UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DEL PORTAL PARA LA CONSULTA DE TEMAS APROBADOS PARA SEMINARIOS Y/O TESIS DE LA DIVISIÓN DE INGENIERÍA ELÉCTRICA

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTAN:

ALVARO ALFONSO MONDRAGÓN SALAS

EDGAR SOLÍS TORRES

DIRECTOR DE TESIS:

ING. JOSÉ ARTURO ORIGEL COUTIÑO

MÉXICO, D.F., CIUDAD UNIVERSITARIA

2005

Dedicatorias y Agradecimientos

Principalmente quiero agradecerle a Dios por permitirme llegar hasta este día, le doy gracias por darme salud e inteligencia para terminar mi licenciatura.

También quiero agradecerle a mi familia, muy especialmente a mi mamá por apoyarme en todo este tiempo, y sobre todo por confiar siempre en mi, gracias a mis tías pues sin su apoyo no hubiera logrado nada.

Quiero darle las gracias a la UNAM, a la Facultad de Ingeniería y a todos mis maestros y sinodales por compartir sus conocimientos y su valioso tiempo conmigo.

Y por último quiero darle las gracias a mis amigos (tanto a los que conocí en la universidad como a los que conocí a lo largo de mi vida) que me apoyaron y que me impulsaron a seguir adelante aún en los momentos más difíciles, ustedes saben que pueden contar conmigo.

"Somos mortales por nuestros temores, pero inmortales por nuestras aspiraciones."

Alvaro Alfonso Mondragón Salas

Agradezco a Dios por permitirme vivir intensamente cada momento de mi vida y por cruzar en mi camino a las personas adecuadas en el momento adecuado.

Este trabajo y en general la culminación de esta etapa de mi vida los dedico de forma muy especial a mis padres. A mi madre, por siempre preocuparse en que no me faltará amor, cariño, comprensión, principios, educación y bienestar; por tener la fuerza necesaria para sacarme adelante a pesar de las adversidades; por ser fuente de inspiración en cada deseo de superación, ¡mis logros te pertenecen!. A mi padre por apoyarme en todo momento, por todo tu esfuerzo y empeño puesto en mi educación.

A mis hermanos: Rogelio, Noé y David por todo su apoyo, amistad, cariño, comprensión, por soportarme y compartir conmigo todos estos años. A Laura, que se ha vuelto como una hermana, por su apoyo incondicional en momentos difíciles. A Gael y Alexis por ponerle ese toque de alegría a mi vida. A mis tías Martha y María, a mis primos Mario y Fernando por su apoyo.

A Ixsacbé, ¡por ser tú!, por tu amistad y sobre todo por ser mi otro yo.

A mis amigos Ireri, Chío, Roberto (Ropic), Ernesto por escucharme y ser cómplices.

Sin formalismos y con toda sinceridad, quiero agradecer a todos mis amigos: Edgar, Ángel, Carlos, Alicia, Jacobo, Fernando, Constantino, Vianney, Axel, Roberto (Coria), Astrid, Oswaldo, Mónica, Lizelly, Nelly, Daniel, Iván y todos aquellos que escapan de mi mente en estos instantes por todos los momentos compartidos, por sus palabras de aliento, por su ayuda y por hacer que prevalezca una de las cosas más importantes de mi vida. ¡Gracias... totales!.

A Víctor Armenta, por brindarme su amistad y por compartir conmigo sus conocimientos.

A Itzel y a Rose por todo su apoyo y por reafirmarme que la amistad nada tiene que ver con el tiempo.

A Alfonso, por compartir conmigo este trabajo, muy en especial a su familia por todo el apoyo recibido.

Agradezco al Ing. José Arturo Origel Coutiño por su confianza y paciencia al dirigir esta tesis. Al personal de la Coordinación de Seminarios: Angie, Mayra, Claudia e Isa por su apoyo y paciencia para con nosotros. A los sinodales, por aceptar ser parte del jurado.

A mi alma mater y a la Facultad de Ingeniería por abrirme sus puertas, a todos mis profesores por la preparación recibida.

Edgar Solís Torres

Índice

Introducción

Actualmente la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, y en específico la División de Ingeniería Eléctrica a través de la Coordinación de Seminarios y Servicio Social, carece de un portal mediante el cual los alumnos que pertenecen a dicha División puedan consultar los temas aprobados para seminarios y/o tesis, así como el banco de temas históricos, lo cual, considerando las herramientas tecnológicas con las que se cuenta hoy en día, representa para la Coordinación de Seminarios un rezago importante en la forma en la que actualmente presta sus servicios, la cual hace más lentos y limitados muchos de los trámites referentes al proceso de titulación.

Con un portal que sea capaz de resolver dichas carencias, el proceso de consulta de temas, tanto vigentes como históricos, sería más eficiente, además de que se facilitaría la difusión de este tipo de información para la Coordinación y habría una mejor interacción con alumnos y profesores.

Lo que se pretende con el presente documento es describir el diseño, desarrollo e implementación del sistema. De igual forma, que el portal a desarrollar cuente con su respectiva documentación para su administración, mantenimiento y uso.

En el primer capítulo se da un panorama general de las funciones y la situación actual de la Coordinación de Seminarios y Servicio Social de la División de Ingeniería Eléctrica de la FI; así mismo se explica la necesidad de desarrollar el portal, los objetivos específicos y resultados esperados de éste.

En el segundo capítulo se da todo el sustento teórico y de investigación para poder llevar a cabo con éxito el desarrollo del sistema. Se hace un análisis de las diferentes metodologías para el desarrollo de software, se proporciona la justificación de las herramientas a utilizar entre otros aspectos.

En el tercer capítulo se describe el proceso seguido para obtener el sistema: análisis de antecedentes, análisis de las partes y funcionamiento del portal, diseño de la base de datos, implementación del sistema, pruebas para la verificación de su correcto funcionamiento o en su caso detección de errores y finalmente su instalación.

Al término de los capítulos se presentan las conclusiones haciendo alusión a los resultados obtenidos, satisfacción de la Coordinación, evaluación de la metodología utilizada. Así mismo se proporcionan los anexos que hacen referencia a la normatividad y lineamientos para la implementación de páginas web en la Facultad de Ingeniería, manual de usuario (partes, pantallas y navegación dentro del portal), manual de administración (para la actualización del portal), glosario de términos

utilizados en este documento y las fuentes utilizadas como apoyo para la elaboración de esta tesis.

Capítulo 1. Marco General

1.1 Coordinación de Seminarios y Servicio Social

La Coordinación de Seminarios y Servicio Social de la División de Ingeniería Eléctrica (encargada de coordinar académica y administrativamente las carreras de Ingeniero en Computación, Ingeniero Eléctrico y Electrónico e Ingeniero en Telecomunicaciones), de la Facultad de Ingeniería de la UNAM, tiene entre sus principales objetivos:

- Apoyar y promover la titulación de los alumnos de la División de Ingeniería Eléctrica con lo cual se concluye el ciclo de nivel licenciatura.

- Apoyar y promover el cumplimiento del Servicio Social como un requisito indispensable para los alumnos de la División de Ingeniería Eléctrica.

- Difundir entre la comunidad las estancias empresariales, prácticas profesionales y ofertas de trabajo que se reciben en la División de Ingeniería Eléctrica.

- Apoyar y promover el cumplimiento de la asignatura de Seminario contemplada en todos los planes de estudio de las carreras de la División de Ingeniería Eléctrica.

- Apoyar y promover la realización de los trabajos de tesis de los alumnos de la División de Ingeniería Eléctrica.

y como funciones:

- Coordinar administrativamente las acciones necesarias para realizar los trámites de:

    1. Cumplimiento del Servicio Social.
    2. Realización de los trabajos de tesis.
    3. Cumplimiento de la asignatura de Seminario.
    4. Estancias empresariales, prácticas profesionales y ofertas de trabajo.
    5. Titulación.

- Atender y orientar a los alumnos y profesores para proceder de acuerdo con los lineamientos establecidos en los trámites señalados.

- Proporcionar información estadística a la División de Ingeniería Eléctrica y a las autoridades de la Facultad de Ingeniería que así lo requieran.

- Interactuar con otras áreas afines y organizaciones para el logro de los objetivos planteados: Coordinación de Seminarios y Servicio Social, Administración Escolar, empresas e instituciones.

1.2 Situación actual

Actualmente la Coordinación de Seminarios y Servicio Social de la DIE cuenta con un sistema hecho en Microsoft Access y una interfaz en Visual Basic con el cual se administran todos los procesos referentes al cumplimiento del Servicio Social e inscripción a Seminarios y/o Tesis. Dicho sistema tiene la capacidad de generar reportes con información de los temas aprobados para seminarios y/o tesis y sus respectivos datos como son: Título, Director, Fecha de aprobación, Cupo, Vacantes, entre otros datos.

El proceso que actualmente sigue un alumno para consultar los temas que ya han sido aprobados (temas propuestos por algún profesor de la Facultad), con sus respectivos directores, cupo, vacantes, entre otros datos de interés, es acudir directamente a la ventanilla de la Coordinación y verificar los listados ahí publicados, para posteriormente proceder con los trámites de inscripción.

1.3 Necesidad del portal

La idea de realizar este portal surge de la necesidad de nosotros como alumnos, de poder consultar los temas aprobados para seminarios y/o tesis desde cualquier lugar que tenga conexión a Internet sin tener que acudir directamente a la ventanilla de la Coordinación, además de poder imprimir un reporte con los temas de nuestro interés y contar con información actualizada de nuevos temas, directores, cupo, vacantes, etc.

Como un segundo problema a resolver con este sistema está el implementar un buscador dentro del portal con el que podamos consultar el banco de temas históricos, pues existe el caso en el que los alumnos conjuntamente con un profesor o persona externa a la Facultad pueden proponer un tema a desarrollar. La problemática presentada en este caso consiste en que muchas veces se proponen temas que ya han sido desarrollados o que se encuentran en desarrollo y no existe forma alguna de verificar antes de que sea rechazado, si el tema que se está proponiendo es válido o no. Con la información proporcionada para temas históricos se pretende que tanto alumnos como directores cuenten con la suficiente información como para proponer un tema nuevo, o en el caso que ya se haya tratado el tema propuesto, poder consultar el trabajo por escrito del mismo para poder delimitarlo o proponer alguna mejora o actualización.

De igual forma se pretende que aquellos alumnos extemporáneos cuenten con información más digerible respecto a los temas a los que pueden inscribirse, para lo cual se implementará un apartado con información del Programa de Apoyo a la Titulación (PAT), y un listado con temas pertenecientes exclusivamente a este programa.

Por último, el implementar un buscador, también resuelve el problema de generar reportes con información específica de temas relacionados con algún tema o director, reportes que son requeridos administrativamente por la propia Coordinación, y a manera de estadísticas por la División de Ingeniería Eléctrica y autoridades de la Facultad de Ingeniería.

1.4 Objetivos

Objetivo general

Desarrollar un portal para la consulta de temas aprobados para seminarios y/o tesis de la División de Ingeniería Eléctrica.

Objetivos particulares

- Desarrollar un portal que tenga como principal objetivo que tanto alumnos como profesores puedan consultar los temas aprobados para seminarios y/o tesis de la DIE con información referente a los mismos (Título, Director, Fecha de aprobación, Cupo/Vacantes, etc.).

- Desarrollar un portal mediante el cual se pueda consultar el banco de temas históricos y directores con el propósito de poder proponer con conocimiento de causa algún tema.

- Desarrollar un sistema que sea capaz de proporcionar reportes con información específica de temas y directores con fines administrativos y estadísticos para el personal de la Coordinación y de la propia Facultad.

Capítulo 2. Marco Teórico y Conceptual

2.1 Ingeniería del software

El software de computadora, se ha convertido en el *alma mater*. Es la máquina que conduce a la toma de decisiones comerciales. Sirve como la base de investigación científica moderna y de resolución de problemas de ingeniería. Es el factor clave que diferencia los productos y servicios modernos. Está inmerso en sistemas de todo tipo: de transportes, médicos, de telecomunicaciones, militares, procesos industriales, entretenimientos, productos de oficina…, la lista es casi interminable.

Los programas informáticos están omnipresentes, y el público los ve como un hecho tecnológico de la vida. En muchos ejemplos, las personas dejan su trabajo, bienestar, seguridad, entretenimiento, decisiones y sus propias vidas en manos del software informático.

La tecnología que deberían utilizar aquellos que construyen software informático, aquellos que lo deben hacer bien, la tecnología que acompaña a un proceso, un juego de métodos y un conjunto de herramientas es a lo que se llama *ingeniería de software*.

El software se compone de programas, datos y documentos. Cada uno de estos elementos componen una configuración que se crea como parte del proceso de la ingeniería de software. El intento de la ingeniería de software es proporcionar un marco de trabajo para construir software con mayor calidad.

2.2 El software como producto

Hoy en día el software tiene un doble papel. Es un producto y, al mismo tiempo, el vehículo para hacer entrega de un producto. Como producto, hace entrega de la potencia informática del hardware informático. Como vehículo utilizado para hacer entrega del producto, el software actúa como la base de control de la computadora (sistemas operativos), la comunicación de información (redes), y la creación y control de otros programas (herramientas de software y entornos).

El software hace entrega de lo que muchos creen que será el producto más importante del siglo veintiuno, la información. El software transforma datos personales para que los datos sean más útiles en un contexto local; gestiona información comercial para mejorar la competitividad; proporciona el acceso a redes

de información para todo el mundo; y también proporciona el medio de adquirir información en todas sus formas.

2.3 Definición de software

La descripción de software en un libro de texto podría tomar la forma siguiente: *el software es (1) instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados, (2) estructuras de datos que permiten a los programas manipular adecuadamente la información, y (3) documentos que describen la operación y el uso de programas*. No hay duda de que podrían ofrecerse otras definiciones más completas, pero nosotros necesitamos algo más que una definición formal.

2.3.1 Características del software

Para poder comprender lo que es software (y consecuentemente la ingeniería de software), es importante examinar las características del software que lo diferencian de otras cosas que los hombres pueden construir. Cuando se construye hardware, el proceso creativo humano (análisis, diseño, construcción, prueba) se traduce finalmente en una forma física.

El software es un elemento del sistema que es lógico, en lugar de físico. Por tanto el software tiene unas características considerablemente distintas a las del hardware:

1. *El software se desarrolla, no se fabrica en un sentido clásico*. Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente diferentes. En ambas actividades la buena calidad se adquiere mediante un buen diseño, pero la fase de construcción del hardware puede introducir problemas de calidad que no existen (o son fácilmente corregibles) en el software. Ambas actividades dependen de las personas, pero la relación entre las personas dedicadas y el trabajo realizado es completamente diferente para el software. Ambas actividades requieren la construcción de un <<producto>>, pero los métodos son diferentes.

2. *El software no se <<estropea>>*. El hardware exhibe relativamente muchos fallos al principio de su vida (estos fallos son atribuibles normalmente a defectos del diseño o de la fabricación); una vez corregidos los defectos, la tasa de fallos cae hasta un nivel estacionario donde permanece durante un cierto período de tiempo. Sin embargo, conforme pasa el tiempo, los fallos vuelven a presentarse a medida que los componentes del hardware sufren los

efectos acumulativos de la suciedad, la vibración, los malos tratos, las temperaturas extremas y muchos otros males externos. Sencillamente, el hardware comienza a estropearse (Figura 2.1).

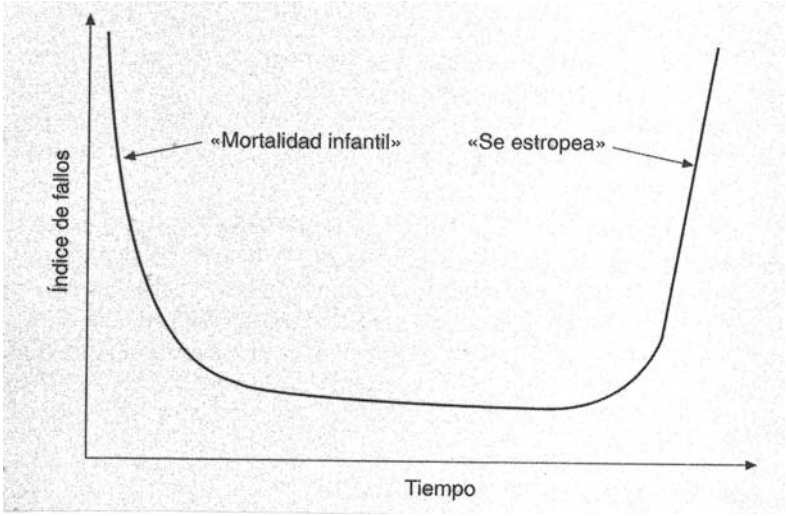


Figura 2.1 Curva de fallos del hardware

El software no es susceptible a los males del entorno que hacen que el hardware se estropee. Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen, suponiendo que no se introducen nuevos errores, la curva se aplana (Figura 2.2).

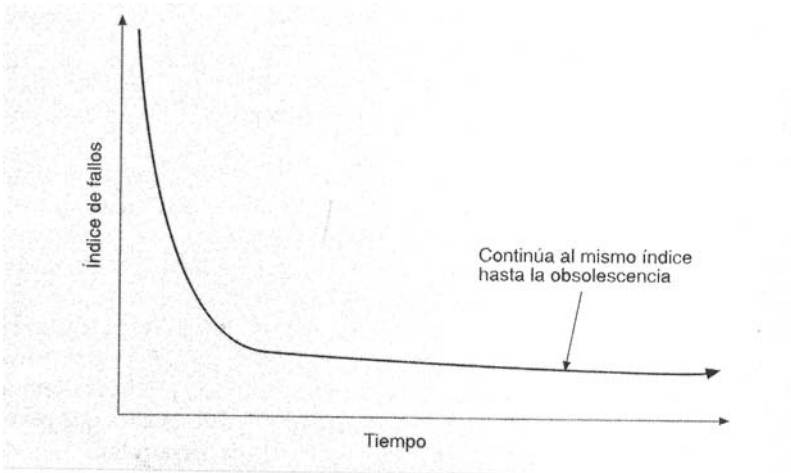


Figura 2.2 Curva de fallos del software (idealizado)

Sin embargo, la implicación es clara, el software no se estropea. ¡Pero se deteriora!. Esto que parece una contradicción, puede comprenderse mejor considerando la Figura 2.3. Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que

se introduzcan nuevos defectos, haciendo que la curva de fallos tenga picos. Antes de que la curva pueda volver al estado estacionario original, se solicita otro cambio, haciendo que de nuevo se cree otro pico. Lentamente, el nivel mínimo de fallos comienza a crecer; el software se va deteriorando debido a los cambios.

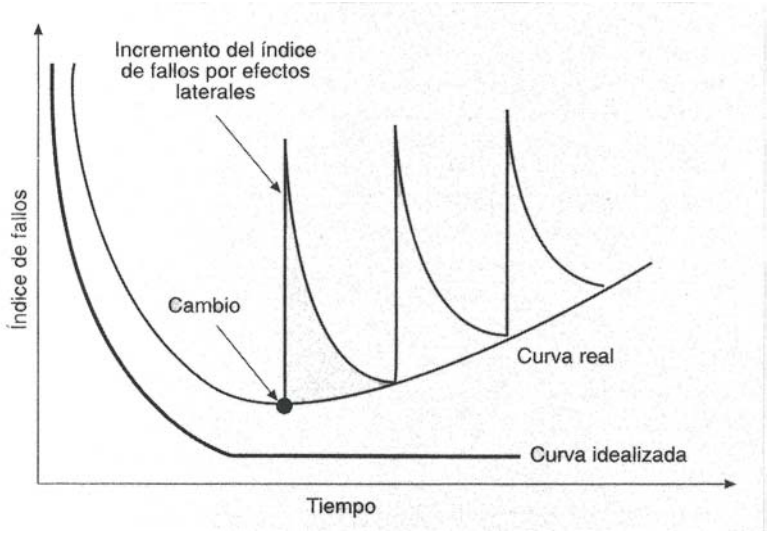


Figura 2.3 Curva real de fallos del software

2.3.2 Aplicaciones del software

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo). El contenido y determinismo de la información son factores importantes a considerar para determinar la naturaleza de una aplicación de software. El contenido se refiere al significado y a la forma de la información de entrada y de salida. El determinismo de la información se refiere a la predecibilidad del orden y del tiempo de llegada de los datos. Un programa de ingeniería acepta datos que están en un orden predefinido, ejecuta el algoritmo sin interrupción y produce los datos resultantes en un informe o formato gráfico. Se dice que tales aplicaciones son determinadas. Un sistema operativo multiusuario, por otra parte, acepta entradas que tienen un contenido variado y que se producen en instantes arbitrarios, ejecuta algoritmos que pueden ser interrumpidos por condiciones externas y produce una salida que depende de una función del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

Software de sistemas. El software de sistemas es un conjunto de programas que han sido escritos para servir a otros programas. El área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

Software de tiempo real. El software que mide/analiza/controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Hay que tener en cuenta que el término <<tiempo real>> tiene un significado diferente de <<interactivo>> o <<tiempo compartido>>. Un sistema de tiempo real debe responder dentro de unas ligaduras estrictas de tiempo. El tiempo de respuesta de un sistema interactivo (o de tiempo compartido) puede ser normalmente sobrepasado sin que se produzca ningún desastre.

Software de gestión. El procesamiento de información comercial constituye la mayor de las áreas de aplicación del software. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamiento de datos, las aplicaciones de software de gestión también realizan cálculo interactivo.

Software de ingeniería y científico. El software de ingeniería y científico está caracterizado por los algoritmos de <<manejo de números>>. Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática.

Software empotrado. Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo.

Software de computadoras personales. El mercado del software de computadoras personales ha germinado en la pasada década. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, aplicaciones financieras, de negocios y personales, y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

Software de inteligencia artificial. El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Actualmente, el área más activa de la IA es la de los sistemas expertos, también llamados sistemas basados en el conocimiento. Sin embargo, otras áreas de aplicación para el software de IA es el reconocimiento de patrones (imágenes y voz), la prueba de teoremas y los juegos. En los últimos años se ha desarrollado una nueva rama del software de IA llamada redes neuronales artificiales.

2.4 El software como proceso

Definimos un proceso de software como un marco de trabajo de las tareas que se requieren para construir software de alta calidad. Un proceso de software define el enfoque que se toma cuando el software es tratado por la ingeniería, pero la tecnología del software también acompaña a las tecnologías que pueblan el proceso (métodos técnicos y herramientas automatizadas).

2.5 Ingeniería del software: una tecnología estratificada

Definición propuesta por Fritz Bauer:

[La ingeniería del software] es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales.

Esta definición no dice mucho sobre los aspectos técnicos de la calidad del software; no se enfrenta directamente con la necesidad de la satisfacción del cliente o de la entrega oportuna del producto; omite la mención de la importancia de mediciones y métricas; tampoco expresa la importancia de un proceso avanzado.

El IEEE ha desarrollado una definición más completa:

Ingeniería de software: (1) La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software. (2) El estudio de enfoques como en (1).

2.5.1 Proceso, métodos y herramientas

La ingeniería del software es una tecnología multicapa (Figura 2.4). Cualquier enfoque de ingeniería (incluida ingeniería del software) debe descansar sobre un empeño de organización de calidad. La gestión total de calidad y las filosofías similares fomentan una cultura continua de mejoras de procesos, y es esta cultura la que conduce últimamente al desarrollo de enfoques cada vez más robustos para la ingeniería del software. Los cimientos que son la base de la ingeniería del software están orientados hacia la calidad.

El fundamento de la ingeniería del software es la capa *proceso*. El proceso define un marco de trabajo para un conjunto de *áreas clave de proceso* que se debe establecer para la entrega efectiva de la tecnología de la ingeniería del software.

Los *métodos* de la ingeniería del software indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento.

Las *herramientas* de la ingeniería del software proporcionan un soporte automático o semi-automático para el proceso y para los métodos.

