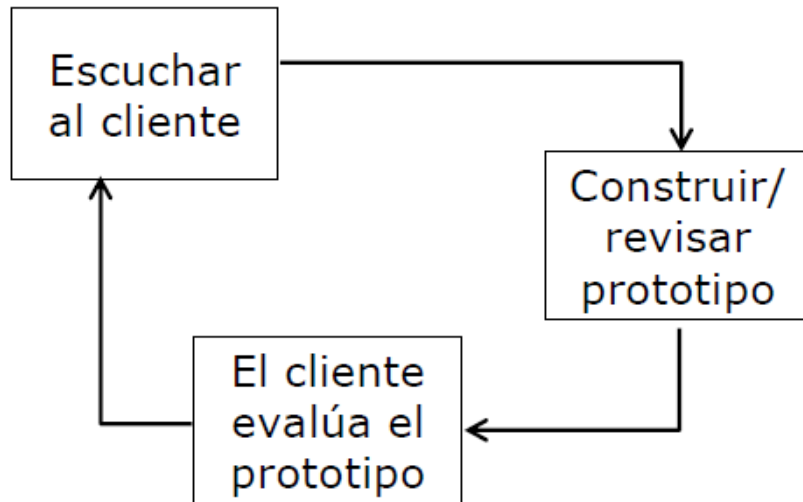
- ✓ **Análisis y definición de requerimientos.** Mediante consultas con los usuarios se definen los servicios, restricciones y metas del sistema.
- ✓ **Diseño del software y del sistema.** Divide los requerimientos en sistemas de software y de hardware. Establece una arquitectura completa del sistema e identifica y describe las abstracciones fundamentales del sistema de software.
- ✓ **Implementación y pruebas unitarias.** Durante esta etapa, el diseño de software se lleva a cabo como un conjunto de unidades, lo cual implica verificar que cada unidad cumpla con sus especificaciones.
- ✓ **Integración y pruebas del sistema.** Los programas o unidades individuales se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software.
- ✓ **Funcionamiento y mantenimiento.** Es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico, el mantenimiento implica corregir errores no descubiertos.



Lo más importante a tomar en cuenta es que ninguna fase puede comenzar hasta que la anterior termine, su principal desventaja es la inflexibilidad.

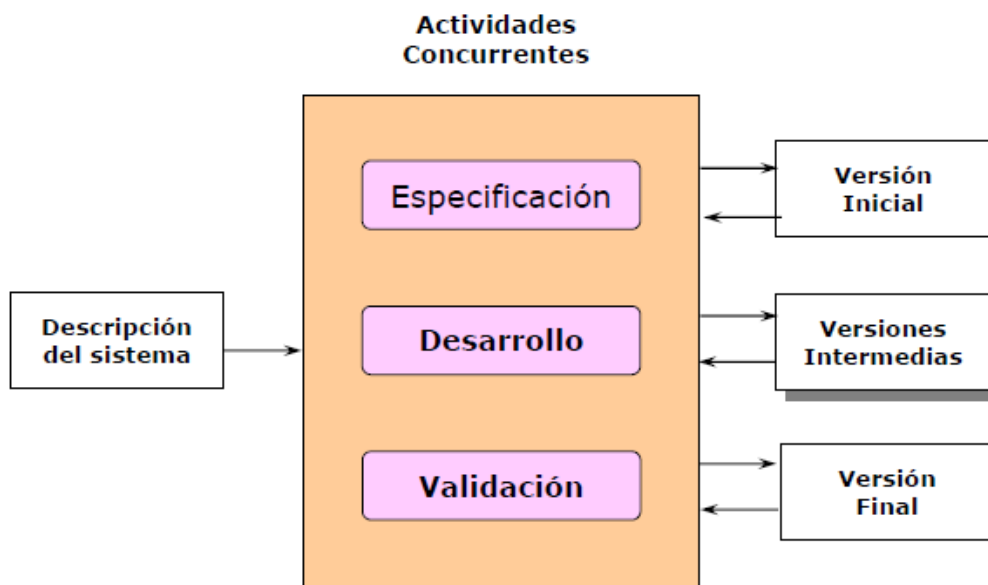
Prototipado

Se usa un prototipo para dar al usuario una idea concreta de lo que va a hacer el sistema. Se aplica cada vez que la rapidez de desarrollo es esencial.



Existen dos tipos de esta:

Prototipado evolutivo: el prototipo inicial se refina progresivamente hasta convertirse en versión final.



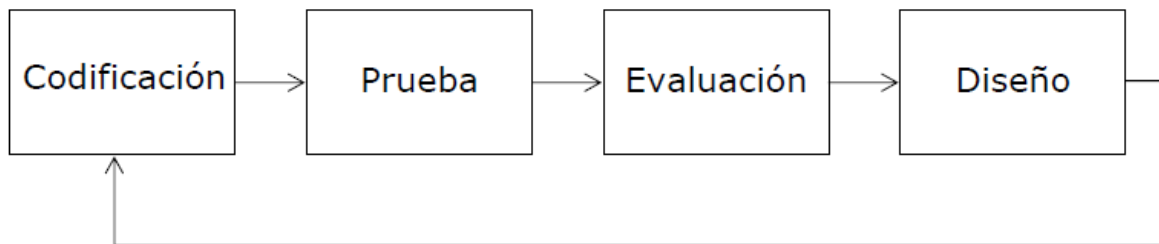
Prototipado desechable: De cada prototipo se extraen ideas buenas que se usan para hacer el siguiente, pero cada prototipo se tira entero.

Extreme Programming (XP)

Un modo ligero, eficiente, de bajo riesgo, flexible, predecible, científico y fácil para producir software. Tiene alta visibilidad debido a ciclos muy cortos y una planificación incremental, se adapta a cambios de negocio.

- Basado en pruebas automatizadas escritas por desarrolladores y clientes (test-driven)
- Alta comunicación
- Diseño evolutivo
- Colaboración entre programadores

Consta de 4 fases:



Actividades en XP

Busca equilibrio entre las necesidades a corto plazo de los programadores y las de largo plazo del proyecto.

ATRIBUTOS DE UN BUEN SOFTWARE

Un software de calidad debe de cumplir con características indispensables.

Mantenibilidad. El software debe de poder evolucionar para cumplir con las características y necesidades de cambio de los clientes.

Confiabilidad. El software debe de ser fiable, seguro, es decir, no debe causar daños físicos o económicos en caso de que el sistema falle.

Eficiencia. El software debe aprovechar al máximo los recursos del sistema.

Usabilidad. El software debe de ser fácil de usar.

ESTIMACIÓN DEL TAMAÑO DEL PROYECTO

Métricas para la estimación de tamaño

Se clasifican en dos:

Medidas relacionadas al tamaño

- Se relacionan con el tamaño de la salida de alguna actividad.
- Líneas de código (LDC).
- Depende del lenguaje de programación y en general no es una buena medida para la programación orientada a objetos (POO).

Medidas con relación a la función

Son medidas que se relacionan con la funcionalidad del software. Entre las más comunes están:

- Puntos de fusión (PF).
- Puntos de objeto (PO).

DETERMINACIÓN DE LA VIABILIDAD DE UN PRODUCTO DE SOFTWARE

Existen tres formas principales para evaluar la viabilidad de los proyectos de sistemas:

- En base a su operación.
- En base a su capacidad técnica.
- En base a su economía.

VIABILIDAD TÉCNICA

El analista debe averiguar si es posible desarrollar el nuevo sistema teniendo en cuanto a los recursos técnicos actuales. De no ser así, ¿se puede actualizar o complementar el sistema de tal forma que pueda cumplir con lo que se requiere? Si no es posible complementar o actualizar los sistemas existentes, la siguiente pregunta es si existe o no la tecnología que cumpla con las especificaciones. Al mismo tiempo, el analista puede preguntar si la organización cuenta con el personal que tenga la habilidad técnica suficiente para lograr los objetivos. De no ser así, la pregunta es si pueden o no contratar programadores, probadores, expertos o demás personal adicional que pueda tener habilidades de programación distintas a las del personal existente, o si tal vez pueden subcontratar un tercero para que se haga cargo del proyecto. Otra de las preguntas es si hay o no paquetes de software disponibles que puedan lograr sus objetivos, o si hay que personalizar el software para la organización.

VIABILIDAD ECONÓMICA

La viabilidad económica es la segunda parte de la determinación de recursos. Los recursos básicos que considerar son el tiempo de usted como analista y el tiempo de su equipo de análisis de sistemas, el costo de realizar un estudio de sistemas completo (incluyendo el tiempo de los empleados con los que usted va a trabajar), el costo del tiempo del empleado de la empresa, el costo estimado del hardware y el costo estimado del software o del desarrollo de software.

La empresa afectada debe ser capaz de ver el valor de la inversión que está considerando antes de comprometerse con un estudio de sistemas completo. Si los costos a corto plazo no se ven eclipsados por las ganancias a largo plazo o no producen una reducción inmediata en los costos de operación, entonces el sistema no es económicamente viable y el proyecto no debe continuar.