Figura 2.4 Capas de ingeniería del software

2.5.2 Una visión general de la ingeniería del software

La ingeniería es el análisis, diseño, construcción, verificación y gestión de entidades técnicas (o sociales). Con independencia de la entidad a la que se va aplicar ingeniería, se deben cuestionar y responder las siguientes preguntas:

- ¿Cuál es el problema a resolver?
- ¿Cuáles son las características de la entidad que se utiliza para resolver el problema?
- ¿Cómo se realizará la entidad (y la solución)?
- ¿Cómo se construirá la entidad?
- ¿Qué enfoque se va a utilizar para no contemplar los errores que se cometieron en el diseño y en la construcción de la entidad?
- ¿Cómo se apoyará la entidad cuando varios usuarios soliciten correcciones, adaptaciones y mejoras de la entidad?

El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto. Cada fase se enfrenta con una o varias cuestiones de las destacadas anteriormente.

La *fase de definición* se centra sobre el *qué*. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto.

La *fase de desarrollo* se centra en el *cómo*. Es decir, durante el desarrollo un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función como una arquitectura del software, cómo han de implementarse detalles procedimentales, cómo han de caracterizarse las interfaces, cómo ha de traducirse el diseño en un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba.

La *fase de mantenimiento* se centra en el *cambio* que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente.

2.6 Modelos de proceso de software

Para resolver los problemas reales de una industria, un ingeniero del software o un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos, capas de herramientas descritos en la sección 2.5.1 y las fases genéricas discutidas en la sección 2.5.2. Esta estrategia a menudo se llama *modelo de proceso* o *paradigma de ingeniería del software*. Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

Todo el desarrollo del software se puede caracterizar como un *bucle* de resolución de problemas (Figura 2.5) en el que se encuentran cuatro etapas distintas: status quo, definición de problemas, desarrollo técnico e integración de soluciones. *Status quo* <<representa el estado actual de sucesos>>; la definición de problemas identifica el problema específico a resolverse; el desarrollo técnico resuelve el problema a través de la aplicación de alguna tecnología, y la integración de soluciones ofrece los resultados.

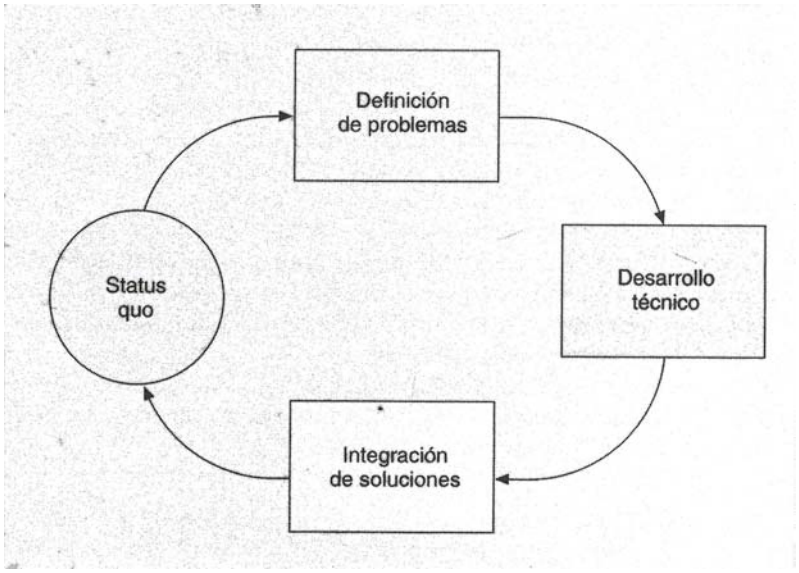


Figura 2.5 Las fases de un bucle de resolución de problemas

2.7 El modelo lineal secuencial

La Figura 2.6 ilustra el modelo *lineal secuencial* para la ingeniería del software. Llamado algunas veces <<ciclo de vida básico>> o <<modelo en cascada>>, el modelo lineal secuencial sugiere un enfoque sistemático, secuencial del desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

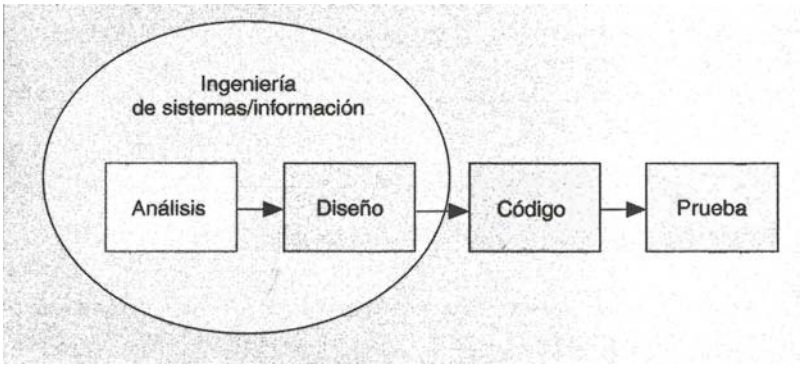


Figura 2.6 El modelo lineal secuencial

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software. Sin embargo, la crítica del paradigma ha puesto en duda su eficacia. Entre los problemas que se encuentran algunas veces en el modelo lineal secuencial se incluyen:

1. Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo. Aunque el modelo lineal puede acoplar interacción, lo hace indirectamente.

2. A menudo es difícil que el cliente exponga explícitamente todos los requisitos.

3. El cliente debe tener paciencia. Una versión de trabajo del (los) programa(s) no estará disponible hasta que el proyecto esté muy avanzado.

4. Los responsables del desarrollo del software siempre se retrasan innecesariamente.

Cada uno de estos errores es real. Sin embargo, el paradigma del ciclo de vida clásico tiene lugar definido e importante en el trabajo de la ingeniería del software. Proporciona una plantilla en la que se encuentran métodos para análisis, diseño, codificación, pruebas y mantenimiento. El ciclo de vida clásico sigue siendo el modelo de proceso más extensamente utilizado por la ingeniería del software. Pese a tener debilidades, es significativamente mejor que un enfoque hecho al azar para el desarrollo del software.

2.8 El modelo de construcción de prototipos

Un cliente a menudo define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento, o salida.

El paradigma de construcción de prototipos (Figura 2.7) comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos, y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un <<diseño rápido>>. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y lo utiliza para refinar los requisitos del software a desarrollar. La interacción ocurre cuando el prototipo satisface las necesidades del cliente, a la vez que permite que el desarrollador comprenda mejor lo que se necesita hacer.

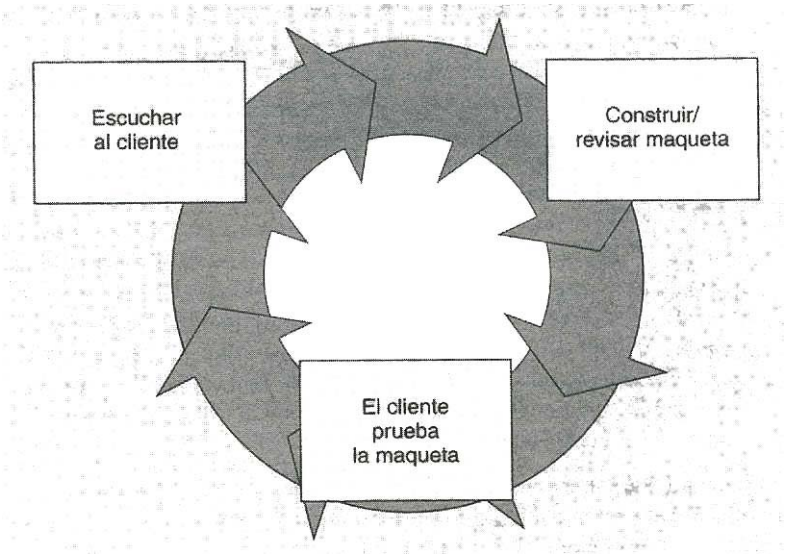


Figura 2.7 El paradigma de construcción de prototipos

Aunque pueden surgir problemas, la construcción de prototipos puede ser un paradigma efectivo para la ingeniería del software. La clave es definir las reglas del juego al comienzo; es decir, el cliente y el desarrollador se deben poner de acuerdo en que el prototipo se construya para servir como un mecanismo de definición de requisitos. Entonces se descarta (al menos en parte), y se realiza la ingeniería del software con una visión hacia la calidad y la facilidad de mantenimiento.

2.9 El modelo DRA

El Desarrollo Rápido de Aplicaciones (DRA) (Rapid Application Development, RAD) es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. El modelo DRA es una adaptación a <<alta velocidad>> del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un <<sistema completamente funcional>> dentro de períodos cortos de tiempo (Figura 2.8).

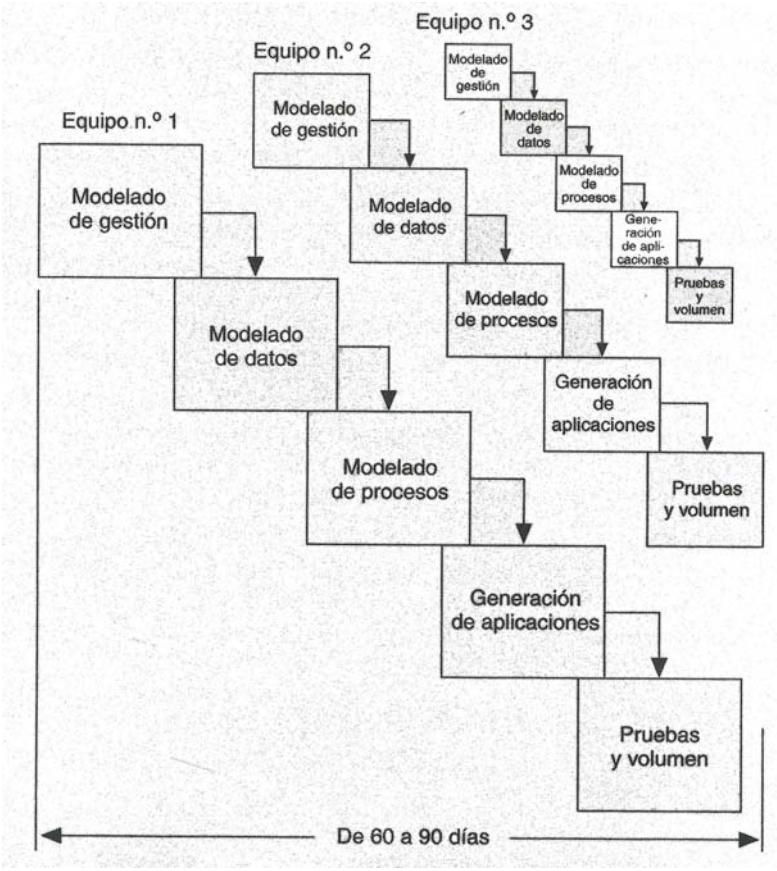


Figura 2.8 El modelo DRA

Al igual que todos los modelos de proceso, el enfoque DRA tiene inconvenientes:

1. Para proyectos grandes aunque por escalas, el DRA requiere recursos humanos suficientes como para crear el número correcto de equipos DRA.

2. DRA requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso, por ninguna de las partes constituyentes, los proyectos DRA fracasarán.

No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modularizar adecuadamente, la construcción de los componentes necesarios para DRA será problemático. Si está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistemas, el enfoque DRA puede que no funcione. DRA no es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el nuevo software requiere un alto grado de interoperatividad con programas de computadora ya existentes.

2.10 Modelos de procesos evolutivos de software

El modelo lineal secuencial se diseña para el desarrollo en línea recta. En esencia, este enfoque en cascada asume que se va entregar un sistema completo una vez que la secuencia lineal haya finalizado. El modelo de construcción de prototipos se diseña para ayudar al cliente (o al que desarrolla) a comprender los requisitos. En general, no se diseña para entregar un sistema de producción. En ninguno de los paradigmas de ingeniería del software se tiene en cuenta la naturaleza evolutiva del software.

Los modelos evolutivos son iterativos. Se caracterizan por la forma en que permiten a los ingenieros del software desarrollar versiones cada vez más completas del software.

2.10.1 El modelo incremental

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetitivamente) con la filosofía interactiva de construcción de prototipos. Como muestra la Figura 2.9, el modelo incremental aplica secuencias lineales de forma sorprendente de la misma forma que progresa el tiempo en el calendario. Cada secuencia lineal produce un <<incremento>> del software.

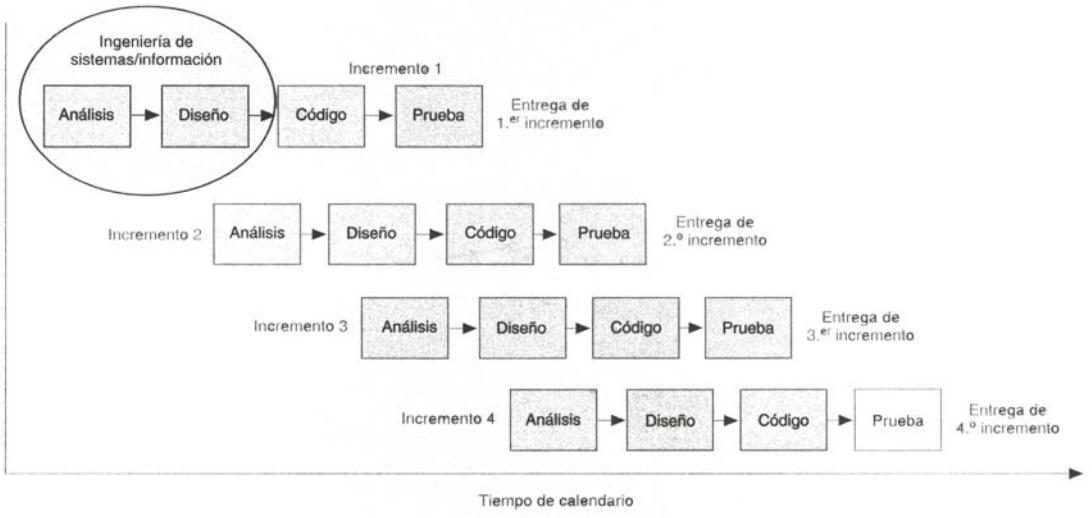


Figura 2.9 El modelo incremental

El modelo de proceso incremental, como la construcción de prototipos y otros enfoques evolutivos, es interactivo por naturaleza. Pero a diferencia de la construcción de prototipos, el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son versiones

<<desmontadas>> del producto final, pero proporcionan la capacidad que sirve al usuario y también proporciona una plataforma para la evaluación por parte del usuario.

El desarrollo incremental es particularmente útil cuando la dotación de personal no está disponible para una implementación completa en cuanto a la fecha límite de gestión que se haya establecido para el proyecto. Los primeros incrementos se pueden implementar con menos personas. Si el producto central es bien recibido, se puede añadir más personal (si se requiere) para implementar el incremento siguiente.

2.10.2 El modelo en espiral

El *modelo en espiral*, propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. En el modelo en espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas de ingeniería del sistema.

El modelo en espiral se divide en un número de *actividades estructurales*, también llamadas *regiones de tareas*. La Figura 2.10 representa un modelo en espiral que contiene seis regiones de tareas:

- **Comunicación con el cliente**. Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.

- **Planificación**. Las tareas requeridas para definir recursos, el tiempo y otras informaciones relacionadas con el proyecto.

- **Análisis de riesgos**. Las tareas requeridas para evaluar riesgos técnicos y de gestión.

- **Ingeniería**. Las tareas requeridas para construir una o más representaciones de la aplicación.

- **Construcción y adaptación**. Las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.

- **Evaluación del cliente**. Las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

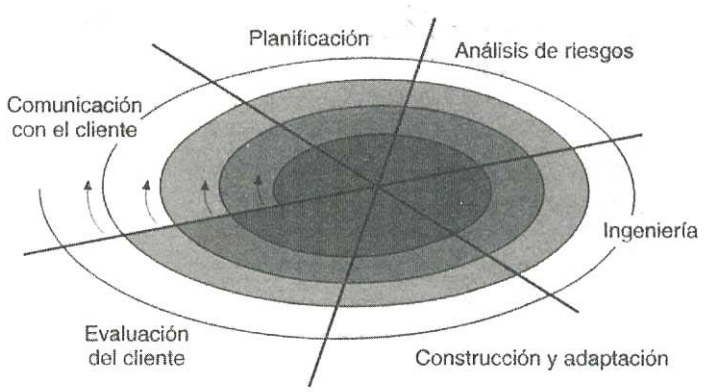


Figura 2.10 Modelo en espiral típico

Cuando empieza este proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral produce el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software.

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.

Al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede resultar difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la evaluación del riesgo, y cuenta con esta habilidad para el éxito. Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas.

2.10.3 El modelo de ensamblaje de componentes

Las tecnologías de objetos proporcionan el marco de trabajo técnico para un modelo de proceso basado en componentes para la ingeniería del software. El paradigma de orientación a objetos enfatiza la creación de clases que encapsulan tanto los datos como los algoritmos que se utilizan para manejar los datos. Si se diseñan y se implementan adecuadamente, las clases orientadas a objetos son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadora.

El modelo de ensamblaje de componentes (Figura 2.11) incorpora muchas de las características del modelo en espiral. Es evolutivo por naturaleza, y exige un

enfoque interactivo para la creación del software. Sin embargo, el modelo ensamblador de componentes configura aplicaciones desde componentes preparados de software (algunas veces llamados <<clases>>).

Las clases (llamadas componentes en la Figura 2.11) creadas en los proyectos de ingeniería del software anteriores se almacenan en una *biblioteca de clases* o *depósito*. Una vez identificadas las clases candidatas, la biblioteca de clases se examina para determinar si estas clases ya existen. En caso de que así fuera, se extraen de la biblioteca y se vuelven a utilizar. Si una clase candidata no reside en la biblioteca, se aplican los métodos orientados a objetos. Se compone así la primera interacción de la aplicación a construirse, mediante las clases extraídas de la biblioteca y las clases nuevas construidas para cumplir las necesidades únicas de la aplicación. El flujo del proceso vuelve a la espiral y volverá a introducir por último la iteración ensambladora de componentes a través de la actividad de ingeniería.

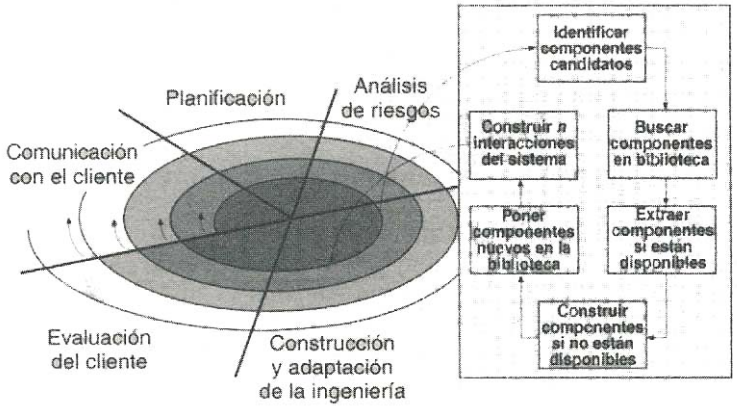


Figura 2.11 Modelo de ensamblaje de componentes

El modelo ensamblador de componentes lleva a la reutilización del software, y la reutilización proporciona beneficios a los ingenieros del software. Según estudios de reutilización, el ensamblaje de componentes lleva a una reducción del 70 por ciento de tiempo de ciclo de desarrollo, un 84 por ciento del costo del proyecto y un índice de productividad del 26,2.

2.10.4 El modelo de desarrollo concurrente

El modelo de proceso concurrente se puede representar en forma de esquema como una serie de actividades técnicas importantes, tareas y estados asociados a ellas. Por ejemplo, la actividad de *ingeniería* definida para el modelo en espiral, se lleva a cabo invocando las tareas siguientes: modelado de construcción de prototipos y/o análisis, especificación de requisitos, y diseño.

La Figura 2.12 proporciona una representación esquemática de una actividad dentro del modelo de proceso concurrente. La actividad –análisis- se puede encontrar en uno de los estados destacados anteriormente en cualquier momento dado. De forma similar, otras actividades (p. ej.: diseño o comunicación con el cliente) se pueden representar de una forma análoga. Todas las actividades existen concurrentemente, pero residen en estados diferentes. La actividad de *análisis* (que existía en el estado **ninguno** mientras que comenzaba la comunicación inicial con el cliente) ahora hace una transición al estado **bajo desarrollo**. Sin embargo, si el cliente indica que se deben hacer cambios en requisitos, la actividad análisis cambia del estado **bajo desarrollo** al estado **cambios en espera**.

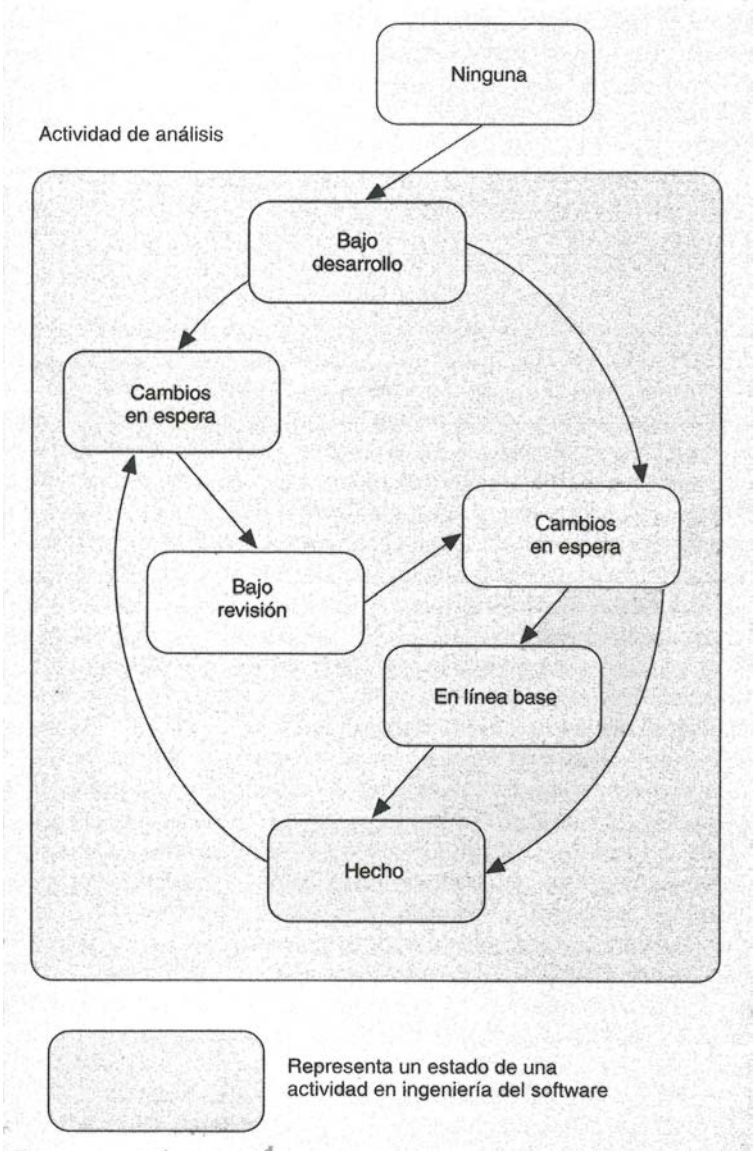


Figura 2.12 Un elemento del modelo de proceso concurrente

El modelo de proceso concurrente se utiliza a menudo como el paradigma de desarrollo de aplicaciones cliente/servidor. Un sistema cliente/servidor se compone de un conjunto de componentes funcionales. Cuando se aplica a cliente/servidor, el modelo de proceso concurrente define actividades en dos dimensiones: una *dimensión de sistemas* y una *dimensión de componentes*. Los aspectos del nivel de sistemas se afrontan mediante tres actividades: *diseño*, *ensamblaje* y *uso*. La dimensión de componentes se afronta con dos actividades: *diseño* y *realización*. La concurrencia se logra de dos formas: (1) las actividades de sistemas y de componentes ocurren simultáneamente y pueden modelarse con el enfoque orientado a objetos descrito anteriormente; (2) una aplicación cliente/servidor típica se implementa con muchos componentes, cada uno de los cuales se puede diseñar y realizar concurrentemente.

En realidad, el modelo de proceso concurrente es aplicable a todo tipo de desarrollo de software y proporciona una imagen exacta del estado actual de un proyecto. En vez de confinar las actividades de ingeniería del software a una secuencia de sucesos, define una red de actividades. Todas las actividades de la red existen simultáneamente con otras. Los sucesos generados dentro de una actividad dada o en algún otro lugar en la red de actividad inician las transiciones entre los estados de una actividad.

2.11 El modelo de métodos formales

El modelo de métodos formales acompaña a un conjunto de actividades que conducen a la especificación matemática del software de computadora. Los métodos formales permiten que un ingeniero del software especifique, desarrolle y verifique un sistema basado en computadora aplicando una notación rigurosa y matemática.

Cuando se realizan métodos formales durante el desarrollo, proporcionan un mecanismo para eliminar muchos de los problemas que son difíciles de superar con paradigmas de la ingeniería del software. La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen más fácilmente, no mediante una revisión a propósito para el caso, sino mediante la aplicación del análisis matemático.

Aunque todavía no hay un enfoque establecido, los modelos de métodos formales ofrecen la promesa de un software libre de defectos. Sin embargo, se ha hablado de una gran preocupación sobre su aplicabilidad en un entorno de gestión:

1. El desarrollo de modelos formales actualmente es bastante caro y lleva mucho tiempo.

2. Se requiere un estudio caro porque pocos responsables del desarrollo de software tienen los antecedentes necesarios para aplicar métodos formales.

3. Es difícil utilizar los modelos como un mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos.

2.12 Técnicas de cuarta generación

El término <<técnicas de cuarta generación>> (T4G) abarca un amplio espectro de herramientas de software que tienen algo en común: todas facilitan al ingeniero del software, la especificación de algunas características del software de alto nivel. Luego, la herramienta genera automáticamente el código fuente basándose en la especificación del técnico. El paradigma T4G para la ingeniería del software se orienta hacia la posibilidad de especificar el software usando formas de lenguaje especializado o notaciones gráficas que describan el problema que hay que resolver en términos que los entienda el cliente. Actualmente, un entorno para el desarrollo de software que soporte el paradigma T4G puede incluir todas o algunas de las siguientes herramientas: lenguajes no procedimentales de consulta a bases de datos, generación de informes, manejo de datos, interacción y definición de pantallas, generación de códigos, capacidades gráficas de alto nivel y capacidades de hoja de cálculo.

Para aplicaciones pequeñas, se puede ir directamente desde el paso de recolección de requisitos al paso de implementación, usando un *lenguaje de cuarta generación* no procedimental (L4G). Sin embargo, es necesario un mayor esfuerzo para desarrollar una estrategia de diseño para el sistema, incluso si se utiliza un L4G. El uso de T4G sin diseño (para grandes proyectos) causará las mismas dificultades (poca calidad, mantenimiento pobre, mala aceptación por el cliente) que se encuentran cuando se desarrolla software mediante los enfoques convencionales.

Al igual que todos los paradigmas del software, el modelo T4G tiene ventajas e inconvenientes. Los defensores aducen reducciones drásticas en el tiempo de desarrollo del software y una mejora significativa en la productividad de la gente que construye el software. Los detractores aducen que las herramientas actuales de T4G no son más fáciles de utilizar que los lenguajes de programación, que el código fuente producido por tales herramientas es <<ineficiente>> y que el mantenimiento de grandes sistemas de software desarrollados mediante T4G, es cuestionable.

Resumiendo, las técnicas de cuarta generación ya se han convertido en una parte importante del desarrollo del software. Cuando se combinan con enfoques de ensamblaje de componentes, el paradigma T4G se puede convertir en el enfoque dominante hacia el desarrollo del software.

2.13 Un enfoque actual sobre la calidad del software

Uno de los principales problemas que enfrenta actualmente el campo de la computación es sin duda alguna la calidad del software. Desde la década de los 70's este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales:

1. ¿Cómo obtener un software con calidad?
2. ¿Cómo evaluar la calidad del software?

Ambas interrogantes conllevan amplias respuestas, pero están estrechamente ligadas con el concepto de la calidad del software, que es el resultado de la primera y la fuente de la segunda.

El concepto de calidad en los productos de software debe formularse de forma particular. Primero es conveniente indicar sus características diferenciadoras frente a otros productos: el software se desarrolla, no se fabrica en el sentido clásico; es inmaterial y no se estropea con el uso o el tiempo (aunque tiene un ciclo de vida); su fiabilidad es difícil de comprobar; la mayoría del software se construye a medida y necesita de actualización permanente; es dependiente del entorno donde se ejecuta.

La calidad en la ingeniería de software es una disciplina cuyo horizonte de madurez está aun lejos, y que se caracteriza por la proliferación de normas, métodos y herramientas incompatibles entre sí.

2.14 Definición de calidad en la ingeniería del software

"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario". (IEEE, Std. 610-1990).

"Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos explícitos no establecidos formalmente, que desea el usuario". (Pressman, 1998).

La calidad en la ingeniería del software, que depende en gran medida de la pericia del equipo que lo desarrolla, puede definirse como un conjunto de características o cualidades que lo caracterizan y que determinan su utilidad y existencia, tales como: eficiencia, fiabilidad, usabilidad, funcionalidad, mantenibilidad, portabilidad, seguridad, integridad, etc., variando la importancia de cada una de ellas de un producto a otro.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Un software elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de software para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

2.15 ¿Cómo obtener un software de calidad?

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La política establecida debe estar sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.

El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del software.

El principio administrativo contempla las funciones de planificación y control del desarrollo del software, así como la organización del ambiente o centro de ingeniería de software.

El principio ergonómico define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación.

En las empresas de Ingeniería de software, la calidad se obtiene mejorando día a día el proceso de producción, mantenimiento y gestión del software. Para optimizar la calidad de los productos y/o servicios es preciso conocer al cliente y sus necesidades, conocer a la competencia y poseer un modelo de calidad. Esto último

permitirá incrementar la fiabilidad, reducir el mantenimiento, aumentar la satisfacción del cliente, mejorar la dirección del proyecto, detectar errores pronto e incrementar el beneficio. Pero también la madurez de los procesos de desarrollo y las técnicas adecuadas para mejorarlos y tener una gestión adecuada de las mejoras de dichos procesos.

Hay que saber en qué nivel de madurez, en cuanto a la calidad, se encuentra la organización, para poder determinar qué tipo de acciones son las más adecuadas en cada momento. Una organización inmadura se caracteriza por:

1. Realizar procesos improvisados; incluso procesos especificados no son seguidos ni se exige su cumplimiento.
2. "Reaccionar"; los directivos "apagafuegos" se centran en resolver las crisis momentáneas.
3. Sobrepasar presupuestos y calendarios, pues no se basan en estimaciones reales.
4. Disminuir la calidad y la funcionalidad para cumplir fechas.
5. Carecer de bases objetivas para juzgar la calidad o para resolver problemas.
6. Reducir o eliminar actividades de mejora de la calidad (revisiones, pruebas, etc.) cuando los proyectos se retrasan.

Crosby propone un modelo de 5 etapas de madurez de la gestión de calidad en la empresa:

1. Incertidumbre: realidad confusa y sin compromiso.
2. Despertar: reconocimiento de la importancia de la gestión de la calidad pero sin un compromiso de invertir en ella.
3. Ilustración: compromiso de mejora, enfrentando los problemas sin desviar la acción a otros.
4. Sabiduría: participación, medición precisa e intentos de hacer permanentes las mejoras logradas.
5. Certeza: gestión de la calidad como parte esencial de los sistemas de la empresa.

El "Software Engineering Institute" de la Universidad de Carnegie - Mellon, se basó en el modelo de "madurez" de Phillip Crosby para crear un método para valorar el proceso de desarrollo de software: Modelo de Madurez de la Capacidad para Crear Software, CMM (Capability Maturity Model)

El CMM: Es un método. Por una parte se centra en el estudio de los procesos de software, en contraposición al interés por los productos software. Ejemplo: definición y desarrollo de los requisitos del software; generación de datos de prueba; planificación de la instalación del software. Por otra, se fundamenta en que la "madurez" del proceso es un indicador para construir software de calidad.

El método tiene 5 niveles de "madurez" y 18 áreas clave, cada una de éstas incluyen prácticas clave, que a su vez encierran preguntas concretas.

El modelo, basado en sondeos, permite que las empresas puedan conocer en qué nivel se encuentra actualmente su proceso de producción de software, qué tipo de mejoras prioritarias se deben realizar y en qué aspectos o actividades críticas del proceso de desarrollo pueden intervenir con cierta garantía de éxito (estrategia de mejora del proceso).

Finalmente es recomendable tener en cuenta las siguientes "medidas" a aplicar para asegurar la calidad de los productos software:

- En el contrato de software: formular claramente y de forma precisa las responsabilidades, productos a entregar, etc. del proveedor; usar cláusulas estándar y listas de comprobación; revisar los borradores del contrato (comprador económico, auditor, abogado, etc.); negociar el borrador del contrato con el proveedor; obtener la orden del contrato firmada; modificar el contrato en caso de "problemas" serios.

- En el producto software: definir los requerimientos de calidad del producto; utilizar herramientas apropiadas para describir los requerimientos; evaluar la viabilidad; guías de programación; guías de documentación; documentación técnica consistente; gestión de la configuración; revisión de la documentación técnica; revisión de código; pasos de test; predicción del rendimiento; formación técnica de los miembros del equipo.

- En el proceso de software: proveer con antelación suficiente los recursos necesarios: personas, hardware, software, herramientas, etc.; estructurar el proceso de software por fases; descripción clara del trabajo de cada miembro del equipo; reuniones del proyecto planificadas; reuniones de decisión de fase planificadas; control periódico de las tendencias y costos de los hitos; control de resultados e información de los hitos; monitorización periódica del riesgo y su prevención; información y motivación de los miembros del equipo; formación de los miembros del equipo.

- En la documentación de software: estándares de contenido: estructura, sumario, glosario, índice, etc.; estándar de presentación: formato de la hoja, identificación, clasificación, paginación, estado, etc.; instrucciones al equipo: guías de estilo de formulación, visualización, ejemplos; suministro de herramientas; administración de los documentos de forma profesional; revisiones / versiones.

La calidad es hoy una de las mayores ventajas competitivas para las empresas desarrolladoras de software.

La calidad es un concepto vacío si no se tienen en cuenta las necesidades reales de los clientes. Así pues, algunas definiciones de calidad serían: calidad es el cumplimiento de los requisitos del cliente; calidad es satisfacer al cliente; o calidad es lo que el cliente dice o piensa que debe ser.

Estas definiciones necesitan desarrollar los siguientes aspectos básicos: las expectativas del cliente deben ser traducidas en requisitos; desarrollar un sistema para cumplir con dichos requisitos de forma previsible, a la primera; establecer una metodología de actuación para llegar a "cero defectos" en el trabajo; disponer de un procedimiento de medida para el control de la satisfacción del cliente. Previamente a ello, será preciso conocer perfectamente y a fondo las capacidades de la empresa: su organización y procedimientos internos de gestión; sus métodos estandarizados; su sistema de control y seguimiento de los procesos de desarrollo; y sus herramientas de soporte.

Una organización no preparada para dar calidad, en el sentido esperado por sus clientes, caerá muy rápidamente en la situación de expectativas no cumplidas o no alcanzadas.

La calidad tiene muchos enemigos, entre los más significativos están: la inconsistencia, la falta de disciplina, los compromisos insostenibles y la impaciencia.

Para establecer un "movimiento" para la calidad, en las empresas, debe iniciarse un mecanismo que identifique las necesidades del cliente. No se debe presuponer nada y la mejor manera de conocer qué quieren y cómo lo quieren es preguntándoles directamente a ellos. Escucharles de forma activa, asesorarles y demostrarles que sus necesidades son prioritarias para la empresa.

Focalizar en la calidad, como mejora, nos lleva al concepto actual de gestión de la calidad, lo que implica tres tipos de actuaciones: la planificación de la calidad en el futuro; la implantación y el desarrollo de actividades y programas para la consecución de la calidad; el control de los resultados obtenidos.

La calidad dirigida a los resultados (incrementar el rendimiento, mejoras en el control de costos, incrementar el beneficio) es una estrategia que contempla: gestionar la calidad con ciclos de mejora cortos; buscar mejoras inmediatas y rápidas, las que tengan elevadas posibilidades de éxito; que estén unidas a los objetivos de la empresa; no invertir demasiado tiempo en instaurar procedimientos, sí en obtener resultados; implantar sólo los procedimientos absolutamente necesarios que aseguren formación, motivación, participación, compromiso, modelos y métricas.

Como resumen diremos que el esquema de trabajo de un equipo directivo comprometido con la calidad es:

Establecer métodos y mecanismos que permitan a la empresa conocer mejor las expectativas de los clientes.

- Evita que se produzca la fragmentación de la empresa.
- Se ocupa de establecer objetivos, normas y estándares de calidad.
- Gestiona al personal, convirtiéndolo en un eficaz y decidido apoyo a la calidad.
- Implanta eficaces sistemas de supervisión y control, orientados a la calidad.
- Incorpora la visión del consumidor para evitar deficiencias en el diseño.

- Estimula y desarrolla el trabajo en equipo.
- Establece eficaces mecanismos de “feedback”.
- Establece eficaces mecanismos de comunicación.

El modelo de Calidad Total postula que: la calidad es responsabilidad de todos, en especial de los niveles de dirección; la dirección debe convertir toda la empresa en un "sistema de calidad"; la calidad debe ser construida en todas las fases y procesos, empezando por el diseño; la empresa debe garantizar la calidad en las fases de uso, consumo o posesión del producto / servicio; la calidad se genera en todas las áreas de la organización; la calidad no la determina la empresa, la definen y califican los clientes (visión "desde fuera").

2.16 Diferencias de las WebApp respecto de otros sistemas de software

A continuación mostramos los rasgos distintivos de los sistemas de software que funcionan desde Internet, esto enfocándonos en la idea de que un sitio web ofrece el servicio y múltiples usuarios hacen uso de él.

Apoyándonos en R.S. Pressman y en nuestra experiencia, podemos afirmar que los principales rasgos distintivos para las aplicaciones web (WebApp) son:

Según el autor:

- Intensivas en red.
- Dirigidas por el contenido.
- En continua evolución.
- Inmediatez (el tiempo entre el concepto y su colaboración en el mercado es muy reducido).
- Seguridad: es imperativa, pues virtualmente todo el mundo puede acceder al sistema.
- Estética: para atraer al usuario y que regrese.

De nuestra experiencia personal:

- Ubicada en el esquema Cliente/Servidor, donde el cliente son los navegadores y el servidor puede ser un servidor o una red de servidores.
- Limitación de protocolo: HTTP, HTTPS, FTP.
- El cliente es la mayoría de las veces un navegador web, por tanto, el servidor envía sus resultados en un conjunto de lenguajes específicos: HTML, XML, JavaScript, etc.

2.17 ¿Qué es un sitio web?

Un sitio web es un documento electrónico que se utiliza para difundir información en Internet, y en el cual podemos colocar texto, imágenes y sonido, sin que llegue a convertirse en un multimedia.

Los sitios web se distribuyen globalmente a través de miles de sitios, de esta forma muchos usuarios pueden ver alguna página web desde cualquier parte del mundo.

La utilidad de los sitios web es inmensamente amplia, ya que las podemos utilizar como medio educativo, informativo, comercial, etc.; que se pueden presentar de una forma atractiva y dinámica, esto es, la información se puede actualizar con facilidad.

Los sitios web no generan costos de materiales, como los medios impresos, ni por número de copias.

2.18 Principios generales para el diseño de sitios web

Internet es un medio riquísimo tanto en posibilidades como en contenidos. La competitividad existente en la red de redes es enorme debido entre muchas razones a la relativa pequeña inversión que requiere el crear un sitio web. Una navegación engorrosa, puede por tanto hacernos perder de forma inmediata a casi todos nuestros posibles clientes-usuarios. Si a esto añadimos que la construcción de un sitio web se realiza con lenguajes que posibilitan una enorme flexibilidad, podemos ver la gran importancia que tiene el especificar algunos principios de diseño para el desarrollo de nuestro sitio web.

El diseño de sitios web debe seguir los siguientes principios (Tognazzini, 1999):

1. **Anticipación**, el sitio web debe anticiparse a las necesidades del usuario.

2. **Autonomía**, los usuarios deben tener el control sobre el sitio web. Los usuarios sienten que controlan un sitio web si conocen su situación en un entorno abarcable y no infinito.

3. Los **colores** han de utilizarse con precaución para no dificultar el acceso a los usuarios con problemas de distinción de colores (aprox. un 15% del total).

4. **Consistencia**, las aplicaciones deben ser consistentes con las expectativas de los usuarios, es decir, con su aprendizaje previo.

5. **Eficiencia del usuario**, los sitios web se deben centrar en la productividad del usuario, no en la del propio sitio web. Por ejemplo, en ocasiones tareas con mayor número de pasos son más rápidas de realizar para una persona que otras tareas con menos pasos, pero más complejas.

6. **Reversibilidad**, un sitio web ha de permitir deshacer las acciones realizadas

7. **Ley de Fitts** indica que el tiempo para alcanzar un objetivo con el ratón esta en función de la distancia y el tamaño del objetivo. A menor distancia y mayor tamaño más facilidad para usar un mecanismo de interacción.

8. **Reducción del tiempo de latencia**. Hace posible optimizar el tiempo de espera del usuario, permitiendo la realización de otras tareas mientras se completa la previa e informando al usuario del tiempo pendiente para la finalización de la tarea.

9. **Aprendizaje**, los sitios web deben requerir un mínimo proceso de aprendizaje y deben poder ser utilizados desde el primer momento.

10. El **uso adecuado de metáforas** facilita el aprendizaje de un sitio web, pero un uso inadecuado de éstas puede dificultar enormemente el aprendizaje.

11. La **protección del trabajo** de los usuarios es prioritario, se debe asegurar que los usuarios nunca pierdan su trabajo como consecuencia de un error.

12. **Legibilidad**, el color de los textos debe contrastar con el del fondo, y el tamaño de fuente debe ser suficientemente grande.

13. **Seguimiento de las acciones del usuario**. Conociendo y almacenando información sobre su comportamiento previo se ha de permitir al usuario realizar operaciones frecuentes de manera más rápida.

14. **Interfaz visible**. Se deben evitar elementos invisibles de navegación que han de ser inferidos por los usuarios, menús desplegables, indicaciones ocultas, etc.

Otros principios para el diseño de sitios web son (Nielsen):

a) Los usuarios deben ser capaces de alcanzar sus objetivos con un mínimo esfuerzo y unos resultados máximos.

b) Un sitio web no ha de tratar al usuario de manera hostil. Cuando el usuario comete un error el sistema ha de solucionar el problema, o en su defecto sugerir varias soluciones posibles, pero no emitir respuestas que meramente informen del error culpando al usuario.

c) En ningún caso un sitio web puede venirse abajo o producir un resultado inesperado. Por ejemplo no deben existir enlaces rotos.

d) Un sitio web debe ajustarse a los usuarios. La libertad en el uso de un sitio web es un término peligroso, cuanto mayor sea el número de acciones que un usuario pueda realizar, mayor es la probabilidad que cometa un error. Limitando el número de acciones al público objetivo se facilita el uso de un sitio web.

e) Los usuarios no deben sufrir sobrecarga de información. Cuando un usuario visita un sitio web y no sabe donde comenzar a leer, existe sobrecarga de información.

f) Un sitio web debe ser consistente en todos los pasos del proceso. Aunque pueda parecer apropiado que diferentes áreas tengan diseños diferentes, la consistencia entre los diseños facilita al usuario el uso de un sitio.

g) Un sitio web debe proveer de un feedback a los usuarios, de manera que éstos siempre conozcan y comprendan lo que sucede en todos los pasos del proceso.

2.19 Navegación en un sitio web

- Un punto importante dentro de un sitio web es la página principal, en ella se debe indicar de qué se trata el sitio. En esta página se debe incluir un texto lo más breve y claro posible explicando al visitante qué puede encontrar en el sito web, debe ser meramente informativa y debe impulsar a la acción al visitante. La página inicial es el lugar donde el visitante decide que hará, si entrar al contenido de ésta o abandonar el sitio.

- Todas las páginas que conforman el sitio deben tener un área de navegación con enlaces que permitan moverse dentro del mismo, esto con el principal objetivo de que el usuario no se sienta en ningún momento perdido dentro del sito web. El uso de una misma plantilla no quiere decir que la página sea aburrida, por el contrario, utilizar la misma plantilla mantiene una página consistente, limpia y ordenada.

- Las áreas de navegación sólo deben contener un número reducido de links, es decir, sólo los más útiles para los usuarios, con un máximo de 6 ó 7.

- Se debe aplanar la estructura de la página, es decir, la información de un sitio web no debe estar excesivamente jerarquizada. La obligatoriedad de navegar a través de muchas páginas antes de llegar al objetivo provocará la pérdida de muchos usuarios.

2.20 Estructura de contenidos en un sitio web

- Una página web no debe contener mucho texto, no más de unas pocas líneas. Los usuarios no leen, ojean velozmente en busca de la información que les interesa. “Las páginas deben ser ojeables”.

- Todo elemento de información presentado compite con el resto para captar la atención del usuario y por ello es crucial evitar presentar información superflua.

- La estructura de la información del texto debe tener las siguientes características:

- Texto estructurado: palabras resaltadas en negrita/color, listas numeradas…
- Contenidos estructurados con sumarios y tablas de contenidos.
- Títulos y subtítulos claros, simples y concisos.
- Párrafos cuyo contenido tenga una sola idea.
- Redacción en estilo de pirámide invertida, comenzando los textos por la conclusión y finalizando por los detalles.
- Usar la mitad de palabras que usaría en la redacción de un texto.
- Lenguaje objetivo (sin adjetivos, palabras redundantes o afirmaciones fortuitas…).
- Buena combinación de colores de texto y fondo: texto claro sobre fondo oscuro o viceversa (evitando la fatiga ocular).
- No usar textos parpadeantes o deslizantes, dificultan la lectura e imposibilitan a prestar atención a otro punto de la página.
- El lenguaje simple e informal es preferido al académico (la lectura es más rápida).

Capítulo 3. Desarrollo

3.1 Levantamiento de requerimientos

Cliente: Coordinación de Seminarios y Servicio Social de la División de Ingeniería Eléctrica.

Durante la entrevista se obtuvieron los requisitos que deberá tener el portal, quedando sujeto a cambios y refinamiento posteriores. Entendiendo como requisito una condición o capacidad que debe cumplir el sistema.

De acuerdo a las necesidades del cliente, el portal está delimitado de la siguiente forma:

1. El portal deberá ser capaz de realizar acciones que permitan mejorar tanto en rapidez como en eficiencia las acciones que el personal encargado de la Coordinación realiza manualmente, y por consiguiente sustituirlas.

2. El portal sólo se enfocará al área de temas de tesis y/o seminarios, ya que ésta es la que presenta mayor movimiento y consumo de tiempo por parte del personal encargado de realizar estas actividades.

3. El portal deberá ser totalmente confiable y seguro ya que los registros que se manejan en la Coordinación son muy importantes.

4. El portal deberá presentar una interfaz muy amigable para todos los usuarios.

5. Se deberán optimizar recursos.

6. El portal deberá ser capaz de utilizarse por personal que no tenga conocimientos sólidos en computación.

7. En caso de utilizar el sistema que se maneja diariamente en la Coordinación, se hizo hincapié en que éste no debe tener acceso a Internet por ningún motivo, además de que debe ser usado lo menos posible, todo esto para evitar cualquier infección de virus o ataques vía Internet.

8. El cliente señala que el portal que desarrollemos deberá ser capaz de almacenar los registros de una manera más robusta y segura pero sin interferir con la velocidad de operación del mismo.