VIABILIDAD OPERACIONAL

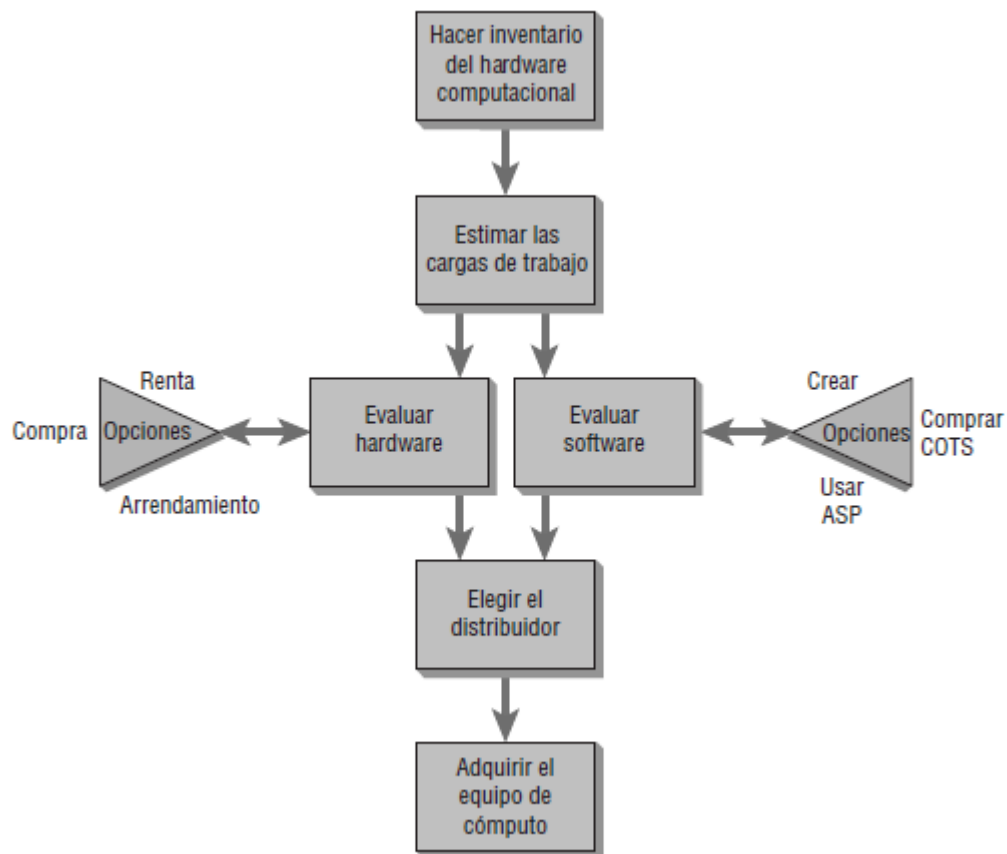
Suponga por un instante que tanto los recursos técnicos como económicos se consideran adecuados. El analista de sistemas debe aún considerar la viabilidad operacional del proyecto solicitado. La viabilidad operacional depende de los recursos humanos disponibles para el proyecto e implica la acción de pronosticar si el sistema funcionará y se utilizará una vez instalado.

Si los usuarios están prácticamente casados con el sistema actual, no ven problemas con él y por lo general no están involucrados en el proceso de solicitar un nuevo sistema, habrá mucha resistencia a la implementación del nuevo. Las probabilidades de que se vuelva funcional en algún momento dado serán bajas.

Por otro lado, si los mismos usuarios han expresado la necesidad de un sistema que sea funcional por más tiempo, de una forma más eficiente y accesible, hay más probabilidades de que el sistema solicitado se llegue a utilizar en un momento dado.

DETERMINACIÓN DE LAS NECESIDADES DE HARDWARE Y SOFTWARE

Lo primero y más importante es realizar un inventario de todo el software y hardware existente para descubrir que hay disponible y que se puede utilizar. La manera más correcta de determinar las necesidades se resume en el siguiente diagrama:



CUANDO CREAR SOFTWARE PERSONALIZADO

Cuando la organización trata de obtener una ventaja competitiva por medio del aprovechamiento del sistema. El crear su propio software tiene ventajas como:

- Responder a las necesidades especializadas de usuarios y empresas.
- Ventaja competitiva al crear software innovador
- Tener personal interno disponible para dar mantenimiento al software

Por otro lado, existen desventajas como:

- Costo inicial el cual resulta considerablemente más alto que la compra de software COTS o la renta de un ASP

HERRAMIENTA CASE

Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadoras). Son diversas Aplicaciones informáticas destinadas a aumentar la productividad en el Desarrollo de software reduciendo el coste de estas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, Compilación automática, documentación o detección de errores entre otras.

Tecnología de las herramientas CASE

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información a la hora de construir software se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta conseguimos agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

CRM

La sigla CRM quiere decir “Customer Relationship Management” (Gestión de las Relaciones con los Clientes). Este término se refiere a un conjunto de prácticas, estrategias de negocios y tecnologías enfocadas en clientes, que van desde pequeñas hasta grandes empresas, quienes las utilizan para gestionar y analizar las interacciones con sus clientes, anticipar sus necesidades y deseos, optimizar la rentabilidad, y aumentar las ventas y la objetividad de sus campañas de captación de nuevos clientes.

CRM almacena información sobre clientes actuales y potenciales (nombres, direcciones, números telefónicos, etc.), y sus actividades y puntos de contacto con la empresa, que incluyen visitas a sitios web, llamadas telefónicas, correos electrónicos y más.

¿Cuál es la función de CRM?

- Rastrea y gestiona de forma eficaz la información de los clientes.
- Conecta a todo su equipo en cualquier dispositivo.
- Captura de manera inteligente los correos electrónicos de los clientes.
- Simplifica tareas repetitivas para que usted pueda hacer un acompañamiento de leads más efectivo.
- Se adapta al crecimiento de su empresa.

señales que indican la necesidad de CRM

- Tiene equipos que trabajan juntos, incluso cuando no están en el mismo lugar.
- Sus equipos de ventas viajan frecuentemente.
- No consigue encontrar rápidamente los datos de clientes para tomar decisiones en el momento.
- Siente que las negociaciones están pasando desapercibidas porque tiene que gestionar todo en planillas y cuadernos.
- Tiene una diversidad de aplicaciones que llama CRM, pero que no están realmente conectadas a una base de datos o “sistema de registros”.
- Su empresa está creciendo muy rápido y usted no está preparado.
- Usted sabe que la experiencia de atención al cliente de su empresa está dejando que desear o está perdiendo más clientes de lo que les gustaría debido a problemas en el servicio.
- Usted o su departamento de TI están sobrecargados con solicitudes de mantenimiento.

METODOLOGÍA RUP

La metodología RUP, abreviatura de Rational Unified Process (o Proceso Unificado Racional), proporcionando técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo.

La metodología RUP utiliza el enfoque de la orientación a objetos en su diseño y está diseñado y documentado el uso de la notación UML (Unified Modeling Language) para ilustrar los procesos en acción. Para la gestión del proyecto, la metodología RUP proporciona una solución disciplinada como las tareas y responsabilidades señaladas dentro de una organización de desarrollo de software.

METODOLOGÍA ERP

Los sistemas de planificación de recursos empresariales (ERP, por sus siglas en inglés, enterprise resource planning) son los sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios.

Los sistemas ERP típicamente manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de la compañía de forma modular.

Los objetivos principales de los sistemas ERP son:

- Optimización de los procesos empresariales.
- Acceso a la información.
- Posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias de reingeniería.

OPENUP

es un método y un proceso de desarrollo de software. es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto no provee lineamientos para todos los elementos que se manejan en un proyecto pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos de OpenUP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

Principios del OpenUP

- ❖ Colaborar para sincronizar intereses y compartir conocimiento. Este principio promueve prácticas que impulsan un ambiente de equipo saludable, facilitan la colaboración y desarrollan un conocimiento compartido del proyecto.
- ❖ Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto. Este principio promueve prácticas que permiten a los participantes de los proyectos desarrollar una solución que maximice los beneficios obtenidos por los participantes y que cumple con los requisitos y restricciones del proyecto.
- ❖ Centrarse en la arquitectura de forma temprana para minimizar el riesgo y organizar el desarrollo.
- ❖ Desarrollo evolutivo para llevar a cabo retroalimentación una mejora continua. Este principio promueve prácticas que permiten a los equipos de desarrollo obtener retroalimentación temprana y continua de los participantes del proyecto, permitiendo demostrarles incrementos progresivos en la funcionalidad.

SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

EL MODELO COCOMO

El Modelo Constructivo de Costes (Constructive Cost Model) fue desarrollado por B. W. Boehm a finales de los 70 y comienzos de los 80. COCOMO es una jerarquía de modelos de estimación de costes software que incluye submodelos básicos, intermedio y detallado.

Modelo Básico

Este modelo trata de estimar, de una manera rápida y más o menos burda, la mayoría de los proyectos pequeños y medianos. Se consideran tres modos de desarrollo en este modelo: orgánico, semi-encajado y empotrado.

Modelo Intermedio

En este modelo se introducen 15 atributos de coste para tener en cuenta el entorno de trabajo. Estos atributos se utilizan para ajustar el coste nominal del proyecto al entorno real, incrementando la precisión de la estimación.

Modelo Detallado

Este modelo puede procesar todas las características del proyecto para construir una estimación. Introduce dos características principales:

- Multiplicadores de esfuerzo sensitivos a la fase. Algunas fases se ven más afectadas que otras por los atributos. El modelo detallado proporciona un conjunto de multiplicadores de esfuerzo para cada atributo. Esto ayuda a determinar la asignación del personal para cada fase del proyecto.
- Jerarquía del producto a tres niveles. Se definen tres niveles de producto. Estos son módulo, subsistema y sistema. La cuantificación se realiza al nivel apropiado, esto es, al nivel al que es más susceptible la variación.

PERSONAL SOFTWARE PROCESS (PSP)

El proceso personal de software es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejora de la productividad personal de los programadores o ingenieros de software, en tareas de desarrollo y mantenimiento de sistemas, mediante el seguimiento del desempeño predicho frente al desempeño real. Con PSP los ingenieros de software pueden adquirir las habilidades necesarias para trabajar en un proceso de software en equipo TSP.

Objetivos

PSP pretende formar ingenieros de software con métodos disciplinados para mejorar su desarrollo personal de software. PSP le ayuda a los desarrolladores a:

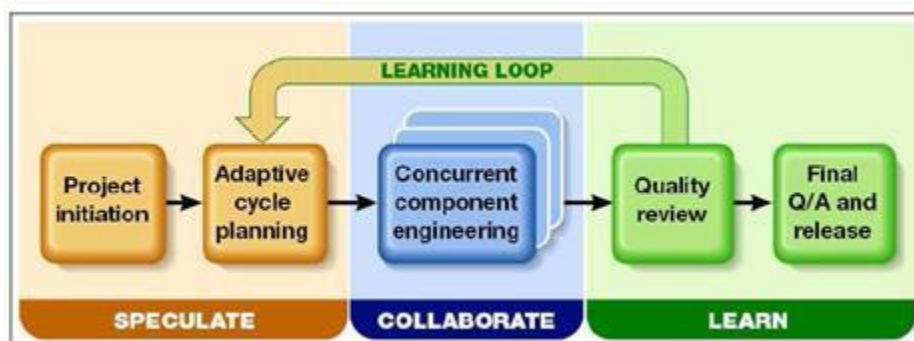
- Mejorar sus habilidades de estimación y planeación.
- Hacer compromisos que se puedan cumplir.
- Administrar la calidad de sus procesos.
- Reducir la cantidad de defectos en sus productos.

METODO AGIL ASD (ADAPTIVE SOFTWARE DEVELOPMENT)

Traducido en español significa Desarrollo Adaptable de Software es un modelo de implementación de patrones ágiles para desarrollo de software. Al igual que otras metodologías ágiles, su funcionamiento es cíclico y reconoce que en cada iteración se producirán cambios e incluso errores.

CARACTERISITICAS





- Iterativo.
- Orientado a los componentes de software (la funcionalidad que el producto va a tener, características, etc.) más que a las tareas en las que se va a alcanzar dicho objetivo.
- Tolerante a los cambios.
- Guiado por los riesgos.
- La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.



DSDM (METODOLOGÍA ÁGIL)

El DSDM (Dynamic Systems Development Method), que provee un marco para el desarrollo ágil de softwares. Este método funciona gracias al involucramiento constante del usuario en este desarrollo que se caracteriza por ser iterativo y sensible a los requerimientos cambiantes, para desarrollar un sistema que reúna las necesidades de la empresa en tiempo y presupuesto.

El método DSDM proporciona un marco de cuatro fases que consisten en:

-  Estudio de factibilidad y negocio
-  Modelo funcional / iteración de prototipo
-  Diseño y construcción de iteración
-  Implementación

SQA (SOFTWARE QUALITY ASSURANCE O ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE)

Implica revisar y auditar los productos y actividades de software para verificar que se cumplen los procedimientos y los estándares, además de proveer a las gerencias apropiadas (incluyendo a la de proyectos) con los resultados de estas revisiones. Por lo tanto, SQA envuelve al PROCESO de Desarrollo de software completo: monitoreando y mejorando el proceso; asegurándose que cualquier estándar y procedimientos adoptados sean seguidos; y, asegurándose que los problemas sean encontrados y tratados.

Definición

SQA es un set de actividades sistemáticas que aseguran que el proceso del software y productos conformados por requerimientos, estándares, y procedimientos. Los procesos incluyen todas las actividades involucradas en el diseño, codificación, pruebas y mantenimiento; Los productos incluyen software, datos asociados, documentación, y toda la documentación para soporte y reportes.

Propósito

Proporcionar visibilidad sobre los procesos utilizados por el proyecto de software y sobre los productos que genera.

Objetivos

- Planificar las actividades de aseguramiento de la calidad.
- Revisar y auditar objetivamente los productos y las actividades para verificar que están conformes con los procedimientos y estándares aplicables.
- Proporcionar los resultados de estas revisiones o auditorías informando a la dirección cuando sea necesaria su mediación.

MODELO CMMI

El CMMI es un enfoque de mejora de procesos que provee a las organizaciones de los elementos esenciales para un proceso efectivo. El CMMI es el Modelo de Madurez de Capacidades Integrado.

Objetivos

- Producir servicios y Productos de alta calidad.
- Crear valor para los accionistas.
- Mejorar la satisfacción del cliente.
- Incrementar la participación en el mercado.
- Ganar reconocimiento en la industria.

DIAGRAMAS DEL UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Recordemos que un modelo es una representación simplificada de la realidad; el modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación, se describirán los diagramas más comunes del UML y los conceptos que representan:







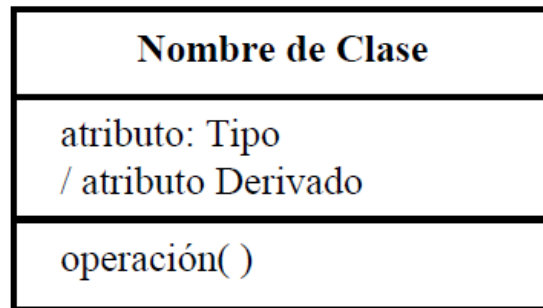
-  Diagrama de Clases
-  Diagrama de Objetos
-  Diagrama de Casos de Uso
-  Diagrama de Estados
-  Diagrama de Secuencias
-  Diagrama de Actividades

Diagrama de Clases

Los diagramas de clases describen la estructura estática de un sistema. Las cosas que existen y que nos rodean se agrupan naturalmente en categorías. Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares. Un ejemplo puede ser la clase “Aviones” que tiene atributos como el “modelo de avión”, “la cantidad de motores”, “la velocidad de crucero” y “la capacidad de carga útil”. Entre las acciones de las cosas de esta clase se encuentran: “acelerar”, “elevarse”, “girar”, “descender”, “desacelerar”. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que representan las asociaciones o maneras en que las clases se relacionan entre sí.

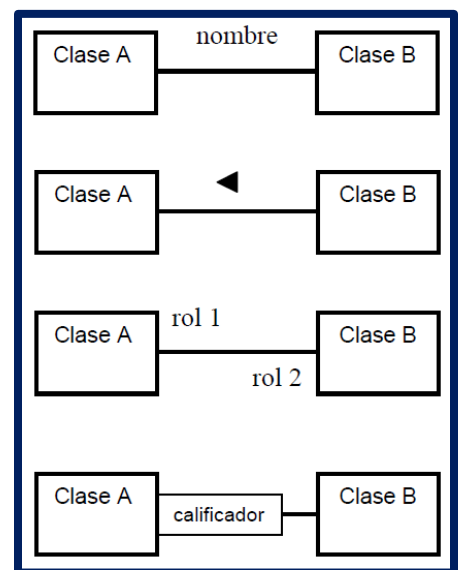
Clase Abstracta

Las clases se representan con rectángulos divididos en tres áreas: la superior contiene el nombre de la clase, la central contiene los atributos y la inferior las acciones.



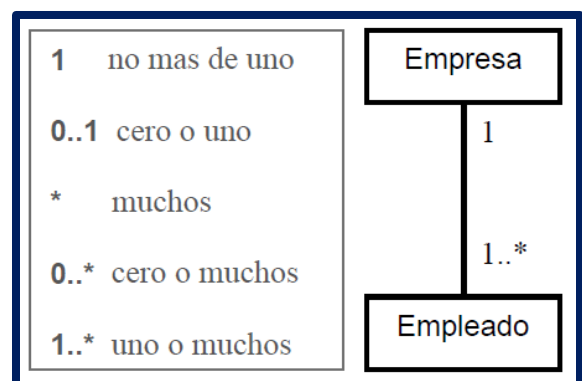
Asociaciones

Las asociaciones son las que representan a las relaciones estáticas entre las clases. El nombre de la asociación va por sobre o por debajo de la línea que la representa. Una flecha rellena indica la dirección de la relación. Los roles se ubican cerca del final de una asociación. Los roles representan la manera en que dos clases se ven entre ellas. No es común el colocar ambos nombres, el de la asociación y el de los roles a la vez. Cuando una asociación es calificada, el símbolo correspondiente se coloca al final de la asociación, contra la clase que hace de calificador.



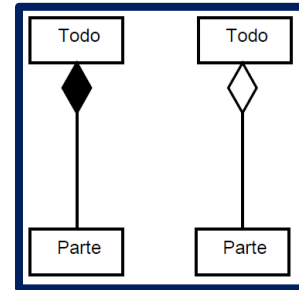
Multiplicidad

Las notaciones utilizadas para señalar la multiplicidad se colocan cerca del final de una asociación. Estos símbolos indican el número de instancias de una clase vinculadas a una de las instancias de la otra clase. Por ejemplo, una empresa puede tener uno o más empleados, pero cada empleado trabaja para una sola empresa solamente.



Composición y Agregación

Composición es un tipo especial de agregación que denota una fuerte posesión de la Clase "Todo", a la Clase "Parte". Se grafica con un rombo diamante relleno contra la clase que representa el todo. La agregación es una relación en la que la Clase "Todo" juega un rol más importante que la Clase "Parte", pero las dos clases no son dependientes una de otra. Se grafica con un rombo diamante vacío contra la Clase "Todo".



Generalización

Es otro nombre para herencia. Se refiere a una relación entre dos clases en donde una Clase "Específica" es una versión especializada de la otra, o Clase "General". Por ejemplo, Honda es un tipo de auto, por lo que la Clase "Honda" va a tener una relación de generalización con la Clase "Auto".

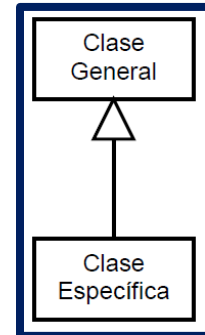
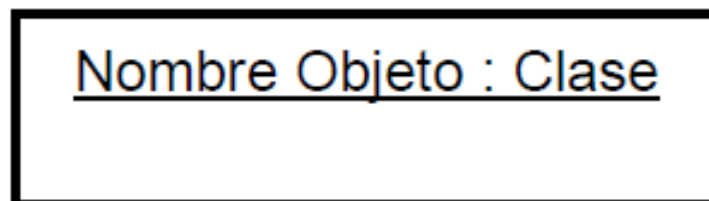


Diagrama de Objetos

Los Diagramas de Objetos están vinculados con los Diagramas de Clases. Un objeto es una instancia de una clase, por lo que un diagrama de objetos puede ser visto como una instancia de un diagrama de clases. Los diagramas de objetos describen la estructura estática de un sistema en un momento particular y son usados para probar la precisión de los diagramas de clases.

Nombre de los objetos

Cada objeto es representado como un rectángulo, que contiene el nombre del objeto y su clase subrayadas y separadas por dos puntos.



Atributos

Como con las clases, los atributos se listan en un área inferior. Sin embargo, los atributos de los objetos deben tener un valor asignado.

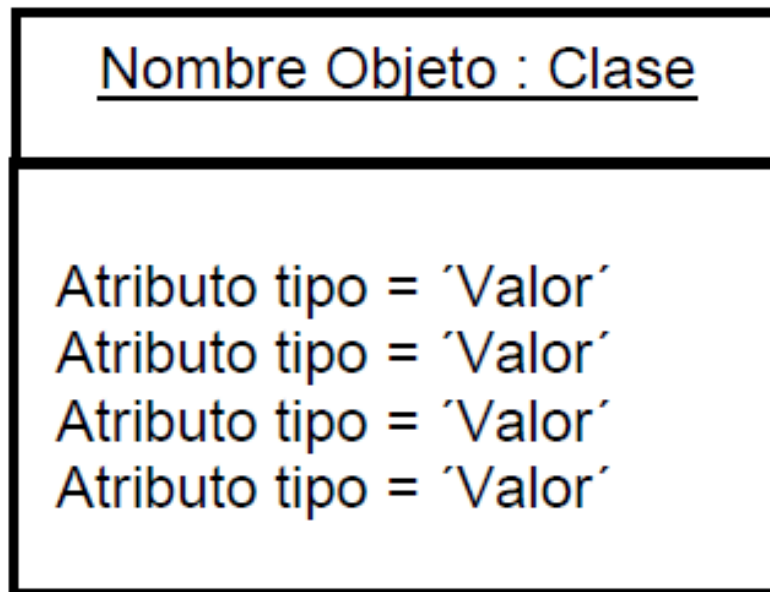


Diagrama de Casos de Uso

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Es una herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente desde el punto de vista del usuario. Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y casos de uso. Los casos de uso son servicios o funciones provistas por el sistema para sus usuarios.