9. El cliente podrá actualizar los datos cada vez que lo requiera.

10. Que el administrador y usuarios puedan imprimir los resultados que se mostraron durante la consulta.

3.2 Propuesta del sistema

Considerando los requerimientos, proponemos como base del desarrollo de nuestro sistema la siguiente arquitectura Cliente/Servidor, con los siguientes bloques (Figura 3.1):

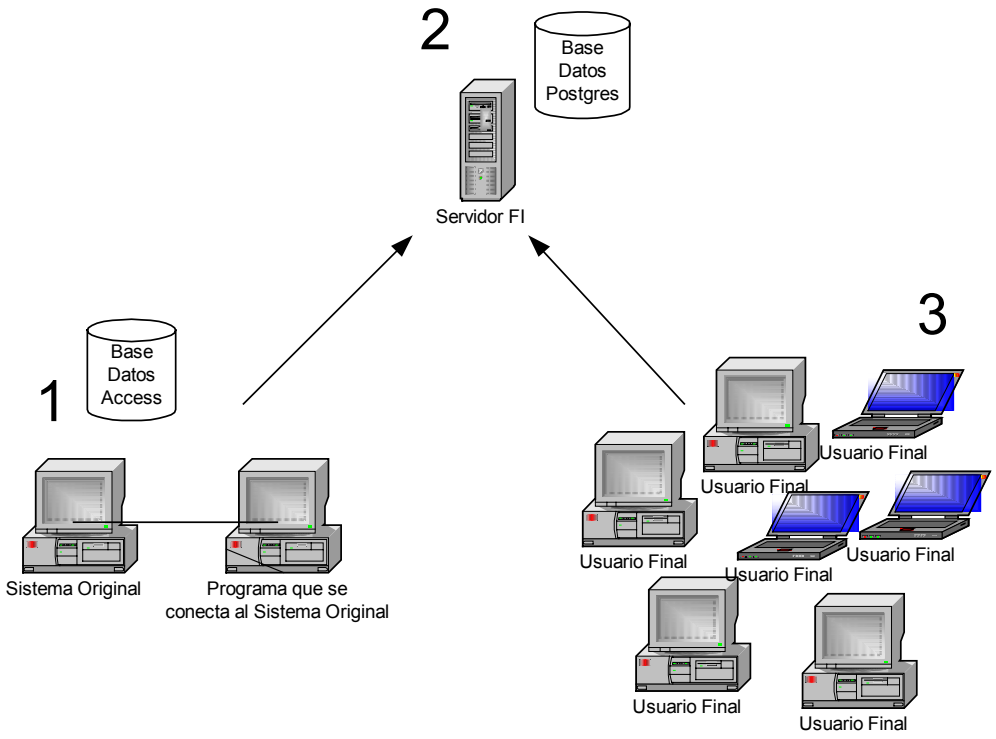


Figura 3.1 Arquitectura Cliente/Servidor del sistema

A continuación se describen las funciones que tendría cada uno de los bloques:

1. Sistema Coordinación de Seminarios y Servicio Social de la DIE.

En esta parte se encontrarán dos computadoras conectadas en red, una de ellas es la que aloja al sistema que administra todos los procesos referentes a los seminarios y servicio social de la DIE, la otra será la computadora destinada a llevar a cabo el proceso de actualización de la página web (consulta, transferencia de archivos al servidor, y actualización de la base de datos del servidor).

2. Servidor de la División de Ingeniería Eléctrica.

En esta parte se encontrarán alojados todos los archivos que conformarán el sitio web en el que se publicarán los temas aprobados para seminarios y/o tesis de la DIE (scripts de php, hojas de estilo, imágenes), así como los archivos que nos ayudarán a mantenerlo actualizado (archivos xml que contendrán actualizaciones, scripts de php que descargarán los archivos xml en la base de datos del servidor).

3. Usuarios finales.

En este último bloque se encontrarán todas aquellas personas (alumnos, profesores, directores, etc.) que deseen a través de la página de la Coordinación de Seminarios de la DIE consultar el banco de información de temas y directores para seminarios y/o tesis (tanto vigentes como históricos).

3.3 Arquitectura Cliente/Servidor

Toda aplicación de software tiene tres funciones fundamentales: administración de los datos, lógica de la aplicación (procesos) y lógica de la presentación (interfaz de usuario). Así, los procesos se efectúan mediante el uso de los dispositivos que forman parte del hardware; a su vez los datos y programas que constituyen parte del software interactúan entre sí realizando las funciones lógicas necesarias para correr una aplicación, misma que genera un despliegue de información (presentación) para el usuario.

La relación entre las funciones de una aplicación y la arquitectura Cliente/Servidor es tal, que los procesos, datos y presentación se ejecutan compartiendo recursos del sistema en red. Esta relación está presente tanto en las PC's como en los sistemas grandes; y pretende lograr la integración de ambas plataformas, combinando lo mejor de cada una dentro de un mismo sistema. Es por ello que ponemos especial énfasis en la arquitectura Cliente/Servidor, tomando a ésta como base fundamental en nuestro sistema.

La arquitectura Cliente/Servidor es la plataforma abierta por excelencia, por la variedad de combinaciones de clientes y servidores que permite conectar en red. Sin embargo, elegir las plataformas, las herramientas, los proveedores y las bases de administración de la arquitectura Cliente/Servidor, además de la tecnología de creación, es una decisión difícil de tomar.

La arquitectura Cliente/Servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos (Figura 3.2).

Figura 3.2 Arquitectura Cliente/Servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, inicia un proceso de diálogo: produce una demanda de información o solicita recursos. La computadora que responde a la demanda del cliente, se conoce como servidor. Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es Internet. Cliente/Servidor es el modelo de interacción más común entre aplicaciones en una red. No forma parte de los conceptos de Internet como los protocolos IP, TCP o UDP, sin embargo todos los servicios estándares de alto nivel propuestos en Internet funcionan según este modelo.

Se puede decir que la arquitectura Cliente/Servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura. No existe una definición específica adoptada universalmente de la arquitectura Cliente/Servidor, las empresas de cómputo enfocan el concepto basándose en la funcionalidad que representa según los servicios que ellas mismas ofrecen.

Características

Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Todos los sistemas desarrollados en arquitectura Cliente/Servidor poseen las siguientes características distintivas de otras formas de software distribuido:

- **Servicio**. El servidor es un proveedor de servicios; el cliente es un consumidor de servicios.

- **Recursos compartidos**. Un servidor puede atender a muchos clientes al mismo tiempo y regular su acceso a recursos compartidos.

- **Protocolos Asimétricos**. La relación entre cliente y servidor es de muchos a uno; los clientes solicitan servicios, mientras los servidores esperan las solicitudes pasivamente.

- **Transparencia de ubicación**. El software Cliente/Servidor siempre oculta a los clientes la ubicación del servidor.

- **Mezcla e igualdad**. El software es independiente del hardware o de las plataformas de software del sistema operativo; se puede tener las mismas o diferentes plataformas de cliente y servidor.

- **Intercambio basados en mensajes**. Los sistemas interactúan a través de un mecanismo de transmisión de mensajes: la entrega de solicitudes y respuestas del servicio.

- **Encapsulamiento de servicios**. Los servidores pueden ser sustituidos sin afectar a los clientes, siempre y cuando la interfaz para recibir peticiones y ofrecer servicios no cambie.

- **Facilidad de escalabilidad**. Los sistemas Cliente/Servidor pueden escalarse horizontal o verticalmente. Es decir, se pueden adicionar o eliminar clientes (con apenas un ligero impacto en el desempeño del sistema); o bien, se puede cambiar a un servidor más grande o a servidores múltiples.

- **Integridad**. El código y los datos del servidor se conservan centralmente; esto implica menor costo de mantenimiento y protección de la integridad de los datos compartidos. Además, los clientes mantienen su individualidad e independencia.

La arquitectura Cliente/Servidor es una infraestructura versátil modular y basada en mensajes que pretende mejorar la portabilidad, la interoperabilidad y la

escalabilidad del cómputo; además es una apertura del ramo que invita a participar a una variedad de plataformas, hardware y software del sistema.

Componentes

Conceptualmente, los componentes de la arquitectura Cliente/Servidor son el cliente, el servidor y la infraestructura de comunicaciones (middleware), como a continuación se explica (Figura 3.3).

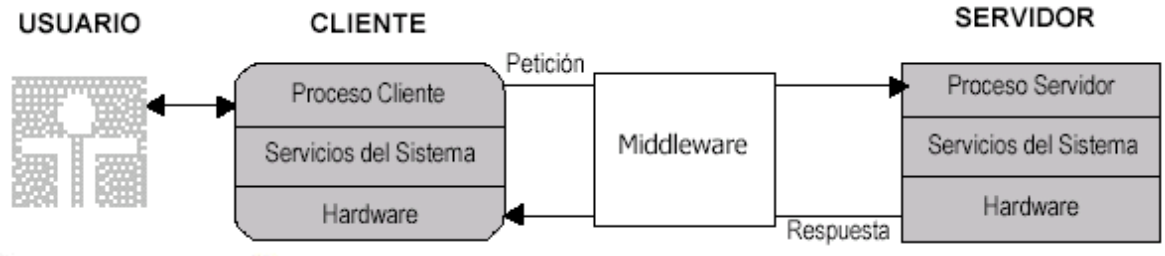


Figura 3.3 Componentes de la arquitectura Cliente/Servidor

Cliente

El cliente es la entidad por medio de la cual un usuario solicita un servicio, realiza una petición o demanda el uso de recursos. Este elemento se encarga, básicamente, de la presentación de los datos y/o información al usuario en un ambiente gráfico.

Se comunica con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad; además, requiere el uso de los recursos de la computadora para cualquier actividad y puede interactuar con uno o varios servidores.

Los clientes se suelen situar en PC's o en estaciones de trabajo, se encargan de realizar el FRONT END, que es la parte de la aplicación que interactúa con el usuario, en ellos permanecen las aplicaciones particulares de cada usuario, y realizan funciones como:

- Manejo de la interfaz del usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Como ejemplos de clientes pueden citarse interfaces de usuario para enviar comandos a un servidor, APIs para el desarrollo de aplicaciones distribuidas, herramientas en el cliente para hacer acceso a servidores remotos (por ejemplo, servidores de SQL) o aplicaciones que solicitan acceso a servidores para algunos servicios.

Servidor

El servidor es la entidad física que provee un servicio y devuelve resultados; ejecuta el procesamiento de datos, aplicaciones y manejo de la información o recursos. En el servidor se realiza el BACK END que es la parte destinada a recibir las solicitudes del cliente y dónde se ejecutan los procesos.

En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PC's poderosas, estaciones de trabajo, minicomputadoras o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación, y contabilidad.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.
- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en computadoras personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

En el servidor permanecen las aplicaciones que deben ser compartidas por varios usuarios. Normalmente, aunque con excepciones, el cliente y el servidor están ubicados en distintos procesadores; incluso, un servidor puede fungir como cliente de otros servidores.

Existen diversos servidores mismos que se clasifican basándose en su funcionalidad; éstos son denominados servidores dedicados ya que administran el uso de algún recurso en particular, por ejemplo: Servidor de archivos, Servidor de bases de datos, Servidor de transacciones, Servidor de groupware, Servidor de objetos, Servidor de web, Servidor de impresoras, Servidor de aplicaciones, Servidor de respaldos.

El proceso del servidor es reactivo, es decir, realiza una acción basándose en una instrucción previa; simplificando, realiza una función posterior a una petición o a la ejecución de una transacción requerida por el cliente, o bien por otro servidor.

Middleware

Para que los clientes y servidores puedan comunicarse se requiere de una infraestructura lógica que proporcione los mecanismos básicos de direccionamiento y transporte. A dicha infraestructura se le denomina *middleware*, el cual es un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores.

El middleware es un módulo intermedio que no pertenece a los dominios del servidor, ni a la interfaz de usuario, ni a la lógica de la aplicación en los dominios del cliente; tampoco debe confundirse con la red física en sí (cableado, señales de radio o infrarrojas), el middleware es una interfaz lógica estándar de los servicios de red. Sus funciones son:

- Independizar las dos entidades: El cliente y el servidor no necesitan saber comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.

- Traducir la información de una aplicación y pasarla a la otra: acepta consultas y datos recuperándolos de la aplicación cliente, los transmite y envía la respuesta de regreso. También genera los códigos de error.

- Controlar las comunicaciones: da a la red las características adecuadas de desempeño, confiabilidad, transparencia y administración.

Existen dos tipos de middleware:

1. El middleware general es el sustrato de la mayoría de las interacciones de Cliente/Servidor. Incluye las pilas de comunicación, directorios distribuidos, servicios de autentificación, horario de la red, llamadas a procedimientos remotos (RPC's), y servicios en cola. Incluye también las extensiones del sistema operativo de redes, como los servicios distribuidos de archivos e impresión y los productos de middleware orientado a mensajes (MOM: message oriented middleware).

2. El middleware de servicios específicos es necesario para cumplir un tipo particular de servicio de Cliente/Servidor; así, existe un middleware específico para los servidores dedicados: middleware para bases de datos, middleware para OLTP, middleware para groupware; middleware para objetos, etc.

El middleware es una herramienta adecuada, que no sólo es flexible y segura, sino que también protege la inversión en tecnología y permite manejar diferentes ambientes de computación.

3.4 Diseño (UML)

El UML (Lenguaje Unificado de Modelado) nació en 1994 cubriendo los aspectos principales de todos los métodos de diseño antecesores, los padres de UML son Grady Booch, James Rumbaugh e Ivar Jacobson. La versión 1.0 de UML fue liberada en Enero de 1997 y ha sido utilizado con éxito en sistemas construidos para toda clase de industrias alrededor del mundo: hospitales, bancos, comunicaciones, aeronáutica, finanzas, etc.

Por sus siglas en inglés, Unified Modeling Language es el lenguaje de modelado de sistemas de software más conocido en la actualidad y prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas de software principalmente, también describe la semántica esencial de lo que estos diagramas y símbolos significan.

UML tiene nueve tipos de diagramas que son utilizados en combinación para proveer todas las vistas de un sistema de los cuáles los principales son: diagramas de caso de uso, de clases, de secuencia y de actividad.

Los principales beneficios de utilizar UML para desarrollar sistemas de cómputo son:

- Mejores tiempos totales de desarrollo (de 50% o más).
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

A continuación se muestra el diagrama de los requerimientos del cliente vistos de manera general (Figura 3.4).

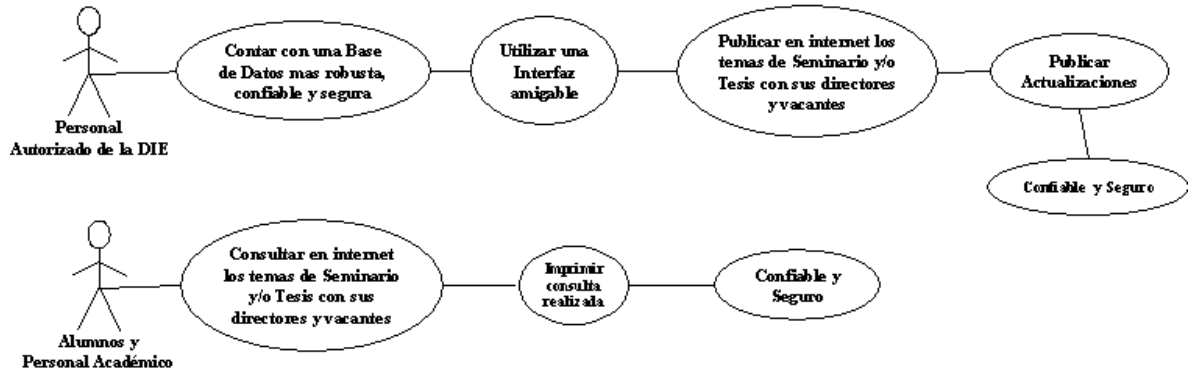


Figura 3.4 Diagrama de requerimientos del cliente

Identificación de los actores que intervienen en el sistema.

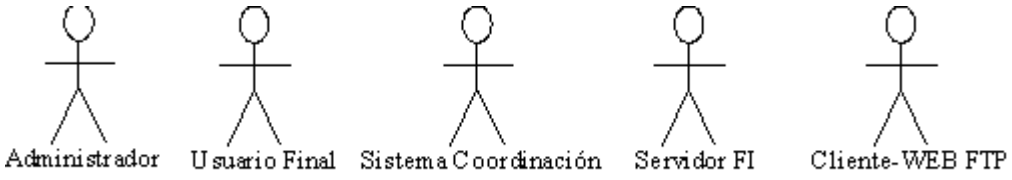


Figura 3.5 Actores que intervienen en el sistema

Descripción de los actores:

Administrador. Es el único usuario autorizado para actualizar y publicar registros en el sistema de la Coordinación y por consiguiente en el portal.

Usuario final. Se refiere a todos los usuarios que podrán consultar el portal (alumnos, profesores, personal académico, etc.)

Sistema Coordinación. Se refiere al sistema de la Coordinación encargado de administrar los procesos de seminarios y servicio social.

Servidor FI. Es el servidor de la Facultad de Ingeniería (donde estará alojado el portal).

Cliente-WEB FTP. Se refiere a la herramienta que se usará para transferir los archivos al Servidor FI.

3.4.1 Casos de uso

Es la técnica más efectiva y a la vez más simple para modelar los requisitos del sistema desde la perspectiva del usuario.

Los casos de uso se utilizan para modelar cómo un sistema o negocio funciona o bien cómo se desea que funcione y son generalmente el punto de partida del análisis con UML.

Este modelo consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema (muñecos). Los casos de uso representan el comportamiento del sistema (elipses).

Caso de uso general del portal

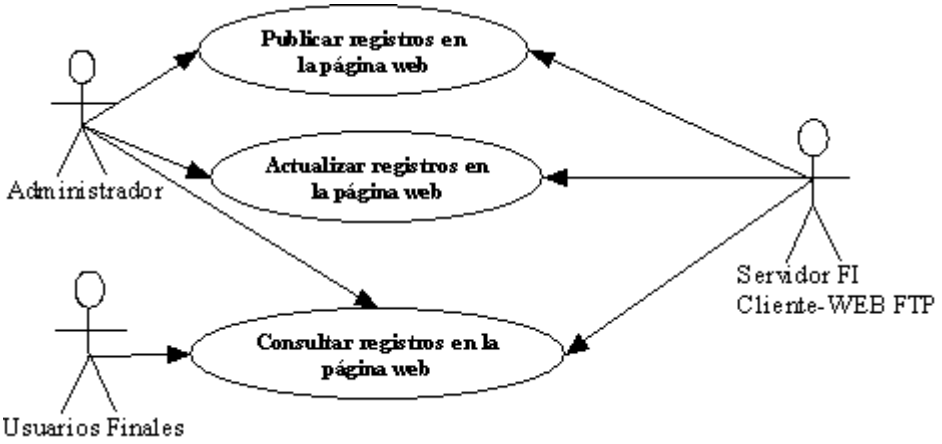


Figura 3.6 Diagrama de caso de uso general

Caso de uso:	**Publicar registros en la página web**
Actores:	Administrador, Servidor FI, Cliente-WEB FTP.
Propósito:	Que el administrador publique los alumnos, profesores y temas aprobados para seminarios y/o tesis.
Resumen:	En este caso se publicarán los temas aprobados para seminarios y/o tesis.
Precondiciones:	No debe haber registros de alumnos, profesores y temas vacíos.

Caso de uso:	**Actualizar registros en la página web**
Actores:	Administrador, Servidor FI, Cliente-WEB FTP.
Propósito:	Que el administrador actualice los alumnos, profesores y temas aprobados para seminarios y/o tesis.
Resumen:	En este caso se realiza la consulta local, transferencia de archivos al servidor y actualización de la base de datos del servidor.
Precondiciones:	El administrador debió haber registrado nuevos alumnos,

	profesores y/o temas.

Caso de uso:	**Consultar registros en la página web**
Actores:	Administrador, Usuario final, Servidor FI.
Propósito:	Que el administrador y/o usuario final consulte(n) los profesores y temas aprobados para seminarios y/o tesis.
Resumen:	El administrador y/o usuario final consultará(n) la página del portal donde encontrará(n) varias opciones para realizar su búsqueda.

Caso de uso principal

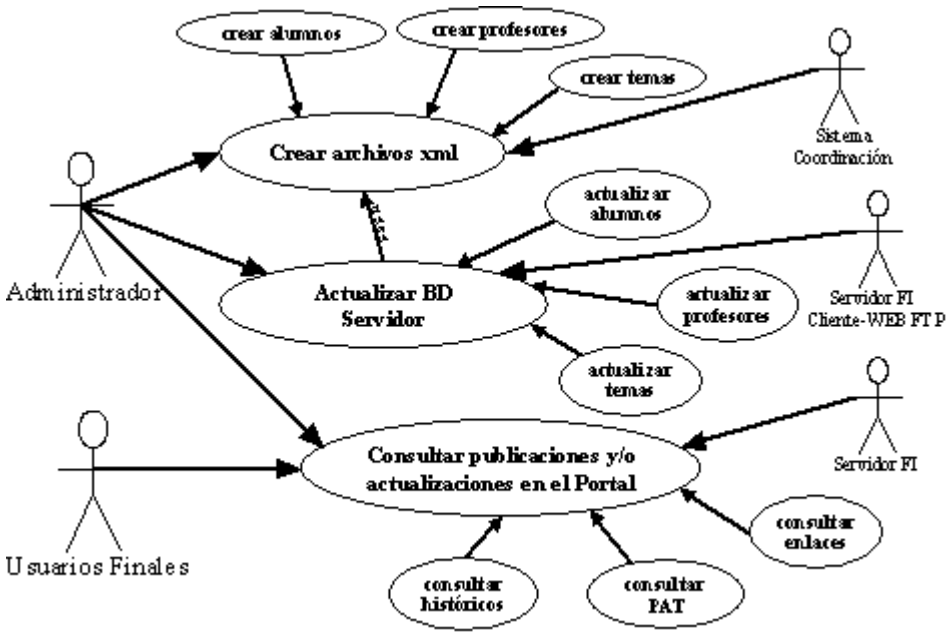


Figura 3.7 Diagrama de caso de uso principal

Caso de uso:	**Crear archivos xml**
Actores:	Administrador, Sistema Coordinación.
Propósito:	Que el administrador cree 3 archivos: alumnos.xml, profesores.xml y temas.xml consultados del sistema de la Coordinación.
Resumen:	El administrador accederá al programa llamado SCSSSDIE.exe donde podrá elegir que archivos son los que desea crear. El administrador deberá crear los 3 archivos para la correcta actualización del portal o bien salir del programa sin realizar ninguna modificación.
Precondiciones:	Se requiere que el administrador haya dado de alta algún alumno, profesor o tema.
Flujo principal:	En la interfaz llamada SCSSSDIE.exe el administrador creará los archivos consultando los registros que se encuentren dados de alta en el sistema de la Coordinación. Aceptar si se han creado

	correctamente.
Subflujos:	Crear alumnos.xml Si se creó el archivo correctamente dar clic en aceptar. Crear profesores.xml Si se creó el archivo correctamente dar clic en aceptar. Crear temas.xml Si se creó el archivo correctamente dar clic en aceptar.
Excepción:	No se creó algún archivo correctamente. Se manda un mensaje de error. Verificar que los datos se hayan capturado correctamente en el sistema de la Coordinación. Intentar de nuevo.

Caso de uso:	**Actualizar la base de datos del servidor (Seminarios)**
Actores:	Administrador, Servidor FI.
Propósito:	Actualizar la base de datos del servidor con los archivos transferidos en el caso de uso anterior.
Resumen:	El administrador podrá elegir que archivos son los que desea actualizar: alumnos, profesores y/o temas. El administrador deberá elegir las 3 opciones para la correcta actualización del portal.
Precondiciones:	Se requiere haber transferido exitosamente los archivos de actualización al servidor.
Flujo principal:	El administrador accederá a la pantalla de actualización. Al seleccionar la opción deseada se ejecutará el archivo de inserción correspondiente.
Subflujos:	Elegir alumnos. Se ejecuta el script inserta.php. Si se actualizó alumnos correctamente aparecerá una pantalla de confirmación. Elegir profesores. Se ejecuta el script insertp.php Si se actualizó profesores correctamente aparecerá una pantalla de confirmación. Elegir temas. Se ejecuta el script insertt.php Si se actualizó temas correctamente aparecerá una pantalla de confirmación.
Excepción:	No se actualizó algún archivo. Se manda un mensaje de error. Cerrar página e intentarlo de nuevo.