Sistema

El rectángulo representa los límites del sistema que contiene los casos de uso. Los actores se ubican fuera de los límites del sistema.

Casos de Uso

Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.

Actores

Los actores son los usuarios de un sistema.

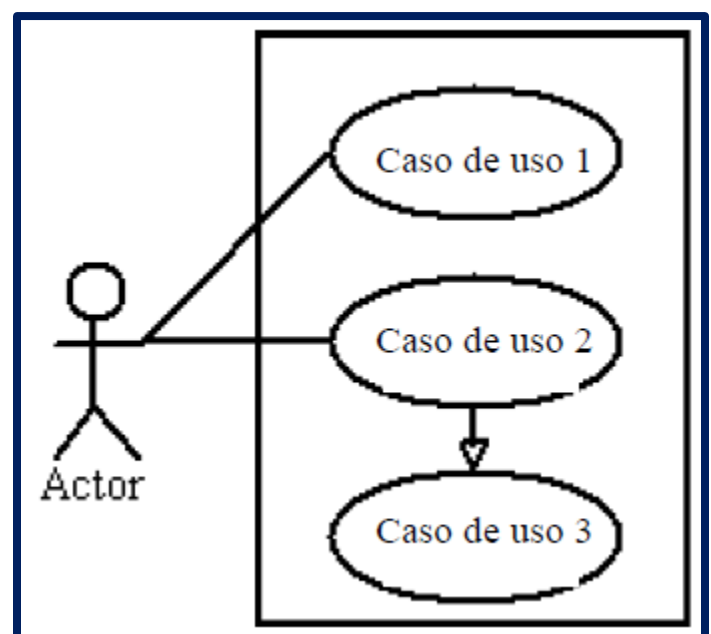


Diagrama de Estados

En cualquier momento, un objeto se encuentra en un estado particular, la luz está encendida o apagada, el auto en movimiento o detenido, la persona leyendo o cantando, etc. El diagrama de estados UML captura esa pequeña realidad.

Ejemplo de diagrama de estado de un avión:

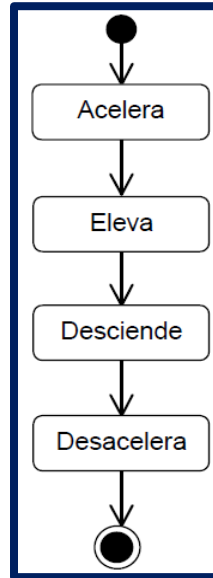


Diagrama de Secuencias

Los diagramas de clases y los de objetos representan información estática. No obstante, en un sistema funcional, los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.

Rol de la Clase

El rol de la clase describe la manera en que un objeto se va a comportar en el contexto. No se listan los atributos del objeto.

Objeto : Clase

Activación

Los cuadros de activación representan el tiempo que un objeto necesita para completar una tarea.

Mensajes

Los mensajes son flechas que representan comunicaciones entre objetos. Las medias flechas representan mensajes asíncronos. Los mensajes asíncronos son enviados desde un objeto que no va a esperar una respuesta del receptor para continuar con sus tareas.

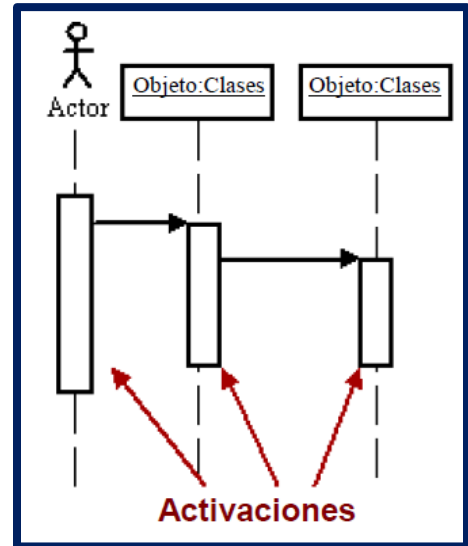
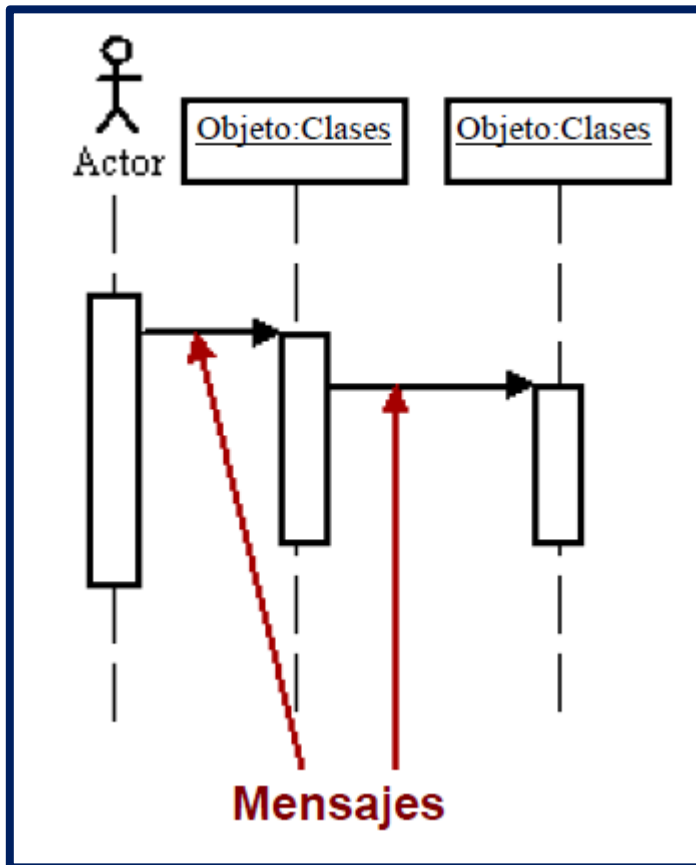
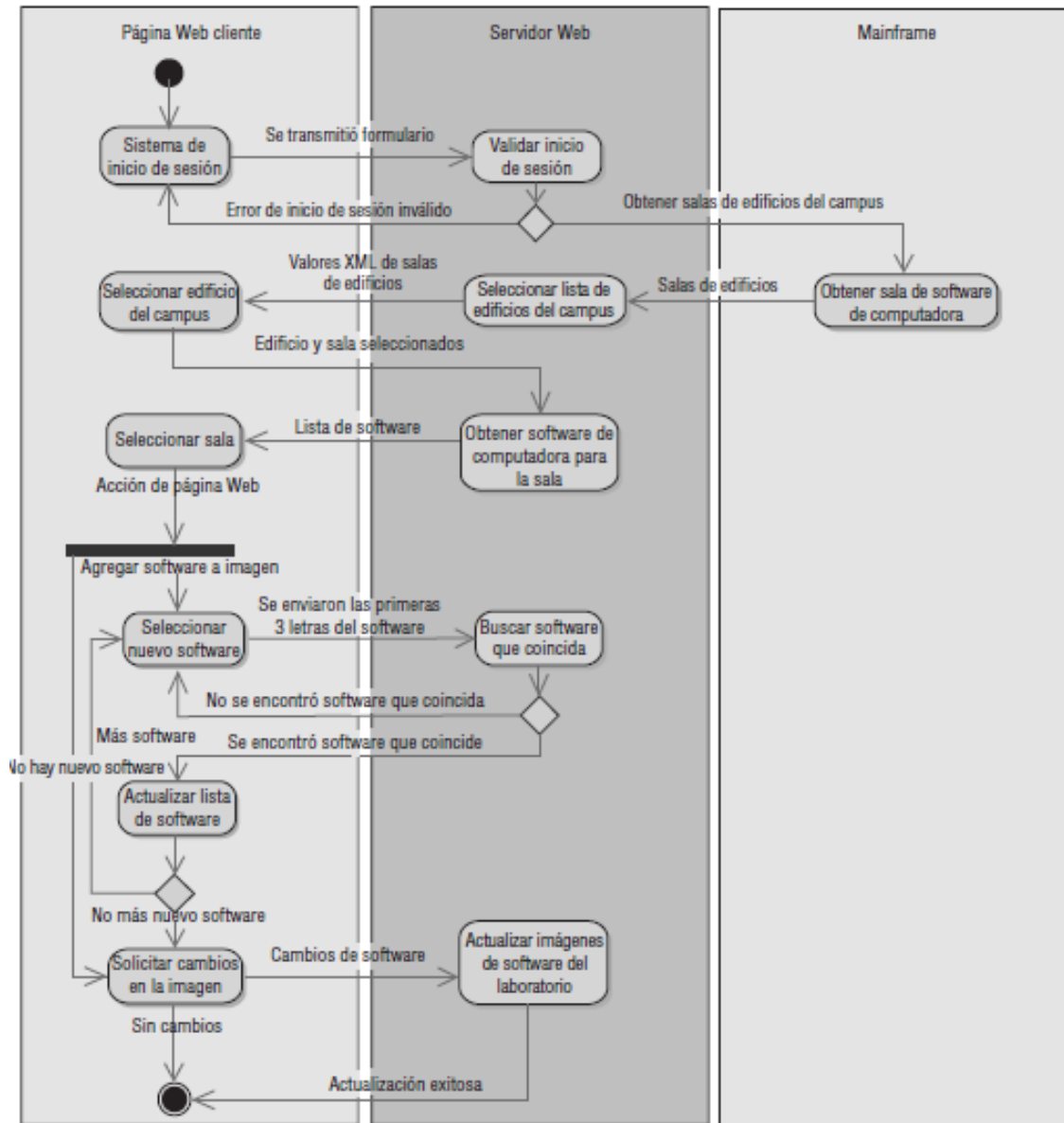


Diagrama de Actividades

Un diagrama de actividades ilustra la naturaleza dinámica de un sistema mediante el modelado del flujo ocurrente de actividad en actividad. Una actividad representa una operación en alguna clase del sistema y que resulta en un cambio en el estado del sistema. Típicamente, los diagramas de actividad son utilizados para modelar el flujo de trabajo interno de una operación.



ESTÁNDARES DE CALIDAD ISO PARA DESARROLLO DE SOFTWARE

El estándar de calidad ISO 9001

Este estándar ha sido adoptado por más de 130 países para su uso, Uno de sus problemas es que no es específico de la industria: está interpretado en términos generales y puede ser interpretado por los desarrolladores de diversos productos como, por ejemplo; cojines, secadores de cabello, automóviles, etc.

Para la industria del software los estándares relevantes son:







ISO 9001: Describe el sistema de calidad utilizado para mantener el desarrollo de un producto que implique diseño.

ISO 9000.3: Este es un documento específico que interpreta el ISO 9001 para el desarrollo del software.

ISO 9004-2: este documento proporciona las directrices para el servicio de facilidades del software como soporte de usuarios.

Factores de calidad ISO 9126



Ha sido desarrollado en un intento de identificar los atributos clave de calidad para el software. El estándar identifica 6 atributos clave de calidad:

-  Funcionalidad: el grado en que el software satisface las necesidades indicadas por los siguientes sub-atributos: idoneidad, corrección, inter-operatividad, conformidad y seguridad.
-  Confiabilidad: cantidad de tiempo que el software está disponible para su uso. Está referido por los siguientes sub-atributos: madurez, tolerancia a fallos y facilidad de recuperación.
-  Usabilidad: grado en que el software es fácil de usar. Viene reflejado por los siguientes sub-atributos: facilidad de comprensión, facilidad de aprendizaje y operatividad.
-  Eficiencia: grado en que el software hace óptimo el uso de los recursos del sistema. Está indicado por los siguientes sub-atributos: tiempo de uso y recursos utilizados.
-  Facilidad de mantenimiento: la facilidad con que una modificación puede ser realizada. Está indicada por los siguientes sub-atributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba.
-  Portabilidad: la facilidad con que el software puede ser llevado de un entorno a otro. Está referido por los siguientes sub-atributos: facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio.

Estándares ISO/IEC 25000



ISO/IEC 2500n – División de Gestión de Calidad

Las normas que forman este apartado definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000. Actualmente esta división se encuentra formada por:

-  ISO/IEC 25000 - Guide to SQuaRE: contiene el modelo de la arquitectura de SQuaRE, la terminología de la familia, un resumen de las partes, los usuarios previstos y las partes asociadas, así como los modelos de referencia.
-  ISO/IEC 25001 - Planning and Management: establece los requisitos y orientaciones para gestionar la evaluación y especificación de los requisitos del producto software.






ISO/IEC 2501n – División del Modelo de Calidad

Las normas de este apartado presentan modelos de calidad detallados incluyendo características para calidad interna, externa y en uso del producto software. Actualmente esta división se encuentra formada por:

-  ISO/IEC 25010 - System and software quality models: describe el modelo de calidad para el producto software y para la calidad en uso. Esta Norma presenta las características y subcaracterísticas de calidad frente a las cuales evaluar el producto software.
-  ISO/IEC 25012 - Data Quality model: define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información.

ISO/IEC 2502n – División de Medición de Calidad

Estas normas incluyen un modelo de referencia de la medición de la calidad del producto, definiciones de medidas de calidad (interna, externa y en uso) y guías prácticas para su aplicación. Actualmente esta división se encuentra formada por:

-  ISO/IEC 25020 - Measurement reference model and guide: presenta una explicación introductoria y un modelo de referencia común a los elementos de medición de la calidad. También proporciona una guía para que los usuarios seleccionen o desarrollen y apliquen medidas propuestas por normas ISO.
-  ISO/IEC 25021 - Quality measure elements: define y especifica un conjunto recomendado de métricas base y derivadas que puedan ser usadas a lo largo de todo el ciclo de vida del desarrollo software.
-  ISO/IEC 25022 - Measurement of quality in use: define específicamente las métricas para realizar la medición de la calidad en uso del producto.
-  ISO/IEC 25023 - Measurement of system and software product quality: define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.
-  ISO/IEC 25024 - Measurement of data quality: define específicamente las métricas para realizar la medición de la calidad de datos.

NORMAS ISO/IEC

ISO 12207 – Modelos de Ciclos de Vida del Software.

Estándar para los procesos de ciclo de vida del software de la organización, Este estándar se concibió para aquellos interesados en adquisición de software, así como desarrolladores y proveedores. El estándar indica una serie de procesos desde la recopilación de requisitos hasta la culminación del software.

El estándar comprende 17 procesos lo cuales son agrupados en tres categorías:

- ✓ Principales
- ✓ De apoyo
- ✓ De organización

Norma ISO/IEC 9126

La norma ISO/IEC 9126 de 1991, es la norma para evaluar los productos de software, esta norma nos indica las características de la calidad y los lineamientos para su uso, las características de calidad y sus métricas asociadas, pueden ser útiles tanto como para evaluar el producto como para definir los requerimientos de la calidad y otros usos. Esta norma definida por un marco conceptual basado en los factores tales como Calidad del Proceso, Calidad del Producto del Software y Calidad en Uso; según el marco conceptual, la calidad del producto, a su vez, contribuye a mejorar la calidad en uso.

La norma ISO/IEC 9126 define la calidad en uso como la perspectiva del usuario de la calidad del producto software cuando éste es usado en un ambiente específico y un contexto de uso específico. Éste mide la extensión para la cual los usuarios pueden conseguir sus metas en un ambiente particular, en vez de medir las propiedades del software en sí mismo.

El modelo de la calidad en uso muestra un conjunto de 4 características: efectividad, productividad, integridad, y satisfacción.

Parte beneficiaria Características	Usuario Final	Organización	Soporte técnico
Efectividad	Efectividad del usuario	Efectividad de las tareas	Efectividad del mantenimiento
Recursos	Productividad del usuario (tiempo)	Coste-Eficiencia (dinero)	Coste del mantenimiento
Consecuencias adversas	Riesgos para el usuario (salud y seguridad)	Riesgo comercial	Corrupción o fallos del software
Satisfacción	Satisfacción del usuario	Satisfacción en la gestión	Satisfacción del mantenimiento

Estándar ISO/IEC 14598

El estándar ISO/IEC 14598 es actualmente usado como base metodológica para la evaluación del producto software. En sus diferentes etapas, establece un marco de trabajo para evaluar la calidad de los productos de software proporcionando, además, métricas y requisitos para los procesos de evaluación de estos. La norma define las principales características del proceso de evaluación

- ✓ Repetitividad.
- ✓ Reproducibilidad.
- ✓ Imparcialidad.
- ✓ Objetividad.

DISEÑO Y PROGRAMACIÓN WEB

AJAX

JavaScript Asíncrono y XML (AJAX) no es una tecnología por sí misma, es un término que describe un nuevo modo de utilizar conjuntamente varias tecnologías existentes. Esto incluye: HTML o XHTML, CSS, JavaScript, DOM, XML, XSLT, y el objeto XMLHttpRequest. Cuando estas tecnologías se combinan en un modelo AJAX, es posible lograr aplicaciones web capaces de actualizarse continuamente sin tener que volver a cargar la página completa. Esto crea aplicaciones más rápidas y con mejor respuesta a las acciones del usuario.

XML

- ❖ XML es un subconjunto de SGML (Estándar Generalised Mark-up Language), simplificado y adaptado a Internet.
- ❖ XML es un metalenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.

Definición

Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos.

¿Para qué sirve XML?

Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

Ventajas de XML

- ❖ Fácilmente procesable
- ❖ Separa radicalmente el contenido y el formato de presentación
- ❖ Diseñado para cualquier lenguaje y alfabeto.

Características

XML es un subconjunto de SGML que incorpora las tres características más importantes de este:

- ❖ Extensibilidad
- ❖ Estructura
- ❖ Validación
- ❖ Basado en texto.
- ❖ Orientado a los contenidos no presentación.
- ❖ Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido.
- ❖ No es sustituto de HTML.
- ❖ No existe un visor genérico de XML.
- ❖ Aplicaciones de XML
- ❖ Publicar e intercambiar contenidos de bases de datos.
- ❖ Formatos de mensaje para comunicación entre aplicaciones (B2B)
- ❖ Descripción de meta contenidos.

APPSERV (TAILANDES)

- Fácil y rápida instalación
- Herramienta para restablecer la contraseña de MySQL
- Podemos decidir que instalar y que omitir, muy útil si ya tenemos alguna versión de MySQL instalada u otra base de datos instalada.

WAMP

- Proyecto respaldado por una empresa.
- Administración de módulos para Apache y extensiones para PHP de manera gráfica.

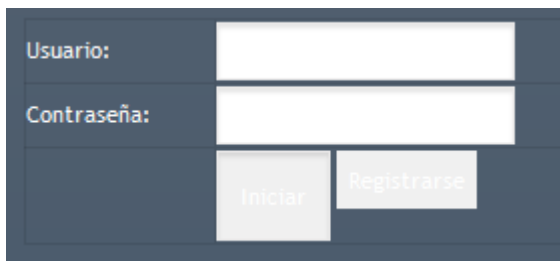
XAMPP

- Respaldado por Apache y Bitnami
- Nos permite elegir que componentes instalar, muy útil para no suplicar instalaciones
- Interfaz donde se concentran todas las opciones de administración
- Fácil instalación y rápido

	AppServ	WAMP	XAMPP
Distintas versiones de PHP	Si	Si	NO
Herramientas/Opciones de administración	Poco	Interfaz grafica	
Opción sobre que instalamos	Si	No	Si
Herramienta recuperación de contraseña MySQL	Si	No	No
Administración de extensiones PHP mediante interfaz	No	Si	No
Administración de módulos de Apache mediante interfaz	No	Si	No

CÓDIGO PARA DEFINIR UNA TABLA CON FILAS Y COLUMNAS

```
<table> <!--Se abre el tag, con este se define la tabla-->
  <tr> <!--Se define un salto de espacio-->
    <td>Usuario:</td><!--se define una columna-->
    <td><input type="text" id="txtusuario" name="txtusuario"></td>
  </tr>
  <tr>
    <td>Contraseña:</td>
    <td><input type="password" id="txtclave" name="txtclave" name="txtnombre"></td>
  </tr>
  <tr><td></td>
  <td align = "right"><input class="boton" type="button" id="boton" value="Iniciar"/>
  <input class="boton" type="submit" name="boton1" value="Registrarse"/></td>
</tr>
</table>
```



CÓDIGO PARA DEFINIR UNA IMAGEN

```
<div class="ws_images"><ul>
  <li></li><!--Se define una imagen-->
  <li></li>
  <li></li>
</ul>
</div>
```

CÓDIGO PARA ALINEAR UN OBJETO

```
<td align = "right"><input class="boton" type="button" id="boton" value="Iniciar"/>
```

En este caso se define un objeto tipo botón, el cual se está alineando a la derecha, la sentencia de alineación se escribe en inglés.