Caso de uso:	**Consultar históricos**
Actores:	Administrador, Usuario final, Servidor FI.
Propósito:	Consultar en Internet los directores y/o codirectores con sus respectivos temas y vacantes que se encuentran vigentes o de semestres anteriores aprobados por la Coordinación.
Precondiciones:	Se requiere que el administrador haya realizado el caso de uso “Actualizar la base de datos del servidor”.
Resumen:	El administrador y/o usuario final podrá(n) elegir diversas

	opciones dentro del buscador para realizar su búsqueda más especializada.
Flujo principal:	El administrador y/o usuario final podrá(n) realizar su búsqueda por semestre, carrera, nombre del director y/o codirector de tema de seminario y/o tesis de registros que se encuentran desde el primer semestre del año 1995 hasta el semestre actual.
Excepción:	La búsqueda no regresa ningún resultado. Se sugiere revisar la ortografía o realizar búsquedas más generales, o con otras palabras. Intentar de nuevo.

Caso de uso:	**Consultar PAT**
Actores:	Administrador, Usuario final, Servidor FI.
Propósito:	Consultar en Internet los temas pertenecientes al PAT aprobados por la Coordinación.
Precondiciones:	Se requiere que el administrador haya realizado el caso de uso “Actualizar la base de datos del servidor”.
Resumen:	El administrador y/o usuario final podrá(n) consultar los temas que pertenezcan al PAT de los 2 últimos semestres, así como consultar información sobre este programa.
Flujo principal:	El administrador y/o usuario final consultará la página en Internet de la Coordinación de Seminarios de la DIE en la cual podrá elegir la opción “Programa de Apoyo a la Titulación (PAT)”.
Excepción:	La búsqueda no regresa ningún resultado. No se han aprobado temas pertenecientes al PAT en los 2 últimos semestres. Seguir consultando actualizaciones.

Caso de uso:	**Consultar enlaces**
Actores:	Administrador, Usuario final, Servidor FI.
Propósito:	Realizar las consultas a diferentes páginas de interés de la UNAM y que tienen estrecha relación con la Facultad de Ingeniería.
Resumen:	El administrador y/o usuario final podrá(n) consultar las páginas de la UNAM, página de la FI y la página de servicio social.
Flujo principal:	El administrador y/o usuario final podrá(n) elegir la página a consultar dando clic en la imagen o animación correspondiente.

Dado que el caso de uso siguiente no forma parte de nuestro portal puesto que no lo implementamos, sólo hacemos uso de él, decidimos sólo citarlo pues consideramos que es importante para comprender mejor el funcionamiento del portal.

Caso de uso:	**Transferir archivos al servidor**
Actores:	Administrador, Cliente-WEB FTP.
Propósito:	Transferir los archivos creados en el caso de uso “Crear archivos xml” al servidor para publicarlos en la página del portal.
Resumen:	El administrador subirá los 3 archivos “xml” creados en el caso de uso “Crear archivos xml”.

Precondiciones:	Se requiere haber ejecutado el caso de uso “Crear archivos xml”.
Flujo principal:	Elegir “carpeta WEB”. Elegir “subir archivo”. Dar clic en examinar. Buscar la carpeta ArchivosXML ubicada en el escritorio de Windows y seleccionar los archivos xml creados en el caso de uso “Crear archivos xml” Dar clic en abrir. Repetir los 2 pasos anteriores para subir los 3 archivos xml. Dar clic en la paloma (subir archivos). Aparecerá pantalla de resultados. Regresar a pantalla principal. Cerrar sesión dando clic en el botón rojo (superior derecha). Cerrar buscador de Windows.
Subflujos:	Subir alumnos.xml Subir profesores.xml Subir temas.xml
Excepción:	La pantalla de resultados muestra un mensaje señalando que no se subieron correctamente los archivos al servidor. Cerrar sesión y cerrar buscador de Windows e intentar de nuevo.

3.4.2 Diagrama de actividades

Un diagrama de actividades es provechoso para entender el comportamiento general de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto.

Un estado de actividad se representa como una caja con los extremos redondeados que contiene una descripción de actividad. Las transacciones simples de terminación se muestran como flechas. Las ramas se muestran como condiciones de guarda en transiciones o como diamantes con múltiples flechas de salida etiquetadas. Una división o una unión de control se representa con múltiples flechas que entran o salen de la barra gruesa de sincronización (Figura 3.8).

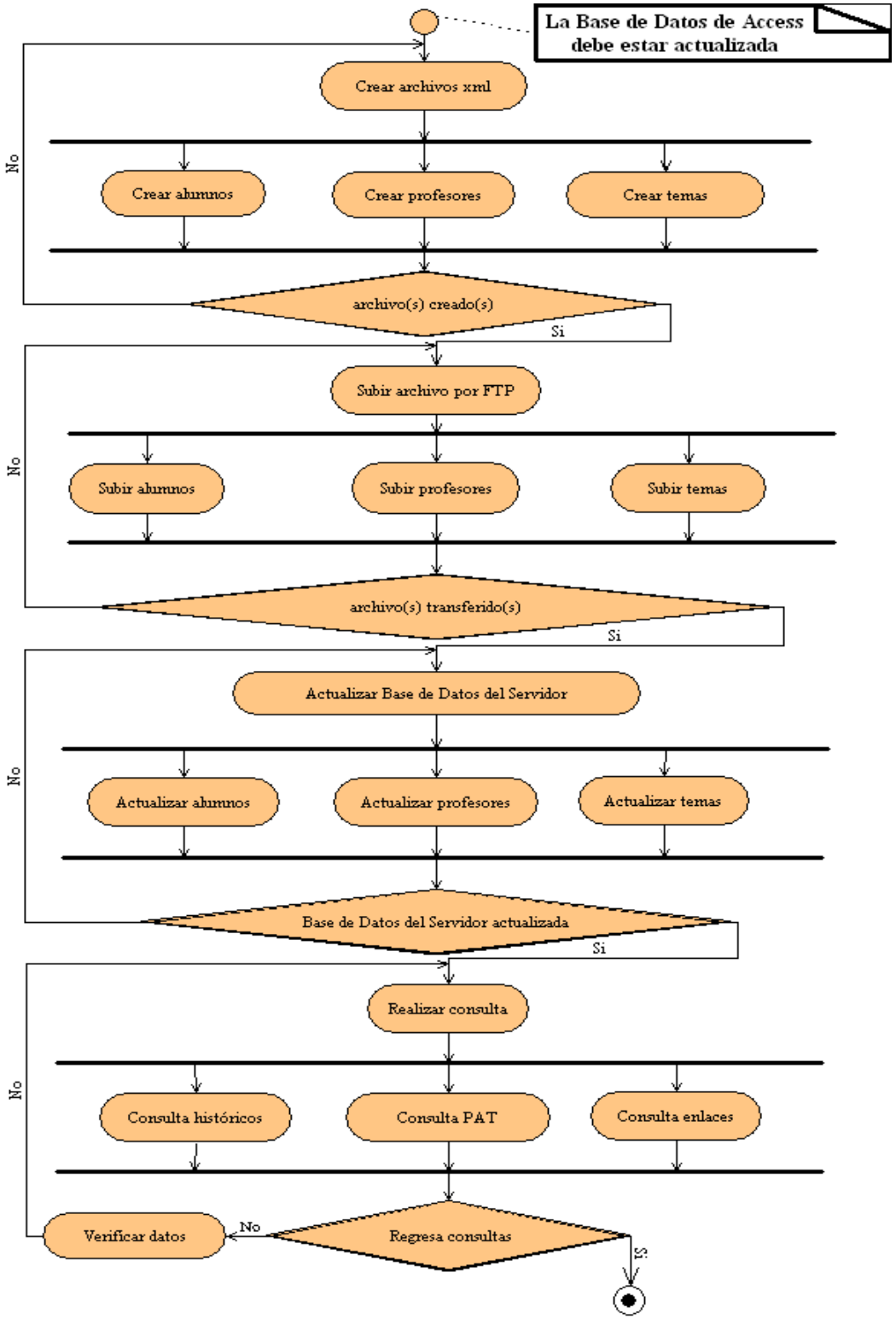


Figura 3.8 Diagrama de actividades

3.4.3 Diagramas de secuencia

Este diagrama muestra la interacción de los objetos entre ellos. El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema a través del tiempo. Este diagrama se modela para cada caso de uso.

Este diagrama muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

En los siguientes diagramas de secuencia sólo se mostrarán los casos para crear y actualizar los alumnos en la base de datos del servidor, puesto que para simplificar los diagramas, decidimos no mostrar los otros 2 casos (profesores y temas) ya que estos diagramas son similares para cada uno de los tres archivos xml: alumnos.xml, profesores.xml y temas.xml.

Diagrama de secuencia para crear los archivos xml de actualización

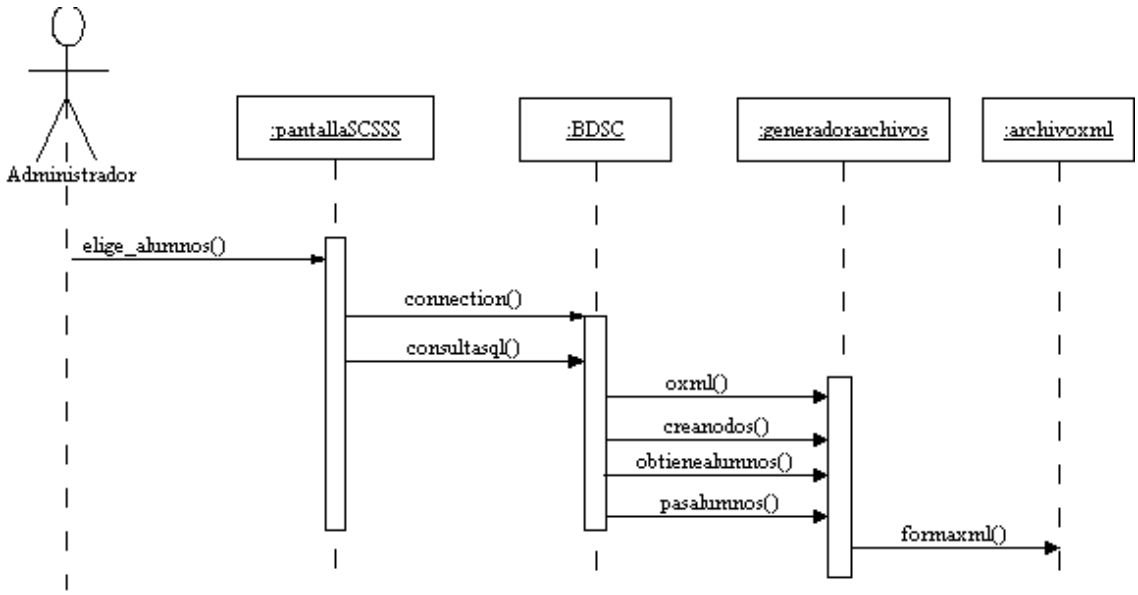


Figura 3.9 Diagrama de secuencia de actualización de archivos

Diagrama de secuencia para actualizar la base de datos del servidor (Seminarios)

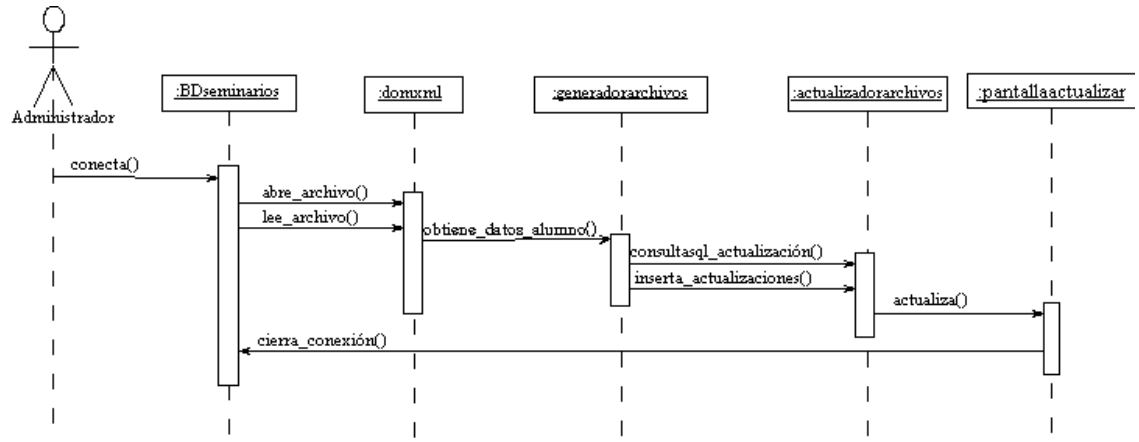


Figura 3.10 Diagrama de secuencia de actualización de la base de datos

Diagrama de secuencia para consultar publicaciones en Internet

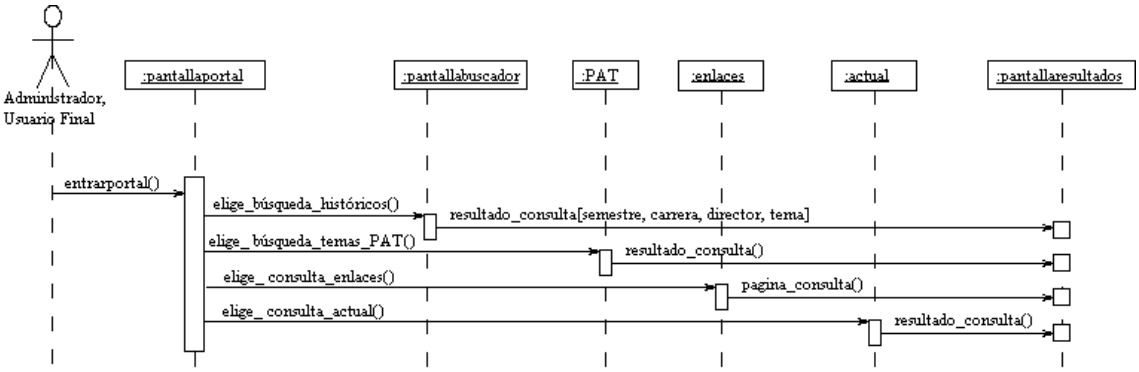


Figura 3.11 Diagrama de secuencia para consultar publicaciones

3.4.4 Diagrama de clases

El diagrama de clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño se usa el mismo diagrama y se modifica para satisfacer los detalles de las implementaciones. Gráficamente las clases se representan en una caja rectangular dividida en 3 compartimentos: en la parte superior se encuentra el nombre de la clase, en la parte media se encuentran los atributos y en la parte inferior se encuentran las operaciones o métodos de la clase. Las clases se relacionan entre sí a través de relaciones que son las líneas

rectas entre dos clases. Los roles son las frases que se encuentran en la relación (Figura 3.12).

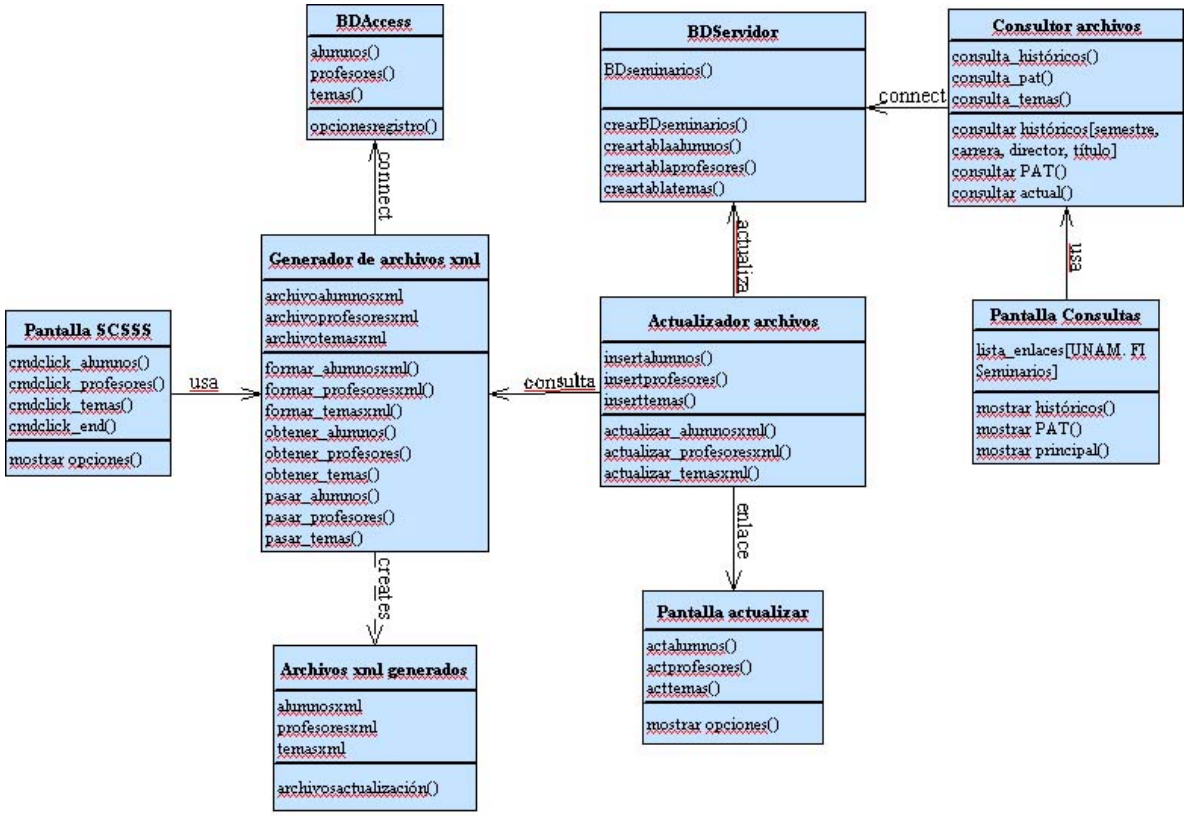


Figura 3.12 Diagrama de clases

3.5 Visual Basic

Visual Basic es un lenguaje de programación desarrollado por Microsoft. Visual Basic es un lenguaje visual que desciende del lenguaje de programación BASIC. Su primera versión fue presentada en 1991 con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Es un lenguaje simple pensado para programadores inexpertos, guiado por eventos, y centrado en un motor de formularios poderoso que facilita el rápido desarrollo de aplicaciones gráficas. Su sintaxis, derivada del antiguo BASIC, ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. No requiere de manejo de punteros. Posee varias bibliotecas para manejo de bases de datos, pudiendo conectar con cualquier base de

datos a través de ODBC (Informix, DBase, Acces, Mysql, SQL Server, etc) a través de ADO.

Es utilizado principalmente para aplicaciones de gestión de empresas, debido a la rapidez con la que puede hacerse un programa que utilice una base de datos sencilla, además de la abundancia de programadores en este lenguaje.

El compilador de Microsoft genera ejecutables que requieren una DLL para que sus ejecutables funcionen, en algunos casos llamada MSVBVMxy.DLL (acrónimo de "MicroSoft Visual Basic Virtual Machine x.y", siendo x.y la versión) y en otros VBRUNXXX.DLL ("Visual Basic Runtime X.XX"), que provee todas las funciones implementadas en el lenguaje. Además existen un gran número de bibliotecas (DLL) que facilitan el acceso a muchas funciones del sistema operativo y la integración con otras aplicaciones.

¿Por qué Visual Basic 6.0 Edición Profesional?

Decidimos utilizar Visual Basic 6.0 Edición Profesional para crear la interfaz de consulta al sistema de la Coordinación de Seminarios y Servicio Social de la DIE llamada SCSSS.exe (bloque 1 de la Figura 3.1) ya que esta versión de Visual Basic incluye una librería llamada ADO (Activex Data Objects) que nos permite acceder a la base de datos de dicho sistema mediante el uso de eventos en los recordsets con el propósito de movernos de una manera más sencilla por los registros.

Las características generales de ADO son:

- Fácil de utilizar.
- Alto rendimiento.
- Control de cursores mediante programación.
- Capacidad de devolver múltiples conjuntos de resultados desde una única consulta.

Para crear los archivos de actualización alumnos,xml, profesores.xml y temas.xml (los cuales nos permiten actualizar la base de datos del servidor), primero utilizamos las propiedades y métodos de ADO para crear y abrir la conexión con la base de datos de Access del sistema de la Coordinación y posteriormente mediante código SQL extraer los datos que nos interesan, una vez obtenidos los resultados de la consulta manipulamos los datos utilizando las propiedades de ADO en conjunto con las de XML y formamos los archivos xml de actualización, este proceso se puede observar en el diagrama de la Figura 3.9 de este capítulo.

3.6 XML

XML es el acrónimo de eXtensible Markup Language (Lenguaje de Marcado Ampliable o Extensible) desarrollado por el World Wide Web Consortium (W3C). Decidimos utilizar este lenguaje para la actualización de la base de datos del servidor, ya que al igual que HTML, XML se basa en documentos de texto plano que por su ligereza nos permite la transmisión por la red de una gran cantidad de datos organizados de tal manera que nos resulte fácil manipularlos y colocarlos en la base de datos.

En general, podemos resumir las principales características de XML en los siguientes puntos:

- En cuanto a la comunicación entre aplicaciones, XML permite representar los datos de una manera muy simple y fácil de transmitir por la red. En los últimos tiempos este uso se está haciendo muy popular con el surgimiento de los servicios web.

- XML utiliza una API llamada DOM que proporciona una representación de los documentos XML en forma de árbol, permitiendo el recorrido y manipulación de los datos.

- Destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones.

3.7 Base de datos

A continuación enunciamos varias definiciones de lo qué es una base de datos ya que consideramos que es uno de los conceptos más importantes dentro del desarrollo de nuestro sistema.

Una base de datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

Las bases de datos son un conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, posibilitando su administración, relación u ordenamiento en base a diferentes criterios.

Conjunto de datos organizados de modo tal que resulte fácil acceder a ellos, gestionarlos y actualizarlos.

Una base de datos es un conjunto ordenado de información almacenado de forma que se pueda acceder a la misma fácil y rápidamente.

3.7.1 Modelo Entidad/Relación

El Modelo Entidad/Relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. El Modelo Entidad/Relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas. Este modelo es el resultado del análisis que se realiza para cada sistema de información que se desea desarrollar (Figura 3.13).

Originalmente, el Modelo Entidad/Relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido.

Entidad

Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual.

Hay dos tipos de entidades: fuertes y débiles. Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil.

Relación (interrelación)

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Una relación se representa mediante una línea que une dos recuadros de entidades o recursivamente une un recuadro de entidad consigo misma.

Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc.

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

La cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial).

Atributo

Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos. Gráficamente, se representan mediante bolitas que cuelgan de las entidades o relaciones a las que pertenecen.

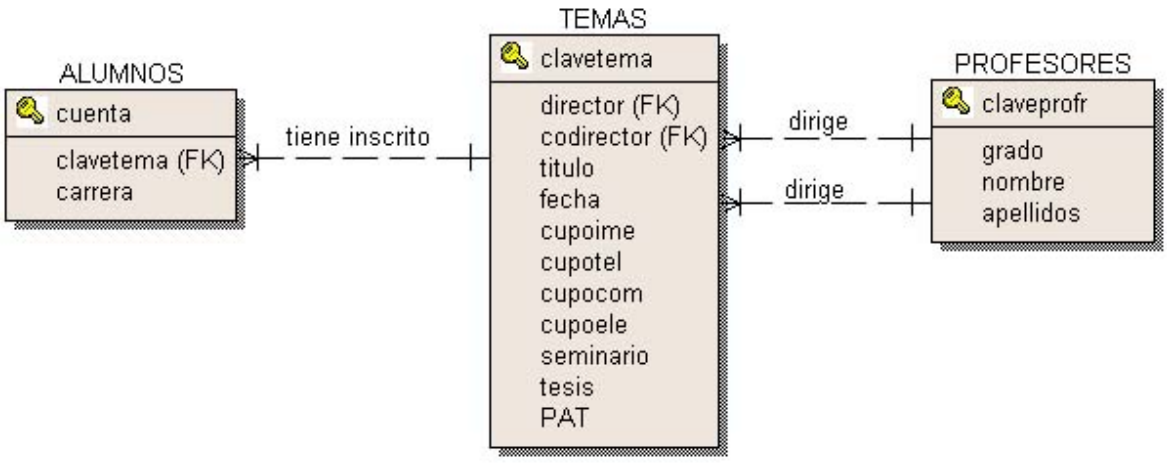


Figura 3.13 Modelo Entidad/Relación

Diccionario de datos

El propósito de un diccionario de datos es documentar la estructura interna de cada tabla, incluyendo todos sus campos y sus tipos de datos con comentarios.

TABLA ALUMNOS

Nombre:	cuenta
Tipo de dato:	char(12)
Obligatorio:	Sí
Descripción:	Número de cuenta del alumno. Llave primaria.

Nombre:	clavetema
Tipo de dato:	char(8)
Obligatorio:	Sí
Descripción:	Número que identifica a cada tema y con el que se da de alta el mismo, está formada de tres partes (año, semestre y grupo o número de tema). Llave foránea.

Nombre:	carrera
Tipo de dato:	integer
Obligatorio:	No
Descripción:	Clave de la carrera en la que está inscrito el alumno.

TABLA TEMAS

Nombre:	clavetema
Tipo de dato:	char(8)
Obligatorio:	Sí
Descripción:	Número que identifica a cada tema y con el que se da de alta el mismo, está formada de tres partes (año, semestre y grupo o número de tema). Llave primaria.

Nombre:	director
Tipo de dato:	integer
Obligatorio:	Sí
Descripción:	Clave del director del tema. Llave foránea.

Nombre	codirector
Tipo de dato:	integer
Obligatorio:	Sí
Descripción:	Clave del codirector del tema. Llave foránea.

Nombre:	titulo
Tipo de dato:	varchar(250)
Obligatorio:	No
Descripción:	Nombre con el que se identifica el tema a desarrollar.

Nombre	fecha
Tipo de dato:	char(10)
No	No
Descripción:	Fecha en la que se aprobó el tema.

Nombre	cupoime
Tipo de dato:	integer
Obligatorio:	No
Descripción:	Número de alumnos de la carrera de Ingeniería Mecánica

	Electricista que pueden estar inscritos en el tema.

Nombre	cupotel
Tipo de dato:	integer
Obligatorio:	No
Descripción:	Número de alumnos de la carrera de Ingeniería en Telecomunicaciones que pueden estar inscritos en el tema.

Nombre	cupocom
Tipo de dato:	integer
Obligatorio:	No
Descripción:	Número de alumnos de la carrera de Ingeniería en Computación que pueden estar inscritos en el tema.

Nombre	cupoele
Tipo de dato:	integer
Obligatorio:	No
Descripción:	Número de alumnos de la carrera de Ingeniería Eléctrica Electrónica que pueden estar inscritos en el tema.

Nombre	seminario
Tipo de dato:	varchar(2)
Obligatorio:	No
Descripción:	Indica si el tema puede ser tomado en cuenta para evaluar la asignatura de Seminario.

Nombre	tesis
Tipo de dato:	varchar(2)
Obligatorio:	No
Descripción:	Indica si el tema puede ser desarrollado como tema de Tesis.

Nombre	pat
Tipo de dato:	varchar(2)
Obligatorio:	No
Descripción:	Indica si el tema pertenece al Programa de Apoyo a la Titulación.

TABLA PROFESORES

Nombre:	claveprofr
Tipo de dato:	integer
Obligatorio:	Sí
Descripción:	Clave con la que se identifica al director o codirector. Llave primaria.

Nombre:	grado

Tipo de dato:	varchar(8)
Obligatorio:	No
Descripción:	Grado académico del director o codirector.

Nombre:	nombre
Tipo de dato:	varchar(40)
Obligatorio:	No
Descripción:	Nombre(s) del director o codirector.

Nombre:	apellidos
Tipo de dato:	varchar(60)
Obligatorio:	No
Descripción:	Apellido paterno y materno del director o codirector.

3.7.2 Modelo Relacional

El Modelo Relacional es la expresión matemática del Modelo Entidad/Relación, acercándose de este modo al diseño de la base de datos. El Modelo Relacional se compone de los siguientes elementos:

Relación

Estructura básica y única del Modelo Relacional, también se le llama tabla y sirve para representar tanto los objetos como las acciones entre ellos. Una relación está formada por un conjunto de atributos llamados columnas y un conjunto de filas llamadas tuplas. Una relación es lo que conocemos como tabla y se le llama relación por que contiene listas (o relaciones) de datos.

- Cada atributo de una relación se llama columna (o campo), tiene un nombre único dentro de esa relación (tabla) y puede guardar valores. El orden de los atributos en una tabla carece de importancia.

- El conjunto de atributos de una relación representa una tupla (fila o registro). El orden en que se almacenan las filas carece de importancia.

De las relaciones (tablas) se derivan los siguientes conceptos:

- La Cardinalidad de una tabla es su número de filas.

- El Grado de una tabla es el número de columnas.

- Un Valor en una tabla viene representado por la intersección entre una fila y una columna.

- Un Valor NULL (nulo) representa la ausencia de información y es distinto del cero y de los espacios en blanco.

Dominios

Es el conjunto de valores que puede tomar un atributo. Existen dos tipos:

- Generales: Los valores se hayan comprendidos entre un máximo y un mínimo.

- Restringidos: Los valores pertenecen a un conjunto específico. Por ejemplo, sexo puede tomar los valores H o M.

Llaves

Cada fila de una relación (tabla) debe ser identificada por un valor único al que llamamos llave. A veces la llave puede ser establecida por una única columna y en otros casos la llave estará compuesta por varias.

La llave debe cumplir estos requisitos:

- Identificación unívoca. En cada fila de la tabla la llave debe identificarla de forma unívoca.

- No redundancia. No se debe descartar ningún atributo de la llave para identificar la fila.

En una fila puede producirse que más de un conjunto de atributos cumpla los requisitos anteriores, a estos atributos se les llama llaves candidatas. Uno de esos atributos ha de ser elegido como llave primaria (PK) y será el identificador de la fila. En el Modelo Relacional existen llaves candidatas, primarias y foráneas.

- La llave primaria (PK) será la columna o conjunto de columnas que permiten identificar cada ocurrencia de la tabla. Ninguna de las columnas que la forman puede tomar valores nulos.

- La llave foránea (FK) será la columna o conjunto de columnas cuyos valores se corresponden con la llave primaria de otra tabla. Las llaves foráneas representan las interrelaciones entre tablas. Según el criterio de nuestra base de datos, las llaves foráneas deberán corresponderse con los valores de las llaves primarias de las tablas a las que apuntan o contener valores nulos.
-

3.7.3 Convertir el Modelo Entidad/Relación en Modelo Relacional

Una vez realizado el modelo conceptual hay que definir el modelo lógico de datos. Los pasos a seguir son los siguientes:

- Cada entidad se convierte en una relación. El nombre de la tabla será el mismo que tiene la entidad: Persona(....).

- Los atributos de las entidades se convierten en columnas de las tablas. Se colocan junto al nombre de la tabla, encerrados entre paréntesis y separados por comas: Persona(DNI, Nombre, Apellidos, Cab_Familia,....).

- Las llaves primarias son los atributos que identifican a cada entidad, se representan en negrita o en distinto color que el resto de los atributos: Persona(**DNI**, Nombre, Apellidos, Cab_Familia,....).

- Las interrelaciones 1:N se transforman propagando la llave hacia la tabla que tiene la cardinalidad máxima N, se representa como llave foránea (FK), subrayando el atributo: Persona(DNI, Nombre, Apellidos, Cab_Familia, Cod_Viv....).

- Las interrelaciones N:N se transforman en una nueva relación (tabla) que tendrá como llave primaria (PK) la concatenación de las llaves primarias de las entidades que asocia, se representa en negrita (u otro color) y subrayado: Posee(**DNI**, **Cod_Viv**).

- Las interrelaciones 1:1 son casos particulares de la N:N o incluso también de la 1:N, por lo que no existe regla fija para su transformación, pudiéndose crear una nueva relación o propagar la llave en cualquiera de los dos sentidos (o en ambos). Para aplicar uno u otro criterio hay que basarse en las cardinalidades mínimas, en el caso de que una posea cardinalidad (0,1) y la otra (1,1) conviene propagar la llave hacia la que tiene (1,1). Si ambas tienen (0,1) la interrelación se convierte en una tabla según lo explicado para las N:N.

3.8 Lenguaje HTML

Una página web la vemos en nuestro navegador, o cliente web, y parece una sola entidad, pero no es así, está compuesta por multitud de diferentes archivos, como son las imágenes, los posibles videos y lo más importante: el código fuente.

El código de las páginas está escrito en un lenguaje llamado HTML, que es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto

de etiquetas que sirven para definir la forma en la cual presentar el texto y otros elementos de la página.

El HTML se creó en un principio con objetivos divulgativos. No se pensó que la web llegaría a ser un área de ocio con carácter multimedia, de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro.

El lenguaje consta de etiquetas que tienen esta forma <B> o <P>. Cada etiqueta significa una cosa, por ejemplo <B> significa que se escriba en negrita (bold) o <P> significa un párrafo, <A> es un enlace, etc. Casi todas las etiquetas tienen su correspondiente etiqueta de cierre, que indica que a partir de ese punto no debe de afectar la etiqueta. Por ejemplo </B> se utiliza para indicar que se deje de escribir en negrita. Así que el HTML no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que queremos aplicar a nuestro documento.

Partes de un documento HTML

Un documento HTML ha de estar delimitado por la etiqueta <html> y </html>. Dentro de este documento, podemos asimismo distinguir dos partes principales:

- El encabezado, delimitado por las etiquetas <head> y </head> donde colocaremos etiquetas de índole informativo como por ejemplo el título de nuestra página.

- El cuerpo, delimitado por las etiquetas <body> y </body>, que será donde colocaremos nuestro texto e imágenes delimitados a su vez por otras etiquetas.

3.9 Editores

El lenguaje HTML se escribe en un documento de texto, por eso necesitamos un editor de textos para escribir una página web. Así pues, el archivo donde está contenido el código HTML es un archivo de texto, con una peculiaridad, que tiene extensión .html o .htm (es indiferente cuál utilizar).

Los editores de texto para escribir en código HTML pueden ser tan sencillos o tan complejos según se requiera. Por ejemplo, para escribir directamente el código HTML a base de texto existe el Bloc de notas, si es el caso de que ya se domina el lenguaje y resulta tedioso escribir directamente el código HTML, existen un tipo de programas (editores) que nos permiten diseñar una página como si estuviéramos escribiendo un documento con un editor del tipo de Word. El editor de HTML es el encargado de vérselas con el lenguaje y programar internamente la página con el código HTML según lo que estamos diseñando.

Con el editor de HTML podemos colocar imágenes, definir estilos, utilizar negritas o cursivas, etc. sin preocuparnos de las etiquetas correspondientes a cada estilo o elemento. Es el editor el que sabe estas etiquetas y las utiliza convenientemente. Este tipo de editores HTML se denominan habitualmente WYSIWYG (What You See Is What You Get) porque cuando trabajamos con ellos lo que vemos que estamos creando con el editor es lo que obtenemos luego cuando visualizamos la página en el navegador.

Como ya se mencionó, existen dos tendencias, hablando de editores, para crear páginas web. Por un lado se tiene la posibilidad de crear las páginas escribiendo directamente el código HTML y por otro utilizando los editores de HTML. Algunas diferencias entre hacerlo de un modo u otro son las siguientes:

Escribiendo el HTML

- Se tiene un mayor dominio sobre el código de la página, y éste queda más limpio.
- Hacer una página cuesta más trabajo y tiempo.

Con un editor WYSIWYG

- El código de la página tiene peor calidad, incluso puede llegar a tener errores.
- Es muy rápido implementar una página

El escoger cualquiera de las dos formas depende del tiempo con el que se cuente para implementar una página web, pero si llegamos a profundizar en el diseño de páginas web, llega un momento en el que hace falta conocer las dos maneras de crear código HTML. A los que programan directamente HTML les hará falta aprender un editor porque eso aumentará su productividad y los que utilizan editores necesitarán aprender un poco de HTML para arreglar alguna cosa que el editor ha hecho mal o simplemente realizar alguna cosa que el editor no puede hacer.

3.9.1 EditPlus 2

Nosotros decidimos escoger como nuestro editor EditPlus 2, ya que nos ofrece herramientas muy interesantes sin dejar de escribir de alguna forma el HTML que deseamos y sin que genere código basura.

EditPlus 2 es un editor de texto creado y diseñado para todos aquellos programadores que trabajan sobre todo con HTML, aunque también es perfectamente válido para otros lenguajes de programación. El programa ofrece una amplia variedad de características incluyendo: sintaxis personalizable con soporte para HTML, ASP, JavaScript, VBScript, PERL, Java, C/C++, y muchos otros lenguajes más, un selector de documentos, herramientas definidas de usuario,

macros, accesos rápidos, corrector de ortografía, poderoso buscador y reemplazador, la habilidad de mostrar el número de línea, barra para HTML, y mucho más. El programa también incluye resalto de errores sintácticos para HTML, ASP, JavaScript, VBScript, Perl, Java, PHP, CSS, y C/C++. Admite e incluye el uso de hojas de estilo para cada lenguaje de programación con el que vayamos a utilizar este programa, y con la capacidad de crear nuestras propias hojas; opción de previsualización del resultado interna al programa (Figura 3.14).

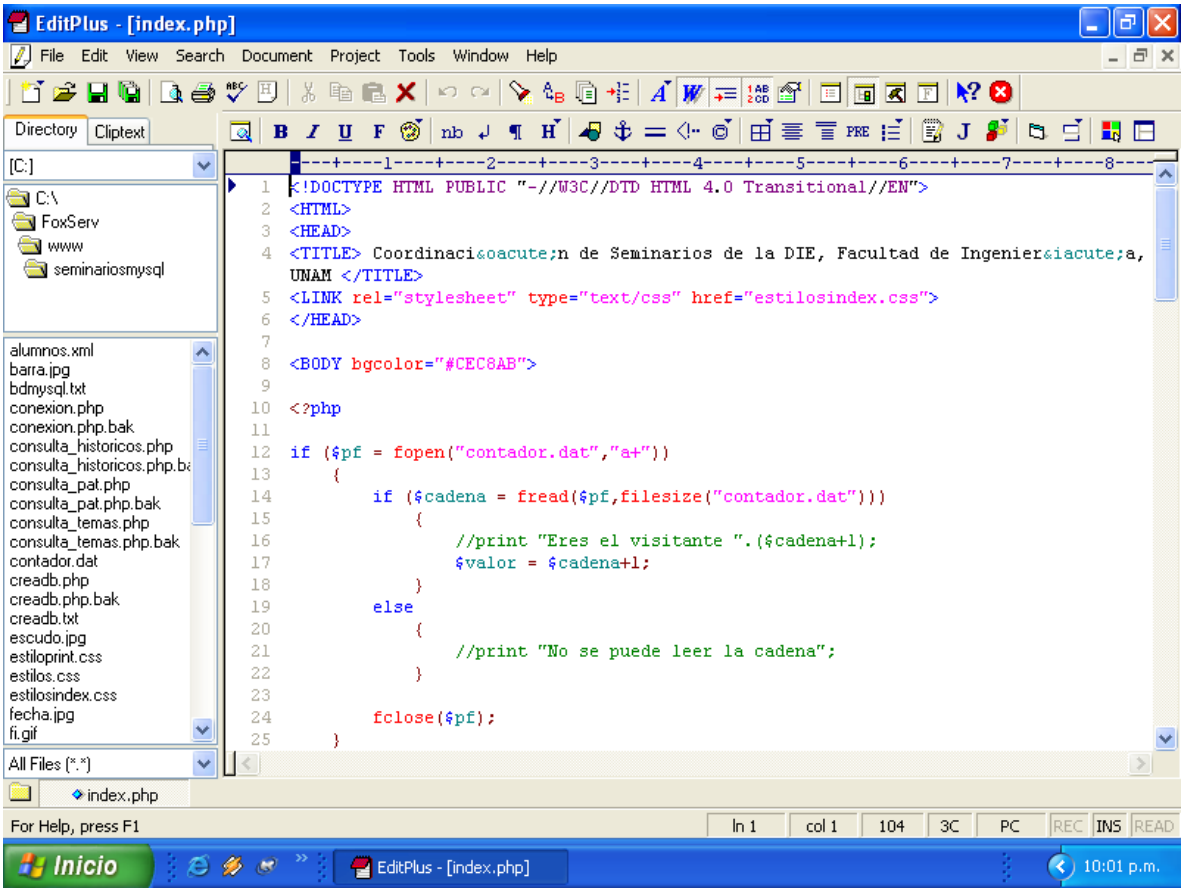


Figura 3.14 Editor EditPlus 2

3.10 CSS

CSS, es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podrían hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores, etc..

CSS son las siglas de Cascading Style Sheets, en español Hojas de Estilo en Cascada. Se trata de una especificación sobre los estilos físicos aplicables a un

documento HTML, y trata de dar la separación definitiva de la lógica (estructura) y el físico (presentación) del documento.

El estilo lógico se refiere a la lógica del documento: cabeceras, párrafos, etc., no se preocupa de la apariencia final, sino de la estructura del documento. Por el contrario, el estilo físico no se preocupa de la estructura del documento, sino por la apariencia final: párrafos con un cierto tipo de letra, tablas con un determinado color de fondo, etc..

La finalidad de las hojas de estilo es crear unos estilos físicos, separados de las etiquetas HTML, y aplicarlos en los bloques de texto en los que se quieran aplicar.

3.11 Manejo de imágenes

Como se ha podido ver anteriormente, el diseño de una página web implica la creación de un archivo en código HTML, pero no es lo único que debemos crear. En la mayoría de los casos también desearemos incluir imágenes y para ello será necesario crear los correspondientes archivos gráficos.

El proceso para incluir una imagen en una página empieza por la creación de la imagen con un programa de diseño gráfico o mediante su digitalización con un escáner. Algunos de los programas de diseño gráfico que existen en el mercado son Photoshop, Paint Shop Pro o Fireworks.

Los tipos de archivos gráficos que soporta Internet son el JPG y el GIF. Tienen características distintas y por tanto usos distintos.

Una vez que tenemos los archivos gráficos los ponemos en el mismo directorio que los archivos HTML o en un subdirectorio de éste y en el código de la página HTML pondremos una etiqueta especial para incluir la imagen, o la insertaremos con nuestro programa editor de HTML.

3.11.1 Adobe Photoshop

Aplicación informática de edición y retoque de imágenes bitmap elaborada por la compañía de software Adobe inicialmente para Apple pero posteriormente también para plataformas PC.

Photoshop en sus primeras versiones trabajaba en un espacio bitmap formado por una sola capa, donde se podían aplicar toda una serie de efectos, textos, marcas

y tratamientos. En cierto modo tenía mucho parecido con las tradicionales ampliadoras.

A medida que ha ido evolucionando el software ha incluido diversas mejoras fundamentales, como la incorporación de un espacio de trabajo multicapa, inclusión de elementos vectoriales, gestión avanzada de color (ICM / ICC), tratamiento extensivo de tipografías, control y retoque de color, efectos creativos, posibilidad de incorporar plugins de terceras compañías, exportación para web, etc..

Photoshop se ha convertido, casi desde sus comienzos, en el estándar mundial en retoque fotográfico, pero también se usa extensivamente en multitud de disciplinas del campo del diseño y fotografía: diseño web, composición de imágenes bitmap, estilismo digital, fotocomposición, edición y grafismos de video y básicamente en cualquier actividad que requiera el tratamiento de imágenes digitales.

3.12 Páginas dinámicas de servidor

Las páginas dinámicas de servidor son aquellas que son reconocidas, interpretadas y ejecutadas por el propio servidor.

Las páginas de servidor son útiles en muchas ocasiones. Con ellas se puede hacer todo tipo de aplicaciones web. Desde agendas, foros, sistemas de documentación, estadísticas, juegos, chats, etc. Son especialmente útiles en trabajos en los que se tiene que acceder a información centralizada, situada en una base de datos en el servidor, y cuando por razones de seguridad los cálculos no se pueden realizar en la máquina del usuario.

Es importante destacar que las páginas dinámicas de servidor son necesarias porque para hacer la mayoría de las aplicaciones web se debe tener acceso a muchos recursos externos a la máquina del cliente, principalmente bases de datos alojadas en servidores de Internet. Un caso claro es un banco: no tiene ningún sentido que el cliente tenga acceso a toda la base de datos, sólo a la información que le concierne.

Las páginas dinámicas de servidor se suelen escribir en un mismo archivo, mezclado con el código HTML. Cuando una página es solicitada por parte de un cliente, el servidor ejecuta los *scripts* y se genera una página resultado, que solamente contiene código HTML. Este resultado final es el que se envía al cliente y puede ser interpretado sin lugar a errores ni incompatibilidades, puesto que sólo contiene HTML.

Luego es el servidor el que maneja toda la información de las bases de datos y cualquier otro recurso, como imágenes o servidores de correo y luego envía al cliente una página web con los resultados de todas las operaciones.

Para escribir páginas dinámicas de servidor existen varios lenguajes. Common Gateway Interface (CGI) comúnmente escritos en Perl, Active Server Pages (ASP), Hipertext Preprocesor (PHP), y Java Server Pages (JSP).

Las ventajas de este tipo de programación son que el cliente no puede ver los scripts, ya que se ejecutan y transforman en HTML antes de enviarlos. Además son independientes del navegador del usuario, ya que el código que reciben es HTML fácilmente interpretable.

Como desventajas se pueden señalar que es necesario un servidor más potente y con más capacidades que el necesario para las páginas de cliente. Además, estos servidores podrán soportar menos usuarios concurrentes, porque se requerirá más tiempo de procesamiento para cada uno.

3.13 PHP

PHP es uno de los lenguajes del lado del servidor más extendidos en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la comunidad de *webmasters* debido sobre todo a la potencia y simplicidad que lo caracterizan.

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores (Figura 3.15).

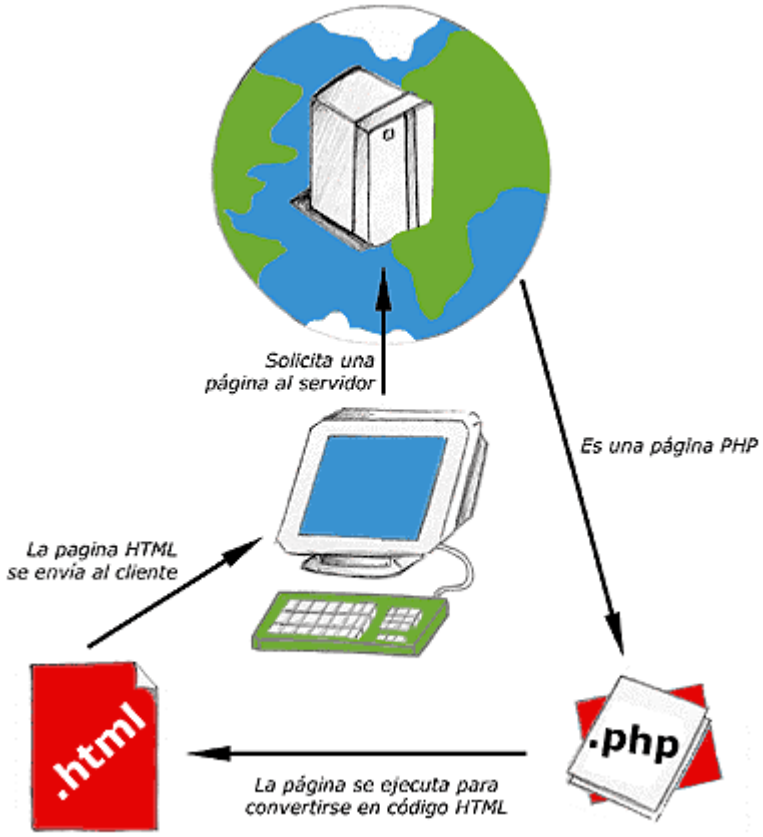


Figura 3.15 Funcionamiento de PHP

PHP se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar, al igual que ocurre con el popular ASP de Microsoft, pero con algunas ventajas como su gratuidad, independencia de plataforma, rapidez y seguridad. Cualquiera puede descargar a través de la página principal de PHP www.php.net y de manera gratuita, un módulo que hace que nuestro servidor web interprete los scripts realizados en este lenguaje. Es independiente de la plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo.

PHP, en el caso de estar montado sobre un servidor Linux o Unix, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria y esto evita las comunicaciones entre componentes COM que se realizan entre todas las tecnologías implicadas en una página ASP.