REDES DE COMPUTADORAS

Una red de computadoras (también llamada red de ordenadores, red de comunicaciones de datos, red informática) es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios. Como en todo proceso de comunicación, se requiere de un emisor, un mensaje, un medio y un receptor. La finalidad principal para la creación de una red de ordenadores es compartir los recursos y la información en la distancia, asegurar la confiabilidad y la disponibilidad de la información, aumentar la velocidad de transmisión de los datos y reducir el costo. Un ejemplo es Internet, el cual es una gran red de millones de ordenadores ubicados en distintos puntos del planeta interconectados básicamente para compartir información y recursos. La estructura y el modo de funcionamiento de las redes informáticas actuales están definidos en varios estándares, siendo el más importante y extendido de todos ellos el modelo TCP/IP basado en el modelo de referencia OSI.

Software

Sistema operativo de red

Permite la interconexión de ordenadores para acceder a los servicios y recursos. Al igual que un equipo no puede trabajar sin un sistema operativo, una red de equipos no puede funcionar sin un sistema operativo de red. En muchos casos el sistema operativo de red es parte del sistema operativo de los servidores y de los clientes.

Software de aplicación

En última instancia, todos los elementos se utilizan para que el usuario de cada estación pueda utilizar sus programas y archivos específicos. Este software puede ser tan amplio como se necesite ya que puede incluir procesadores de texto, paquetes integrados, sistemas administrativos de contabilidad y áreas afines, sistemas especializados, correos electrónicos, etc. El software adecuado en el sistema operativo de red elegido y con los protocolos necesarios permiten crear servidores para aquellos servicios que se necesiten.

Hardware

Para lograr el enlace entre los ordenadores y los medios de transmisión (cables de red o medios físicos para redes alámbricas e infrarrojos o radiofrecuencias para redes inalámbricas), es necesaria la intervención de una tarjeta de red (NIC, Network interface controller), con la cual se puedan enviar y recibir paquetes de datos desde y hacia otros ordenadores, empleando un protocolo para su comunicación y convirtiendo a esos datos a un formato que pueda ser transmitido por el medio (bits, -ceros y unos-).

DISPOSITIVOS DE RED

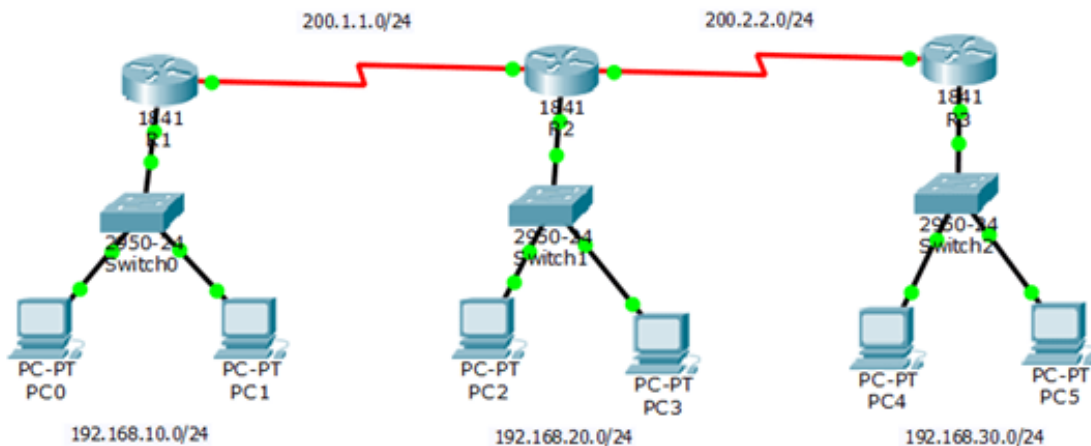
Conmutador

Conmutador (switch) es el dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más host de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red y eliminando la conexión una vez finalizada ésta. Los conmutadores se utilizan cuando se desea conectar múltiples tramos de una red, fusionándolos en una sola red.



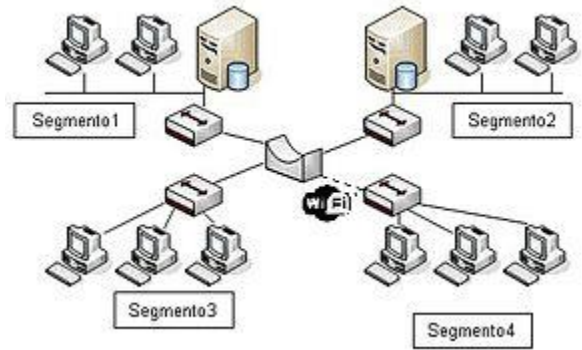
Router

también conocido como enrutador, es un dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI. Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes, entendiendo por subred un conjunto de máquinas IP que se pueden comunicar sin la intervención de un encaminador (mediante puentes de red o un switch), y que por tanto tienen prefijos de red distintos.



Puente de red

Puente de red (en inglés: bridge) es el dispositivo de interconexión de redes de computadoras que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Interconecta segmentos de red (o divide una red en segmentos) haciendo la transferencia de datos de una red hacia otra con base en la dirección física de destino de cada paquete. En definitiva, un bridge conecta segmentos de red formando una sola subred (permite conexión entre equipos sin necesidad de routers). Funciona a través de una tabla de direcciones MAC detectadas en cada segmento al que está conectado. Cuando detecta que un nodo de uno de los segmentos está intentando transmitir datos a un nodo del otro, el bridge copia la trama para la otra subred, teniendo la capacidad de desechar la trama (filtrado) en caso de no tener dicha subred como destino. Para conocer por dónde enviar cada trama que le llega (encaminamiento) incluye un mecanismo de aprendizaje automático (auto aprendizaje) por lo que no necesitan configuración manual.



Punto de acceso inalámbrico

Un punto de acceso inalámbrico (en inglés: wireless access point, conocido por las siglas WAP o AP), en una red de computadoras, es un dispositivo de red que interconecta equipos de comunicación inalámbricos, para formar una red inalámbrica que interconecta dispositivos móviles o tarjetas de red inalámbricas.

Son dispositivos que son configurados en redes de tipo inalámbricas que son intermediarios entre una computadora y una red (Internet o local). Facilitan conectar varias máquinas cliente sin la necesidad de un cable (mayor portabilidad del equipo) y que estas posean una conexión sin limitárseles tanto su ancho de banda. Los WAP son dispositivos que permiten la conexión de un dispositivo móvil de cómputo (computadora, tableta, smartphone) con una red. Normalmente, un WAP también puede conectarse a una red cableada, y puede transmitir datos entre los dispositivos conectados a la red cableada y los dispositivos inalámbricos. Los WAP tienen asignadas direcciones IP, para poder ser configurados.

MODELO OSI

Capa 7: La capa de aplicación

Es la capa del modelo OSI más cercana al usuario; suministra servicios de red a las aplicaciones del usuario. Difiere de las demás capas debido a que no proporciona servicios a ninguna otra capa OSI, sino solamente a

aplicaciones que se encuentran fuera del modelo OSI. Algunos ejemplos de aplicaciones son los programas de hojas de cálculo, de procesamiento de texto y los de las terminales bancarias. La capa de aplicación establece la disponibilidad de los potenciales socios de comunicación, sincroniza y establece acuerdos sobre los procedimientos de recuperación de errores y control de la integridad de los datos. Si desea recordar a la Capa 7 en la menor cantidad de palabras posible, piense en los navegadores de Web.

Capa 6: La capa de presentación

Garantiza que la información que envía la capa de aplicación de un sistema pueda ser leída por la capa de aplicación de otro. De ser necesario, la capa de presentación traduce entre varios formatos de datos utilizando un formato común. Si desea recordar la Capa 6 en la menor cantidad de palabras posible, piense en un formato de datos común.

Capa 5: La capa de sesión

Establece, administra y finaliza las sesiones entre dos hosts que se están comunicando. La capa de sesión proporciona sus servicios a la capa de presentación. También sincroniza el diálogo entre las capas de presentación de los dos hosts y administra su intercambio de datos. Además de regular la sesión, la capa de sesión ofrece disposiciones para una eficiente transferencia de datos, clase de servicio y un registro de excepciones acerca de los problemas de la capa de sesión, presentación y aplicación.

Capa 4: La capa de transporte

Segmenta los datos originados en el host emisor y los reensambla en una corriente de datos dentro del sistema del host receptor. El límite entre la capa de transporte y la capa de sesión puede imaginarse como el límite entre los protocolos de aplicación y los protocolos de flujo de datos. Mientras que las capas de aplicación, presentación y sesión están relacionadas con asuntos de aplicaciones, las cuatro capas inferiores se encargan del transporte de datos. La capa de transporte intenta suministrar un servicio de transporte de datos que aísla las capas superiores de los detalles de implementación del transporte. Específicamente, temas como la confiabilidad del transporte entre dos hosts es responsabilidad de la capa de transporte. Al proporcionar un servicio de comunicaciones, la capa de transporte establece, mantiene y termina adecuadamente los circuitos virtuales. Al proporcionar un servicio confiable, se utilizan dispositivos de detección y recuperación de errores de transporte. Si desea recordar a la Capa 4 en la menor cantidad de palabras posible, piense en calidad de servicio y confiabilidad.

Capa 3: La capa de red

Proporciona conectividad y selección de ruta entre dos sistemas de hosts que pueden estar ubicados en redes geográficamente distintas. Es decir, selecciona ruta, direccionamiento y enrutamiento.

Capa 2: La capa de enlace de datos

Proporciona tránsito de datos confiable a través de un enlace físico. Se ocupa del direccionamiento físico (comparado con el lógico), la topología de red, el acceso a la red, la notificación de errores, entrega ordenada de tramas y control de flujo. Se encarga de las tramas y control de acceso al medio.