Por último señalábamos la seguridad, en este punto también es importante el hecho de que en muchas ocasiones PHP se encuentra instalado sobre servidores Unix o Linux, que son de sobra conocidos como más veloces y seguros que el sistema operativo donde se ejecuta las ASP, Windows NT o 2000. Además, PHP

permite configurar el servidor de modo que se permitan o rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual.

Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. Actualmente PHP se encuentra en su versión 4, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades de las aplicaciones web actuales.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por poner dos ejemplos.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, MSQL, Oracle, Informix, y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, *upload* de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

Servidores web soportados por PHP

- Roxen
- Zeus
- Apache
- TomCat
- IIS (Internet Information Server)
- Xitami

Requerimientos para ejecutar una página en PHP

- Servidor web
- Interprete de PHP

3.13.1 Tareas principales de PHP

Poco a poco PHP se va convirtiendo en un lenguaje que nos permite hacer de todo. En un principio diseñado para realizar poco más que un contador y un libro de visitas, PHP ha experimentado en poco tiempo una verdadera revolución y, a partir de sus funciones, en estos momentos se pueden realizar una multitud de tareas útiles para el desarrollo del web:

Funciones de correo electrónico

Podemos con una facilidad asombrosa enviar un e-mail a una persona o lista parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, persona a responder...

Otras funciones menos frecuentes pero de indudable utilidad para gestionar correos electrónicos son incluidas en su librería.

Gestión de bases de datos

Resulta difícil concebir un sitio actual, potente y rico en contenido que no es gestionado por una base de datos. El lenguaje PHP ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales podremos editar el contenido de nuestro sitio con absoluta sencillez.

Gestión de archivos

Crear, borrar, mover, modificar...cualquier tipo de operación más o menos razonable que se nos pueda ocurrir puede ser realizada a partir de una amplia librería de funciones para la gestión de archivos por PHP. También podemos transferir archivos por FTP a partir de sentencias en nuestro código, protocolo para el cual PHP ha previsto también gran cantidad de funciones.

Tratamiento de imágenes

Evidentemente resulta mucho más sencillo utilizar Photoshop para el tratamiento de imágenes pero hay ocasiones en la que hay que tratar con un número razonable de imágenes enviadas por los clientes, por poner un ejemplo. La verdad es que puede resultar muy tedioso uniformar en tamaño y formato miles de imágenes recibidas día tras día. Todo esto puede ser también automatizado eficazmente mediante PHP.

También puede parecer útil el crear botones dinámicos, es decir, botones en los que utilizamos el mismo diseño y sólo cambiamos el texto. Podremos por ejemplo crear un botón haciendo una única llamada a una función en la que introducimos el estilo del botón y el texto a introducir obteniendo automáticamente el botón deseado.

A partir de la librería de funciones graficas podemos hacer esto y mucho más.

Muchas otras funciones pensadas para Internet (tratamiento de *cookies*, accesos restringidos, comercio electrónico...) o para propósito general (funciones matemáticas, explotación de cadenas, de fechas, corrección ortográfica, compresión de archivos...) son realizadas por este lenguaje. A esta inmensa librería cabe ahora añadir todas las funciones personales que uno va creando por necesidades propias y

que luego son reutilizadas en otros sitios y todas aquellas intercambiadas u obtenidas en foros o sitios especializados.

Como puede verse, las posibilidades que se nos presentan son sorprendentemente vastas.

3.13.2 Trabajar con bases de datos en PHP

Una de las principales ventajas que presenta el trabajar con páginas dinámicas es el poder almacenar los contenidos en bases de datos. De esta forma, podemos organizarlos, actualizarlos y buscarlos de una manera mucho más simple.

El lenguaje PHP, ya hemos dicho, ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales podremos editar el contenido de nuestro sitio con absoluta sencillez.

Esta interacción se realiza, por un lado, a partir de las funciones que PHP nos propone para cada tipo de base de datos y, por otro estableciendo un diálogo a partir de un idioma universal: SQL (Structured Query Language) el cual es común a todas las bases de datos.

Entre algunas Bases de Datos que soporta el leguaje PHP tenemos:

- Adabas
- FilePro
- IBM DB2
- Informix
- Velocis
- Dbase
- InterBase
- FrontBase
- MSQL
- MySQL
- Empress
- Oracle
- PostgreSQL
- Solid
- Sybase

3.13.3 Subir una aplicación PHP al servidor

Nuestro servidor web debe tener un directorio para la publicación de las páginas web. Ese sería el lugar donde hay que subir los archivos .php.

Dependiendo del proveedor con el que trabajemos, el directorio de publicación puede variar. Generalmente, cuando contratamos un alojamiento, nos proporcionan una cuenta de FTP con la cual conectarnos al servidor web y transferir los archivos de nuestro sitio, además de unos datos para la conexión, que serán el nombre del servidor, el usuario y contraseña para el acceso al FTP.

Al conectarnos al servidor con los datos del FTP, que deben ser proporcionados por nuestro proveedor, accederemos a un directorio. Este directorio podría ser el de publicación, aunque generalmente no es así, sino que suele ser un subdirectorio llamado "HTML" o "docs" o algo similar, que cuelga del directorio de inicio en nuestra conexión FTP. Este directorio puede tener nombres distintos en proveedores distintos.

Los archivos se deben subir al directorio de publicación, o a cualquier subdirectorio de éste. Para acceder a ellos bastaría con escribir el nombre del dominio o URL de nuestro alojamiento, seguido del nombre del archivo. Si tuviésemos un archivo llamado hola.php y nuestro alojamiento se ha contratado para el dominio www.midominio.com, deberíamos subir ese archivo al directorio de publicación y accederíamos al archivo escribiendo:

http://www.midominio.com/hola.php

Si creamos subdirectorios dentro del directorio de publicación podremos acceder a ellos escribiendo el nombre del dominio o URL de nuestro alojamiento, seguido del nombre del directorio y el nombre del archivo. Por ejemplo, si creamos un subdirectorio llamado paginas y tenemos dentro un archivo llamado pag1.php, podríamos acceder a él de la siguiente manera.

http://www.midominio.com/paginas/pag1.php

3.14 RDBMS

Un RDBMS es un Sistema Administrador de Bases de Datos Relacional. RDBMS viene del acrónimo en inglés Relational DataBase Management System.

Los RDBMS proporcionan el ambiente adecuado para gestionar una base de datos.

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa.

Hoy en día, son muchas las aplicaciones que requieren acceder a datos. Bien sea un sencillo programa doméstico, bien una *suite* para la gestión empresarial. Estos datos se deben almacenar en algún soporte permanente, y las aplicaciones deben disponer de un medio para acceder a ellos. Normalmente, la forma en que un programa accede a un fichero es a través del sistema operativo. Este provee de funciones como abrir archivo, leer información del archivo, guardar información, etc. No obstante, este procedimiento de acceso a archivos es altamente ineficaz cuando se trata con un volumen elevado de información. Es aquí donde aparecen los Sistemas Gestores de Bases de Datos: proporcionan un interfaz entre aplicaciones y sistema operativo, consiguiendo, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, más fácil de implementar, y, sobre todo, más segura.

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los usuarios de los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

- **Consistencia**. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentre asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado

pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia**. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

SGBD libres

- PostgreSQL
- MySQL
- Firebird
- SQLite

SGBD comerciales

- dBase
- Fox Pro
- Informix
- MAGIC
- Microsoft Access
- Microsoft SQL Server
- MySQL
- Oracle
- Paradox
- PervasiveSQL

3.15 Manejadores de Bases de Datos PostgreSQL y MySQL

Hoy en día existen muchas empresas y sitios web que necesitan mantener de forma eficiente un gran volumen de datos. Muchos de ellos optan por soluciones comerciales, aunque muchas otras confían en el software libre optando por una solución como PostgreSQL o MySQL.

Como en un principio nosotros propusimos como nuestro manejador de base de datos a MySQL, y después por sugerencia del administrador del servidor trabajamos finalmente con PostgreSQL, haremos una comparativa entre estos sistemas de gestión de bases de datos libres que son los más importantes y más usados en la red, los cuales proporcionan soluciones a miles de personas, de forma totalmente gratuita, sin pérdida de eficiencia alguna. Esto con el fin de conducir a la elección más adecuada para cada situación.

3.15.1 PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el National Science Foundation (NSF), y ESL, Inc.

PostgreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99, así como otras características que comentaremos más adelante.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Características de PostgreSQL

A continuación se enumeran las principales características de este gestor de bases de datos:

1. Implementación del estándar SQL92/SQL99.

2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP ...), cadenas de bits, etc. También permite la creación de tipos propios.

3. Incorpora una estructura de datos *array*.

4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.

5. Permite la declaración de funciones propias, así como la definición de disparadores.

6. Soporta el uso de índices, reglas y vistas.

7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.

8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

¿Qué es lo que le falta?

PostgreSQL es un magnífico gestor de bases de datos, capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible. Ésta es, un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable.

3.15.2 MySQL

MySQL es un Sistema de Gestión de Bases de Datos Relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Características de MySQL

Las principales características de este gestor de bases de datos son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

2. Soporta gran cantidad de tipos de datos para las columnas.

3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).

4. Gran portabilidad entre sistemas.

5. Soporta hasta 32 índices por tabla.

6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

¿Qué es lo que le falta?

MySQL surgió como una necesidad de un grupo de personas sobre un gestor de bases de datos rápido, por lo que sus desarrolladores fueron implementando únicamente lo que precisaban, intentando hacerlo funcionar de forma óptima. Es por ello que, aunque MySQL se incluye en el grupo de sistemas de bases de datos relacionales, carece de algunas de sus principales características:

1. Subconsultas: tal vez ésta sea una de las características que más se echan en falta, aunque gran parte de las veces que se necesitan, es posible reescribirlas de manera que no sean necesarias.

2. SELECT INTO TABLE: Esta característica propia de Oracle, todavía no está implementada.

3. *Triggers* y *Procedures*: Se tiene pensado incluir el uso de *procedures* almacenados en la base de datos, pero no el de *triggers*, ya que los *triggers* reducen de forma significativa el rendimiento de la base de datos, incluso en aquellas consultas que no los activan.

4. Transacciones: a partir de las últimas versiones ya hay soporte para transacciones, aunque no por defecto (se ha de activar un modo especial).

5. Integridad referencial: aunque sí admite la declaración de claves ajenas en la creación tablas, internamente no las trata de forma diferente al resto de campos.

Los desarrolladores comentan en la documentación que todas estas carencias no les resultaban un problema, ya que era lo que ellos necesitaban. De hecho, MySQL fue diseñada con estas características, debido a que lo que buscaban era un gestor de bases de datos con una gran rapidez de respuesta. Pero ha sido con la distribución de MySQL por Internet, cuando más y más gente les están pidiendo estas funcionalidades, por lo que serán incluidas en futuras versiones del gestor.

3.15.3 Comparativa

Son muchos los *benchmarks* que se han publicado sobre estos gestores de bases de datos, aunque muchos de ellos tienen una clara tendencia hacia uno de los dos bandos. Es por esto que hay que saber extraer bien las conclusiones a partir de un *benchmark*. Por ejemplo, es importante que las comparaciones entre los dos gestores se realicen en igualdad de condiciones, cosa que por otra parte, muchas veces resulta complicado debido a las distintas implementaciones que poseen estos gestores.

Además, resulta muy complicado diseñar un buen *benchmark*, que tenga en cuenta todas las posibles mejoras de rendimiento que pueda aplicar cada uno de estos gestores. Es lógico pues, que haya algunas pruebas que se le apliquen a un gestor, y que no tenga ningún sentido realizárselas al otro.

A continuación se resumen las conclusiones obtenidas a partir de diversos *benchmarks*, intentando hacer un descarte de los que tenían una clara tendencia hacia uno de los bandos.

Lo mejor de PostgreSQL

Las características positivas que posee este gestor según las opiniones más comunes en Internet, son:

1. Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos *benchmarks* se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).

2. Implementa el uso de *rollbacks*, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.

3. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Lo peor de PostgreSQL

Por contra, los mayores inconvenientes que se pueden encontrar a este gestor son:

1. Consume gran cantidad de recursos.

2. Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.

3. Es de 2 a 3 veces más lento que MySQL.

Lo mejor de MySQL

Es evidente que la gran mayoría de gente usa este gestor en Internet, por lo que encontrar opiniones favorables no ha resultado en absoluto complicado:

1. Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.

2. Su bajo consumo lo hace apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

3. Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.

4. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

5. El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro y de buscadores de aplicaciones.

Lo peor de MySQL

Debido a su gran aceptación en Internet, gran parte de los inconvenientes que se exponen a continuación, han sido extraídos de comparativas con otras bases de datos:

1. Carece de soporte para transacciones, *rollbacks* y subconsultas.

2. El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.

3. No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

3.16 Subir nuestro portal al servidor

Básicamente lo que tenemos que hacer es tomar todos los archivos que componen nuestro sitio web, incluidas imágenes, animaciones, etc. y subirlas a nuestro servidor web. Para ello primero es tarea imprescindible el identificar dónde están todos los archivos de nuestro sitio web.

Dependiendo del alojamiento que tengamos, la manera de subir los archivos cambiará. Existen dos maneras de subir los archivos al servidor, por FTP o a través de una interfaz web.

Subir archivos por FTP

La forma más tradicional de subir archivos es por FTP, que es un servicio más de Internet que se utiliza para transferir archivos por la red. Como lo que queremos hacer es transferir los archivos desde nuestra máquina al servidor, este es el servicio que debemos utilizar.

FTP es uno de los diversos protocolos de la red, concretamente significa File Transfer Protocol (Protocolo de Transferencia de Archivos) y es el ideal para transferir grandes bloques de datos por la red. Se precisa de un Servidor de FTP y un cliente FTP. La mayoría de las páginas web a nivel mundial son subidas a los respectivos servidores mediante este protocolo.

Por defecto utiliza los puertos 20 y 21. El puerto 20 es el utilizado para el flujo de datos entre el cliente y el servidor y el puerto 21 para el flujo de control, es decir, para enviar las órdenes del cliente al servidor. Mientras se transfieren datos a través del flujo de datos, el flujo de control permanece en espera.

Cuando tenemos un alojamiento profesional para nuestro sitio lo más seguro es que nos proporcionen un acceso por FTP a los servidores para subir las páginas.

Como otros servicios de Internet, para utilizar FTP necesitamos un programa especial que se denomina cliente de FTP. Podemos encontrar en el mercado muchos de estos clientes, algunos populares son Cute FTP o FTP Voyager.

Todos los programas de FTP son parecidos, básicamente consisten en una ventana que está partida en dos partes. En una parte podemos ver nuestro disco duro, con sus distintas unidades y carpetas. En la otra parte se puede ver el sistema de archivos del servidor, con sus correspondientes carpetas. Para mover los archivos de un lugar a otro suele bastar con arrastrarlos de una parte de la ventana a la otra.

Una tarea que también puede ser complicada en un principio puede ser el configurar el programa de FTP para que acceda al espacio que tenemos asignado. Los datos de configuración son:

Nombre del servidor FTP: suele tener una forma como ftp.tudominio.com

Usuario: tu nombre de usuario

Password: tu palabra de clave

Podría haber algún dato adicional, como el directorio por defecto, que es el directorio en el que se desea abrir la sesión, pero no es habitual que se nos proporcione este dato porque los accesos por FTP suelen estar configurados para que se acceda directamente al directorio donde están nuestras páginas.

Acceso con interfaz web

Es muy típico que los proveedores de alojamiento gratuito provean de una herramienta de muy fácil uso para subir las páginas. A esta herramienta se accede a través de un navegador web y no es más que un formulario donde se pueden elegir los archivos que se desean subir al servidor, se pulsa un botón, y listo.

3.17 Estructura del portal

Una vez que tenemos preparado el contenido del portal, el siguiente paso es diseñar su estructura con el propósito de facilitar al máximo la navegación dentro de él a los visitantes. Los elementos de navegación y orientación tienen como función básica informar constantemente al usuario acerca de dónde se encuentra, que relación tiene el nodo web que está visualizando respecto al resto de la arquitectura del portal, dónde ha estado y hacia dónde puede ir.

Los puntos que consideramos para diseñar la estructura del portal son los siguientes:

1. Coherencia del diseño

La primera regla para indicarle a nuestro usuario que sigue estando en nuestro portal es manteniendo una coherencia de diseño, es decir, una uniformidad en la estructura de las páginas que forman nuestro portal.

Con el uso de los colores también debemos mantener cierta uniformidad: si de una página a otra cambian los colores completamente, un usuario (no necesariamente despistado) puede pensar que ha sido reenviado a otro sitio web, es decir, puede sentirse perdido.

2. Jerarquía visual

Una vez que conseguimos, en cierta forma, que el usuario sepa que está en nuestro portal, el siguiente paso es que sepa en qué zona de éste se encuentra exactamente.

Los usuarios (en occidente) leen de izquierda a derecha y de arriba hacia abajo. Esto significa que si mantenemos la jerarquía visual podemos indicarle al usuario constantemente dónde está. Cuanto más cerca de la esquina superior izquierda de nuestro lay-out (distribución) coloquemos los elementos, mayor nivel jerárquico tendrán, y conforme los coloquemos más hacia la derecha inferior, menor nivel jerárquico, y por lo tanto serán "partes de". La Figura 3.16 muestra la jerarquía visual de nuestro portal.

Figura 3.16 Jerarquía visual

1. **Encabezado.** Incluye una imagen con el escudo y nombre de la Facultad de Ingeniería.

2. **Barra de navegación y ligas de interés.** Esta zona incluye todos los enlaces a todas las secciones del portal, así como las ligas a sitios relacionados con nuestro portal.

3. **Contenido.** Información referente a la sección en la que nos encontramos, esta zona incluye ligas secundarias.

3. Estructura

Este punto se refiere a la estructura en conjunto de todo el portal, es decir, la forma en que están organizadas todas las páginas que conforman el portal así como la relación que guardan entre ellas. De acuerdo a las necesidades de nuestro portal, utilizamos la estructura de árbol o jerárquica que parte de una página principal (raíz), a partir de la cual se puede acceder a las diferentes secciones del portal (ramas), que a su vez contienen ligas que nos llevan a páginas terciarias (hojas). La Figura 3.17, muestra la estructura de árbol de nuestro portal.

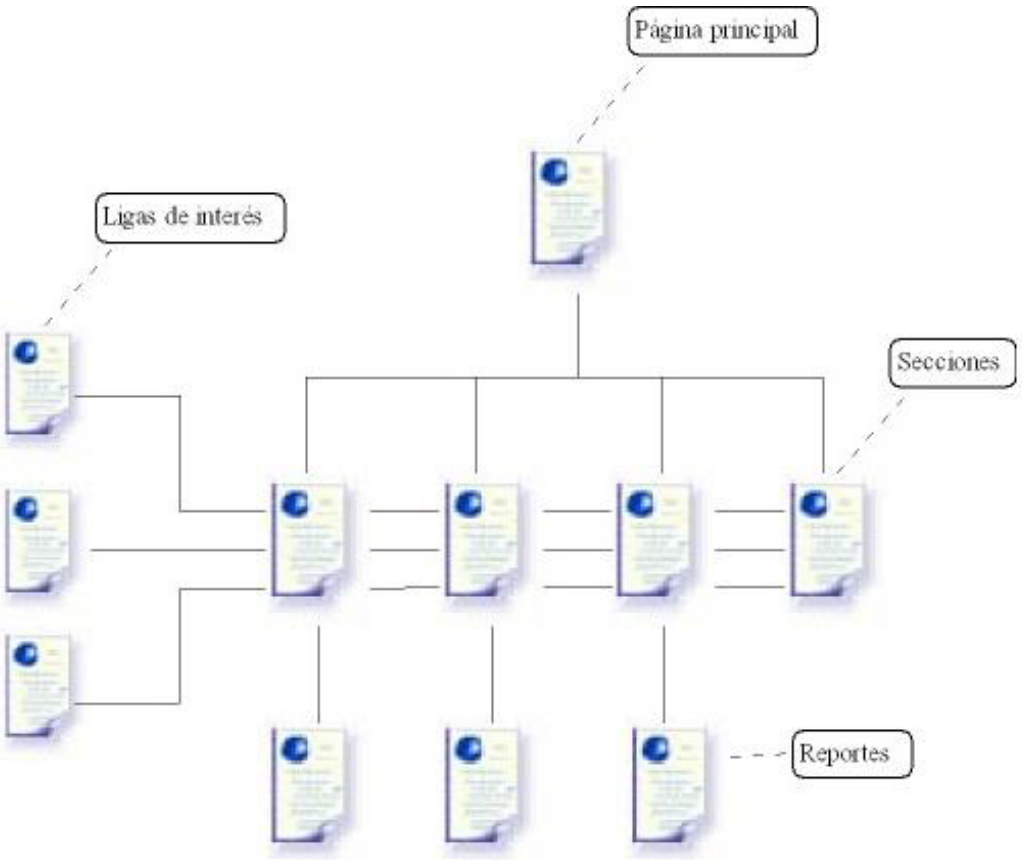


Figura 3.17 Estructura

Página Principal. Esta página contiene mensaje de bienvenida al portal, así como información acerca de lo que se puede encontrar en él. También incluye un contador de visitas con el propósito de saber que tan visitado es.

Secciones. Cada una de estas páginas contiene información referente a la sección en la que nos encontramos, ligas de interés, así como ligas secundarias.

Reportes. Incluyen listado de los temas de seminarios y/o tesis de cada una de las secciones.

Ligas de interés. Páginas que contienen información de interés para los usuarios que visitan el portal.

Conclusiones

El sistema actualmente ya se encuentra terminado y se ha instalado en la Coordinación de Seminarios y Servicio Social de la DIE, se han hecho las pruebas pertinentes y ha sido aceptado para una próxima publicación a toda la comunidad de la Facultad de Ingeniería y en general a cualquier persona que cuente con acceso a Internet.

De este sistema se concluye lo siguiente:

- El modelo en espiral, dentro de los paradigmas del desarrollo de software es, actualmente, la tendencia óptima para el análisis y diseño de sistemas toda vez que incluye iteraciones que permiten desarrollar versiones más completas del software a medida que avanza el proceso y tanto desarrollador como cliente comprenden mejor el funcionamiento del mismo.

- La arquitectura Cliente/Servidor es el modelo base de desarrollo para sistemas web, entre sus principales ventajas es que gran parte del procesamiento es repartida en los clientes. Idealmente, un cliente se conecta al servidor, obtiene la información de su interés y cierra la conexión, reduciendo considerablemente el tráfico en la red. Una de las ventajas principales de esta arquitectura es que el cliente no sólo puede tener un acceso transparente a toda la información que necesita, sino además está habilitado para procesarla como guste.

- UML es el lenguaje de modelado de sistemas que se aplica perfectamente en el análisis y diseño de la Ingeniería de Software, ya que permite tener una visión dinámica del comportamiento del sistema.

- PHP es sin duda una de las mejores opciones para desarrollo web por su gratuidad, su política de código abierto, independencia de plataforma, rapidez, seguridad y compatibilidad con las bases de datos más comunes, entre muchas otras ventajas.

- PostgreSQL y MySQL son los dos principales manejadores de bases de datos de distribución gratuita usados en la red. Nos permiten junto con los lenguajes de programación web mantener un sitio web con un gran volumen de datos y presentarlos de forma dinámica. El optar por uno u otro dependerá básicamente de las necesidades del sistema.

- El World Wide Web (WWW) es actualmente el medio de comunicación por excelencia para la difusión de información a multitud de usuarios de forma simultánea, fácil y a un costo reducido.

Finalmente, el desarrollo de este sistema nos deja la experiencia de haber abordado un problema real, con todos sus pros y contras, crear una solución, y lo más importante, su implementación.

Anexo A. Manual de instalación

Requerimientos de hardware y software para el correcto funcionamiento del sistema:

En el servidor:

- Sistema operativo Windows 95, Windows 98, Windows 2000, Windows XP, Windows NT, Linux, Unís, OS/2, etc.. Es decir, prácticamente cualquier sistema operativo que maneje ambientes de red.
- Un servidor web instalado como por ejemplo, Roxen, Zeus, Apache, TomCat, IIS (Internet Information Server), Xitami.
- Intérprete de PHP (activar las siguientes librerías: php_domxml, php_pgsql).
- Manejador de bases de datos PostgreSQL.
- En cuanto a los requerimientos de hardware, éstos no difieren de los necesarios para contar con el sistema operativo que se haya elegido.