Capa 1: La capa física

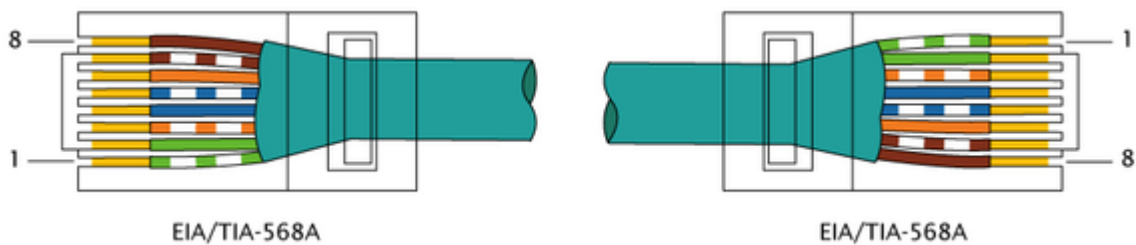
La capa física define las especificaciones eléctricas, mecánicas, de procedimiento y funcionales para activar, mantener y desactivar el enlace físico entre sistemas finales. Las características tales como niveles de voltaje, temporización de cambios de voltaje, velocidad de datos físicos, distancias de transmisión máxima, conectores físicos y otros atributos similares son definidos por las especificaciones de la capa física. Si desea recordar la Capa 1 en la menor cantidad de palabras posible, piense en señales y medios.

CABLE DE PARES TRENZADOS

Físicamente, son cables formados por cuatro pares de cobre trenzados que utilizan unos códigos de colores normalizados para identificar a cada uno de ellos. Cada uno de los 8 hilos del cable tiene un diámetro de entre 0,4 y 0,5 mm (26 AWG ó 24 AWG). El conjunto de cuatro pares viene envuelto en una cubierta externa de PVC. La impedancia característica de estos cables es de 100 Ω . Existen dos posibles configuraciones para estos cables:

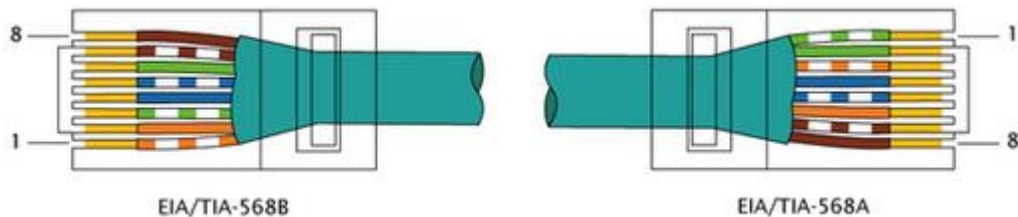
Cable directo T568A

El cable directo de red sirve para conectar dispositivos desiguales, como un computador con un hub o switch. En este caso, ambos extremos del cable deben tener la misma distribución. No existe diferencia alguna en la conectividad entre la distribución 568B y la distribución 568A siempre y cuando en ambos extremos se use la misma, en caso contrario hablamos de un cable cruzado.



Cable cruzado

Un cable cruzado es un cable que interconecta todas las señales de salida en un conector con las señales de entrada en el otro conector, y viceversa; permitiendo a dos dispositivos electrónicos conectarse entre sí con una comunicación full dúplex. El término se refiere comúnmente al cable cruzado de Ethernet, pero otros cables pueden seguir el mismo principio. También permite transmisión confiable vía una conexión Ethernet.



ALGORITMOS DE ENCRYPTACIÓN SIMÉTRICA Y ASIMÉTRICA

ENCRYPTACIÓN SIMÉTRICA

El cifrado mediante clave simétrica significa que dos o más usuarios, tienen una única clave secreta, esta clave será la que cifrará y descifrá la información transmitida a través del canal inseguro. Es decir, la clave secreta la debe tener los dos usuarios, y con dicha clave, el usuario A cifrará la información, la mandará a través del canal inseguro, y a continuación el usuario B descifrá esa información con la MISMA clave que ha usado el usuario A. Algunos algoritmos de clave simétrica:

DES (Data Encryption Standard)

Su arquitectura está basada en un sistema monoalfabético, donde un algoritmo de cifrado aplica sucesivas permutaciones y sustituciones al texto en claro. En un primer momento la información de 64bits se somete a una permutación inicial, y a continuación se somete a una permutación con entrada de 8 bits, y otra de sustitución de entrada de 5 bits, todo ello constituido a través de un proceso con 16 etapas de cifrado.

El algoritmo DES usa una clave simétrica de 64bits, los 56 primeros bits son empleados para el cifrado, y los 8 bits restantes se usan para comprobación de errores durante el proceso. La clave efectiva es de 56 bits, por tanto, tenemos 2^{56} combinaciones posibles, por lo que la fuerza bruta se hace casi imposible.

IDEA (International Data Encryption Algorithm)

Aplica una clave de 128 bits sin paridad a bloques de datos de 64 bits, y se usa tanto para cifrar como para descifrar.

Se alteran los datos de entrada en una secuencia de iteraciones parametrizadas, con el objetivo de producir bloques de salida de texto cifrado de 64 bits. IDEA combina operaciones matemáticas como XOR, sumas con acarreo de módulo 2^{16} y multiplicaciones de módulo $2^{16}+1$, sobre bloques de 16 bits. Según numerosos expertos criptográficos, IDEA es el mejor algoritmo de cifrado de datos existente en la actualidad ya que existen 2^{128} claves privadas que probar mediante el ataque de fuerza bruta.

AES (Advanced Encryption Standard)

Este algoritmo es el más conocido entre los usuarios de Reuters, ya que WPA opera con AES como método de cifrado. Este cifrado puede implementar tanto en sistemas hardware como en software. El sistema criptográfico AES opera con bloques y claves de longitudes variable, hay AES de 128bits, de 192 bits y de 256 bits.

El resultado intermedio del cifrado constituye una matriz de bytes de cuatro filas por cuatro columnas. A esta matriz se le vuelve a aplicar una serie de bucles de cifrado basado en operaciones matemáticas (sustituciones no lineales de bytes, desplazamiento de filas de la matriz, combinaciones de las columnas mediante multiplicaciones lógicas y sumas XOR en base a claves intermedias).

ENCRYPTACIÓN ASIMÉTRICA

La criptografía asimétrica nos ofrece autenticidad, que consiste en:

- ❖ Confidencialidad. Cifrando las comunicaciones.
- ❖ No repudio. Mediante firma electrónica.
- ❖ Integridad. El mensaje que recibimos es de quien dice ser y contiene lo que el remitente escribió.

Usos de este tipo de criptografía:

- ❖ Cifrado y descifrado de mensajes
- ❖ Firmado y verificación de mensajes.
- ❖ Acceso seguro a servicios remotos.
- ❖ Firmado de código.

Cómo funciona

Esta criptografía se basa en dos claves distintas (de ahí el nombre de criptografía asimétrica). Una de las claves se denomina pública y la otra privada. La clave pública (como su nombre indica) puede hacerse pública, por el contrario, la clave privada sólo es conocida por el propietario de esta. Entre los más populares están:

RSA (Rivest, Shamir, Adleman)

Creado en 1978, hoy es el algoritmo de mayor uso en encriptación asimétrica. Tiene dificultades para encriptar grandes volúmenes de información, por lo que es usado por lo general en conjunto con algoritmos simétricos.

Algoritmos de autenticación (o hash)

Una función hash es método para generar claves o llaves que representen de manera casi unívoca a un documento o conjunto de datos. Es una operación matemática que se realiza sobre este conjunto de datos de cualquier longitud, y su salida es una huella digital, de tamaño fijo e independiente de la dimensión del documento original. El contenido es ilegible.

SHA-1

Es parecido al famoso MD5, pero tiene un bloque de 160bits en lugar de los 128bits del MD5. La función de compresión es más compleja que la función de MD5. SHA-1 es más lento que MD5 porque el número de pasos son de 80 (64 en MD5) y porque tiene mayor longitud que MD5 (160bits contra 128bits). Lo que convierte a SHA-1 más robusto y seguro, totalmente apto para VPN's por ejemplo.

BASE DE DATOS

Colección de información, organizada y presentada para servir a un propósito específico, como la facilitación de búsquedas, ordenamientos o procesamiento de datos.

Campo

Elemento de información contenido dentro de un renglón o registro. Equivalente lógico a una columna.

Columna

Conjunto de renglones de una tabla que tienen un atributo común. Contienen un dato individual dentro de cada renglón o registro.

Dato

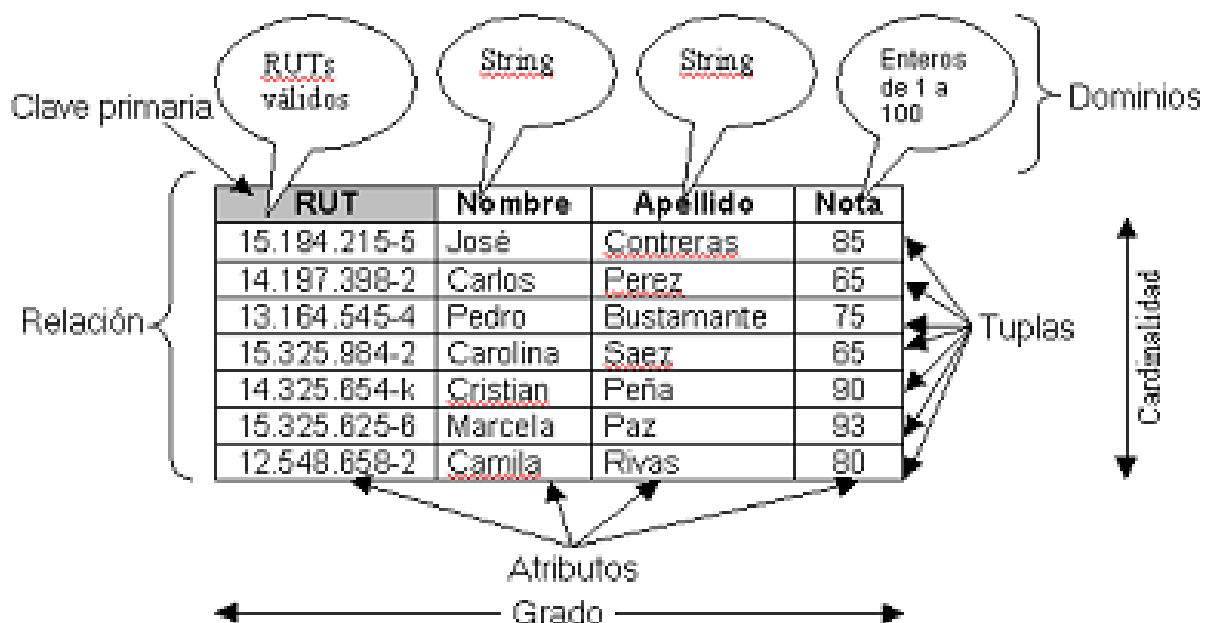
Representación codificada de información para usarla en una computadora. Los datos tienen atributos como tipo y longitud.

Tabla

Colección de renglones (o registros) que tienen columnas (o campos) asociadas.

Tupla

Una tupla es una secuencia de valores agrupados. Una tupla sirve para agrupar, como si fueran un único valor, varios valores que, por su naturaleza, deben ir juntos.



LENGUAJE DE BASE DE DATOS

Un sistema de base de datos proporciona un lenguaje de definición de datos y un lenguaje de manipulación de datos para expresar las consultas a la base de datos y modificaciones a la BD.

LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

Especifica el almacenamiento y los métodos de acceso por el sistema de BD por un conjunto de instrucciones.

LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)

Nos permite la recuperación de información, la inserción, el borrado y la modificación

Diagrama entidad relación

Se utiliza para la representación de sistemas y de información, su simbología se divide en cuatro elementos:

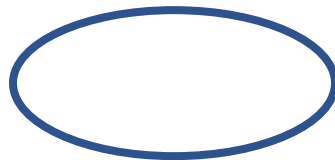
Entidades

Representan abstracciones del mundo real. Y su símbolo es un rectángulo.



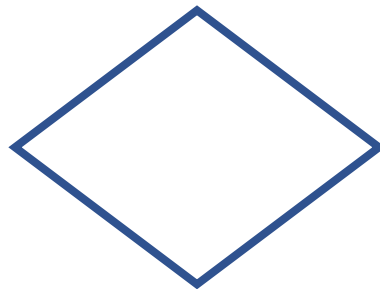
Atributos

Son características únicas de las entidades y son representadas con un ovalo



Diamantes

Representan relaciones entre entidades



Cardinalidad

Es el nivel de correspondencia que existe entre entidades hay 3 tipos_

- 1:1 Uno a uno
- 1:M Uno a muchos
- M:N Muchos a muchos

Consultas SQL (SELECT)

```
SELECT *  
FROM CLIENTES
```

Con el * indicamos que queremos devolver todos los campos. Si CLIENTES dispone de los campos idCliente, nombre y descripcion, lo anterior sería equivalente a:

```
SELECT idCliente, nombre, descripcion  
FROM CLIENTES
```

Consultas SQL (WHERE)

```
SELECT numero, calle  
FROM DIRECCION  
WHERE ciudad = 'Sevilla'
```

Esta consulta devolvería el número y la dirección de todas las direcciones pertenecientes a la ciudad de Sevilla. Como vemos, con WHERE indicamos la condición que deben cumplir los registros de la tabla para ser devueltos en la consulta.

Consultas SQL (BETWEEN)

Para indicar un intervalo de valores.

```
SELECT nombre  
FROM CLIENTES  
WHERE edad BETWEEN 20 AND 35
```

Consultas SQL (ORDENACIÓN)

Esta consulta devolvería todas las ciudades ordenadas por provincia en orden ascendente, y dentro de los de la misma provincia ordenaría las ciudades por orden descendente del número de habitantes. Si no indicamos ASC ni DESC, el comportamiento por defecto será el orden ascendente (ASC).

```
SELECT *  
FROM CIUDAD  
ORDER BY provincia ASC, numhabitantes DESC
```

INNER JOIN

La forma más habitual de INNER JOIN es la que interseca las tablas indicadas en con INNER JOIN por el campo indicado por ON.

```
SELECT nombreCliente, idPedido, fechaPedido  
FROM CLIENTE INNER JOIN PEDIDO  
ON cliente.idCliente = pedido.idCliente
```

LEFT JOIN

El resultado de esta operación siempre contiene todos los registros de la relación izquierda (primera tabla que indicamos), y aquellos de la tabla derecha que cumplen la condición establecida. Para el resto aparecerá en los campos correspondientes a dicha tabla un NULL.

```
SELECT nombreCliente, idPedido, fechaPedido  
FROM CLIENTE LEFT JOIN PEDIDO  
ON cliente.idCliente = pedido.idCliente
```

RIGHT JOIN

El RIGHT JOIN es análogo al LEFT JOIN, pero devolviendo todos los registros de la relación derecha (segunda tabla que aparece), y únicamente aquellos de la tabla izquierda que cumplen la condición del JOIN. El resultado de esta operación siempre contiene todos los registros de la relación derecha (segunda tabla que indicamos), de modo que en aquellos sin equivalente en la parte izquierda tendrán en los campos correspondientes a dicha tabla un NULL.

```
SELECT nombreCliente, idPedido, fechaPedido  
FROM CLIENTE RIGHT JOIN PEDIDO  
ON cliente.idCliente = pedido.idCliente
```

SISTEMAS OPERATIVOS

Es el software que se ejecuta en modo kernel, los sistemas operativos realizan dos funciones básicas que no están relacionadas: proporcionar a los programadores de aplicaciones (y a los programas de aplicaciones, naturalmente) un conjunto abstracto de recursos simples, en vez de los complejos conjuntos de hardware; y administrar estos recursos de hardware. Dependiendo de quién se esté hablando, el lector podría escuchar más acerca de una función o de la otra. Ahora analizaremos ambas.

SISTEMAS OPERATIVOS PARA SERVIDORES

Un servidor no es necesariamente una máquina de última generación de grandes proporciones, no es necesariamente un superordenador; un servidor puede ser desde una computadora de bajo recursos, hasta una máquina sumamente potente (ej.: servidores web, bases de datos grandes, etc. Procesadores especiales y hasta varios terabytes de memoria). Todo esto depende del uso que se le dé al servidor.

FreeBSD

Es un sistema operativo libre para computadoras basado en las CPU de arquitectura Intel, incluyendo procesadores Intel 80386, Intel 80486 (versiones SX y DX), y Pentium. También funciona en procesadores compatibles con Intel como AMD y Cyrix. Actualmente también es posible utilizarlo hasta en once arquitecturas distintas como Alpha, AMD64, IA-64, MIPS, PowerPC y UltraSPARC. FreeBSD está basado en la versión 4.4 BSD-Lite del Computer Systems Research Group (CSRG) de la University of California, Berkeley siguiendo la tradición que ha distinguido el desarrollo de los sistemas BSD.

Linux

Es un núcleo libre de sistema operativo (también suele referirse al núcleo como kernel) basado en Unix. Es uno de los principales ejemplos de software libre y de código abierto. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. El núcleo Linux fue concebido por el entonces estudiante de ciencias de la computación finlandés Linus Torvalds en 1991. Normalmente Linux se utiliza junto a un empaquetado de software, llamado distribución GNU/Linux y servidores.

Mac OS X Server

Es un sistema operativo para servidores desarrollado por Apple Inc. basado en Unix. Es idéntico a su versión de escritorio, pero incluye además herramientas administrativas gráficas para la gestión de usuarios, redes, y servicios de red como LDAP, Servidor de correo, Servidor Samba, DNS, entre otros. También incorpora en sus versiones más recientes un número adicional de servicios y herramientas para configurarlos, tales como Servidor web, herramientas para crear una Wiki, Servidor iChat, y otros más.

Microsoft Servers

(anteriormente llamado Windows Server System) es una marca que abarca una línea de productos de servidor de Microsoft. Esto incluye las ediciones de servidor de Microsoft Windows su propio sistema operativo, así como productos dirigidos al mercado más amplio de negocio. Algunas versiones: Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows HPC Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Small Business Server, Windows Essential Business Server, Windows Home Server.

Solaris

Es un sistema operativo de tipo Unix desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente por Oracle Corporation como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Funciona en arquitecturas SPARC y x86 para servidores y estaciones de trabajo.

Unix

(registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969, por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy. El sistema, junto con todos los derechos fueron vendidos por AT&T a Novell, Inc. Esta vendió posteriormente el software a Santa Cruz Operation en 1995, y esta, a su vez, lo revendió a Caldera Software en 2001, empresa que después se convirtió en el grupo SCO. En 2010, y tras una larga batalla legal, ésta ha pasado nuevamente a ser propiedad de Novell.

PROGRAMACIÓN C++

Sentencias IF

Ejemplo 1:

```
if(numero == 0) //La condicion indica que tiene que ser igual a Cero
{
    cout<<"El Numero Ingresado es Igual a Cero";
}
```

Ejemplo 2:

```
if(numero > 0) // la condicion indica que tiene que ser mayor a Cero
{
    cout<<"El Numero Ingresado es Mayor a Cero";
}
```

Ejemplo 3:

```
if(numero < 0) // la condicion indica que tiene que ser menor a Cero
{
    cout<<"El Numero Ingresado es Menor a Cero";
}
```

Sentencias IF-ELSE

```
if(numero == 0) //La condicion indica que tiene que ser igual a Cero
{
    cout<<"El Numero Ingresado es Igual a Cero";
}
else
{
    if(numero > 0) // la condicion indica que tiene que ser mayor a Cero
    {
        cout<<"El Numero Ingresado es Mayor a Cero";
    }
    else
    {
        if(numero < 0) // la condicion indica que tiene que ser menor a Cero
        {
            cout<<"El Numero Ingresado es Menor a Cero";
        }
    }
}
```

Sentencias SWITCH

```
switch (condición)
{
    case primer_caso:
        bloque de instrucciones 1
        break;

    case segundo_caso:
        bloque de instrucciones 2
        break;

    case caso_n:
        bloque de instrucciones n
        break;

    default: bloque de instrucciones por defecto
}
```

Ejemplo 1

```
switch (numero)
{
    case 0: cout << "numero es cero";
}
```

Ejemplo 2

```
switch (opcion)
{
    case 0: cout << "Su opcion es cero"; break;
    case 1: cout << "Su opcion es uno"; break;
    case 2: cout << "Su opcion es dos";
}
```

Ejemplo 3

```
switch (opcion)
{
    case 1: cout << "Su opcion es 1"; break;
    case 2: cout << "Su opcion es 2"; break;
    case 3: cout << "Su opcion es 3"; break;
    default: cout << "Elija una opcion entre 1 y 3";
}
```

Sentencias FOR

Ejemplo 1:

```
for(int i=1; i<=10; i++)  
{  
    cout<<"Hola Mundo";  
}
```

Ejemplo 2:

```
for(int i=10; i>=0; i--)  
{  
    cout<<"Hola Mundo";  
}
```

Sentencias WHILE

```
while(condicion)  
{  
    código a Repetir  
}
```

donde:

1. condicion es la expresión a evaluar

Ejemplo 1:

```
int contador = 0;  
  
while(contador<=10)  
{  
    contador=contador+1;  
    cout<<"Hola Mundo";  
}
```