Las características del servidor de la DIE donde fue instalado el portal son las siguientes:

- Pentium III 700 Mhz.
- Memoria: 512 Mb. en RAM.
- Sistema Operativo: Linux.
- Intérprete de PHP.
- Manejador de base de datos: PostgreSQL.
- Servidor web: Apache.

En el cliente:

Bloque de administración y actualización:

- Microsoft Visual Basic 6.0 (instalar librería Microsoft Data Access Components y activar las siguientes referencias: Microsoft ActiveX Data Objects 2.7 Library, Microsoft XML, v3.0).

Usuarios finales:

- Por el lado del cliente, no se requiere un sistema operativo específico. Lo único necesario es que cuente con un navegador de web. Tales navegadores pueden ser Netscape 4.5 o posterior, Internet Explorer 4.0 o posterior.

El sistema completo consta de los siguientes archivos:

En el servidor:

El portal está compuesto de los siguientes archivos (todos los archivos deben colocarse en la misma carpeta):

- **Archivos portal:** consulta_historicos.php, consulta_pat.php, consulta_temas.php, historicos.php, index.php, informesycreditos.php, pat.php, principal.php.

- **Hojas de estilo:** estiloprint.css, estilos.css, estilosindex.css.

- **Imágenes y animaciones:** barra.jpg, escudo.jpg, fecha.jpg, fi.gif, fi.jpg, historicos.jpg, imgleft.jpg, imgright.jpg, infycre.jpg, ligas.jpg, lupa.jpg, pat.jpg, relleno.jpg, servsoc.jpg, temas.jpg, unam.swf.

- **Contador de visitas:** contador.dat.

- **Archivos de creación y conexión de la base de datos:** conexion.php, creadb.php.

- **Archivos de actualización:** alumnos.xml, inserta.php, insertp.php, insertt.php, profesores.xml, temas.xml.

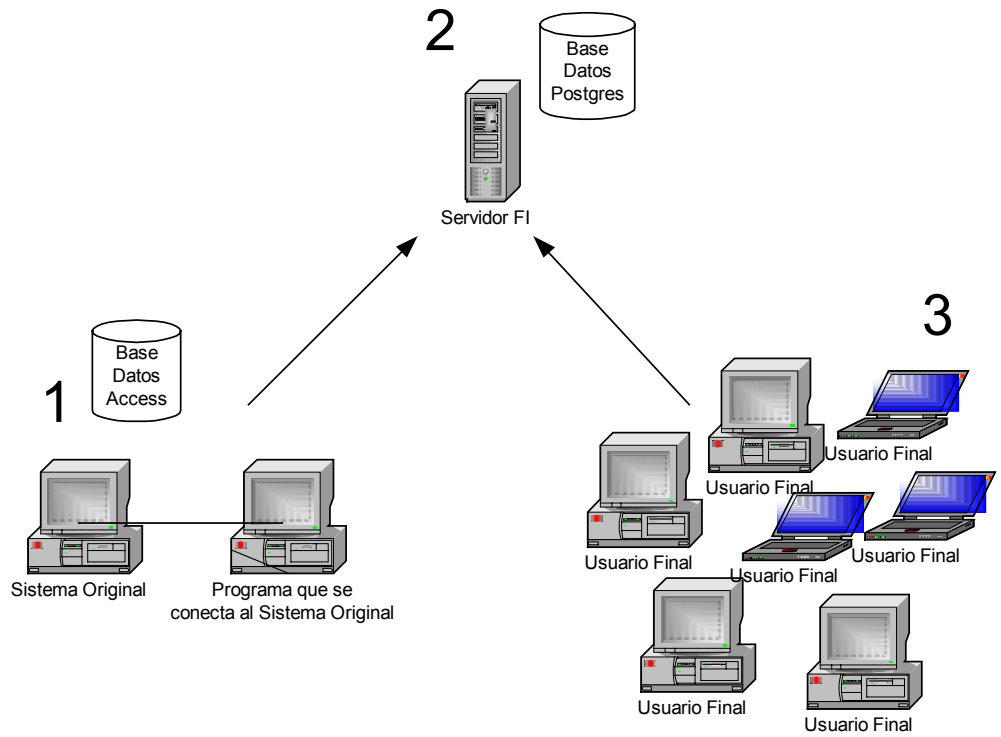
En el cliente:

La parte de administración y actualización está compuesta de los siguientes archivos:

- El proyecto (archivo) llamado proySistemaDIE.vbp contiene los siguientes archivos: frmSistemaDIE.frm, ModuleSistemaDIE.bas (todos los archivos deben colocarse en la misma carpeta).

Anexo B. Manual de administrador

En el siguiente diagrama se muestran los bloques de los que está compuesto el sistema.



A continuación se describe el funcionamiento de cada uno de los bloques:

1. Sistema Coordinación de Seminarios y Servicio Social de la DIE.

En esta parte se encuentran dos computadoras conectadas en red, una de ellas es la que aloja al sistema que administra todos los procesos referentes a los seminarios y servicio social de la DIE, la otra es la computadora destinada a llevar a cabo el proceso de actualización de la página web (consulta, transferencia de archivos al servidor, y actualización de la base de datos del servidor), proceso que se describe a detalle más adelante.

2. Servidor de la División de Ingeniería Eléctrica.

En esta parte se encuentran alojados todos los archivos que conforman el portal en el que se publicarán los temas aprobados para seminarios y/o tesis de la DIE (scripts de php, hojas de estilo, imágenes), así como los archivos que nos ayudarán a mantenerlo actualizado (archivos xml que contienen actualizaciones, scripts de php que descargan los archivos xml en la base de datos del servidor).

3. Usuarios finales.

En este último bloque se encuentran todas aquellas personas (alumnos, profesores, directores, etc.) que deseen a través de la página de la Coordinación de Seminarios de la DIE (**http://alumnosdie.fi-b.unam.mx:443/alum/scsss/**) consultar el banco de información de temas y directores para seminarios y/o tesis (tanto vigentes como históricos).

Actualización del portal.

Con el propósito de que cualquier modificación o actualización hecha al sistema de la Coordinación se vea reflejado en el portal en el que se publicarán los temas aprobados, se deberá llevar a cabo la actualización mediante los siguientes tres pasos:

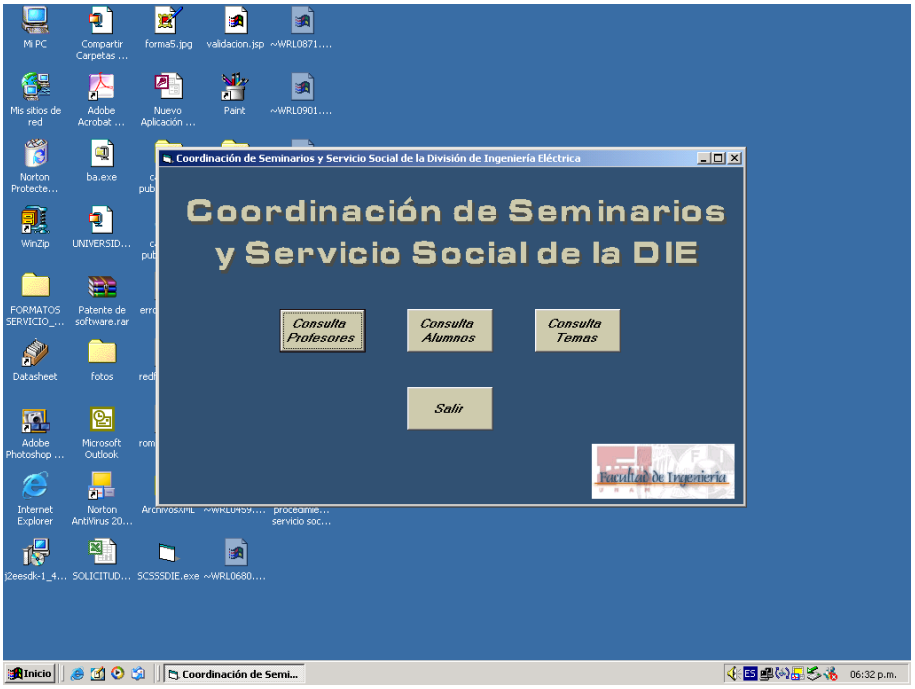
1. Consulta al sistema de la Coordinación de Seminarios y Servicio Social de la DIE para formar los archivos en lenguaje XML (profesores.xml, alumnos.xml y temas.xml), que contendrán todas las actualizaciones hechas al sistema.

2. Envío de los archivos xml al servidor.

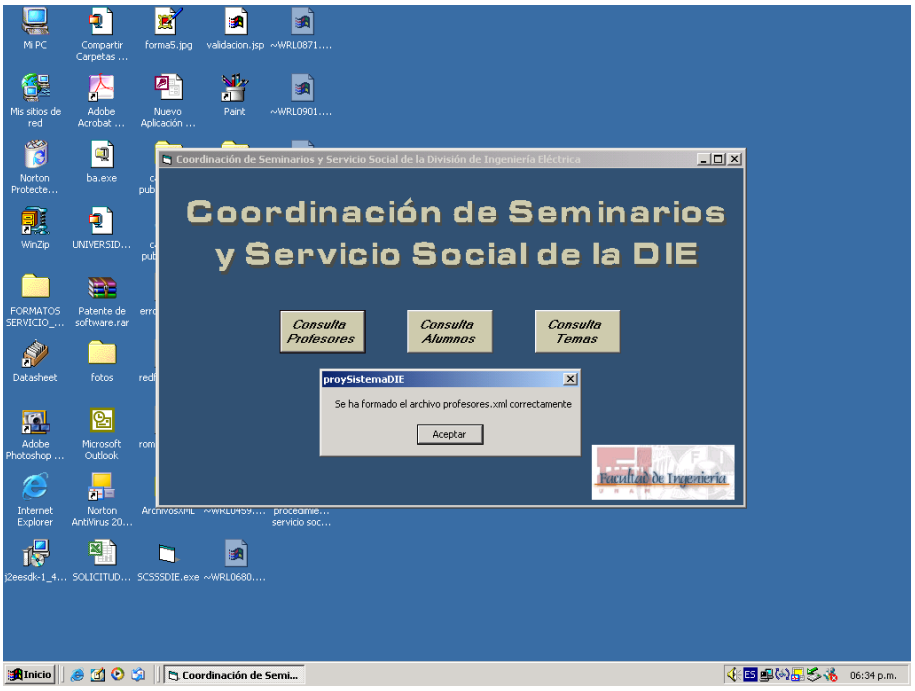
3. Actualización de la base de datos del servidor (descarga de los archivos xml en la base de datos).

1. Consulta al sistema de la Coordinación de Seminarios y Servicio Social de la DIE.

Para hacer la consulta al sistema se deberá dar doble clic sobre el archivo ejecutable SCSSSDIE.exe, ubicado en el Escritorio. Enseguida aparecerá la siguiente pantalla:



Se deberá dar clic, uno por uno, en los tres botones que aparecen, en caso de que se haya realizado la operación con éxito el programa enviará un mensaje señalando que el archivo xml correspondiente se ha formado correctamente, aquí se deberá aceptar el mensaje.



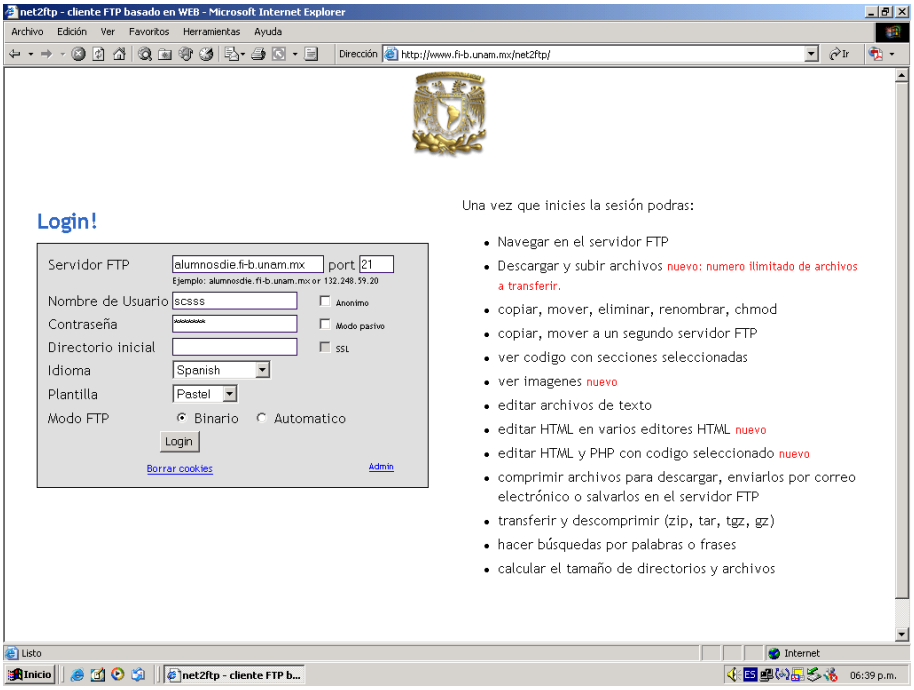
Una vez formados los tres archivos: profesores.xml, alumnos.xml y temas.xml, dar clic en el botón “Salir” para terminar con esta primera parte. Los archivos formados

se almacenarán automáticamente en la carpeta “Archivos XML” ubicada en el Escritorio.

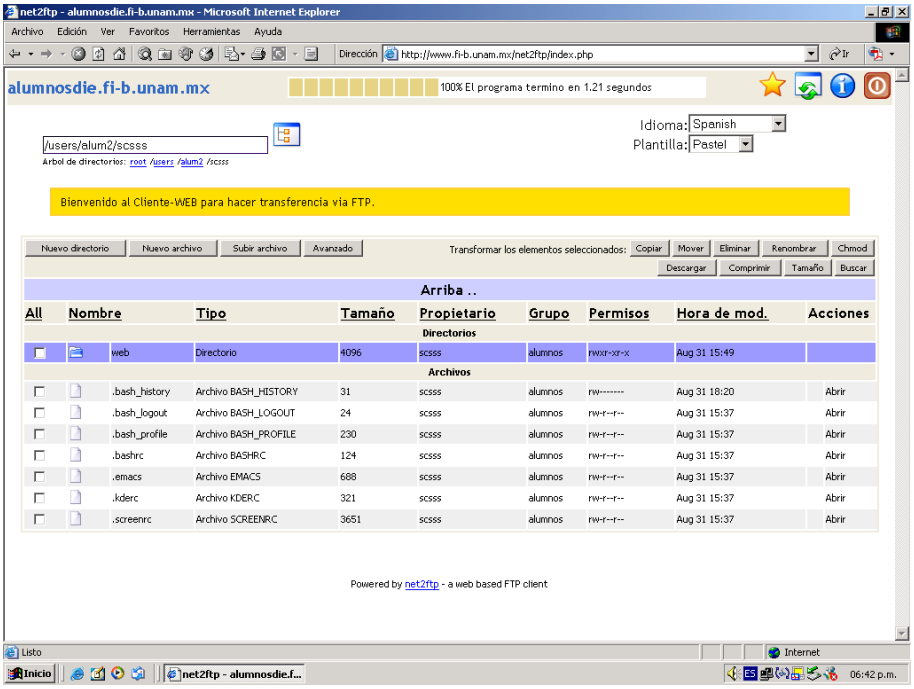
2. Envío de los archivos xml al servidor.

A continuación se procede a subir los archivos xml al servidor para lo cual se deberá acceder a la siguiente página: **http://www.fi-b.unam.mx/net2ftp/**, para tener acceso se deberán llenar únicamente los siguientes campos:

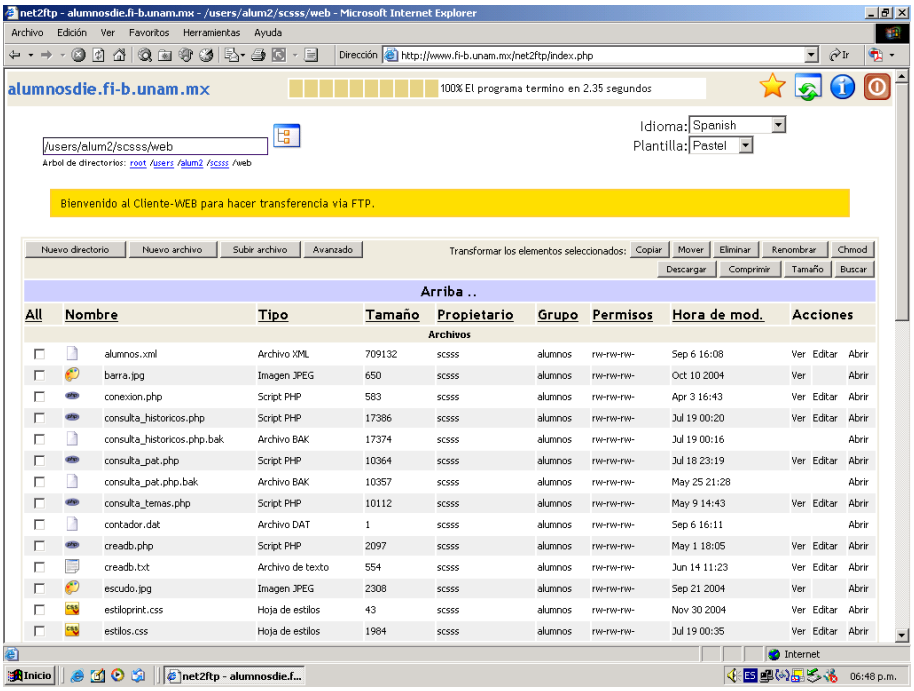
Nombre de Usuario: scsss
Contraseña: *******



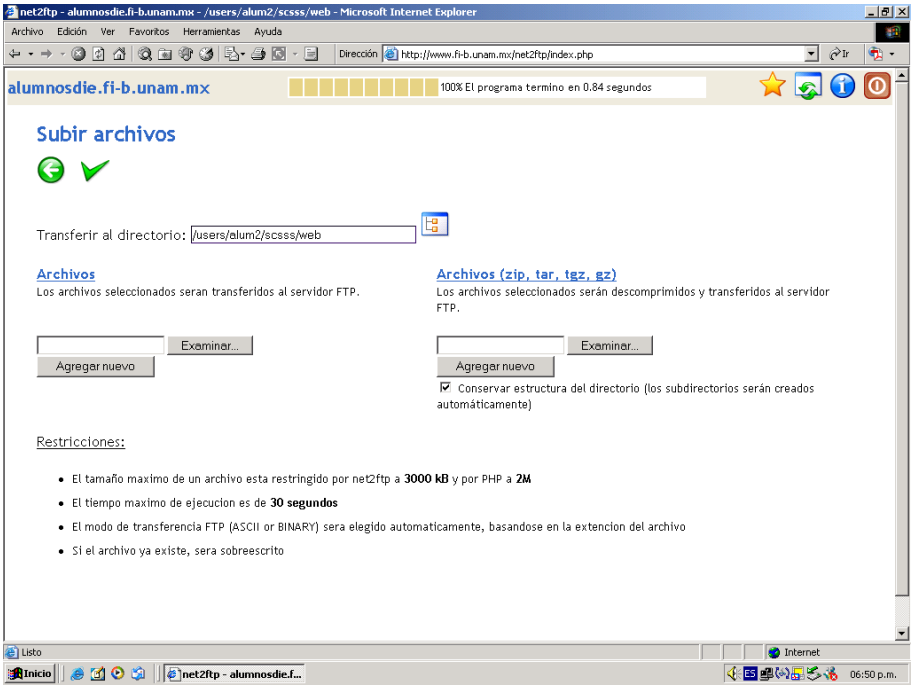
Al dar clic en el botón Login o dar Enter, aparecerá la siguiente pantalla en la que se deberá dar clic sobre la carpeta “web”.



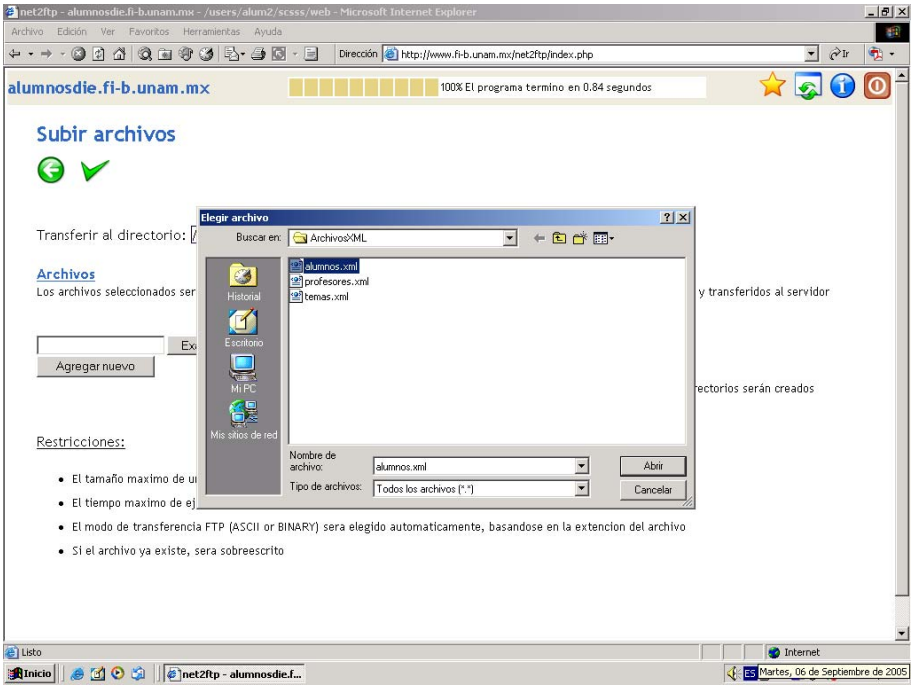
Enseguida estaremos accediendo a la página en la que se encuentran todos los archivos de nuestro portal, aquí se deberá dar clic sobre la opción “Subir archivo”.



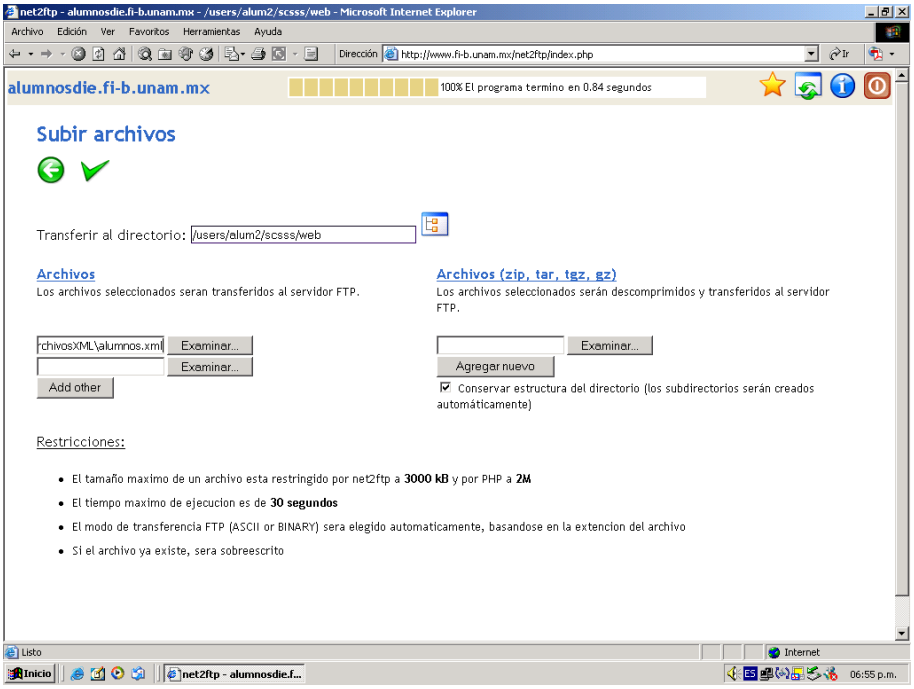
Aparecerá la siguiente pantalla en la que se procederá a subir los archivos (profesores.xml, alumnos.xml y temas.xml), para ello en la parte de Archivos (parte izquierda) se deberá dar clic en el botón “Examinar…”.



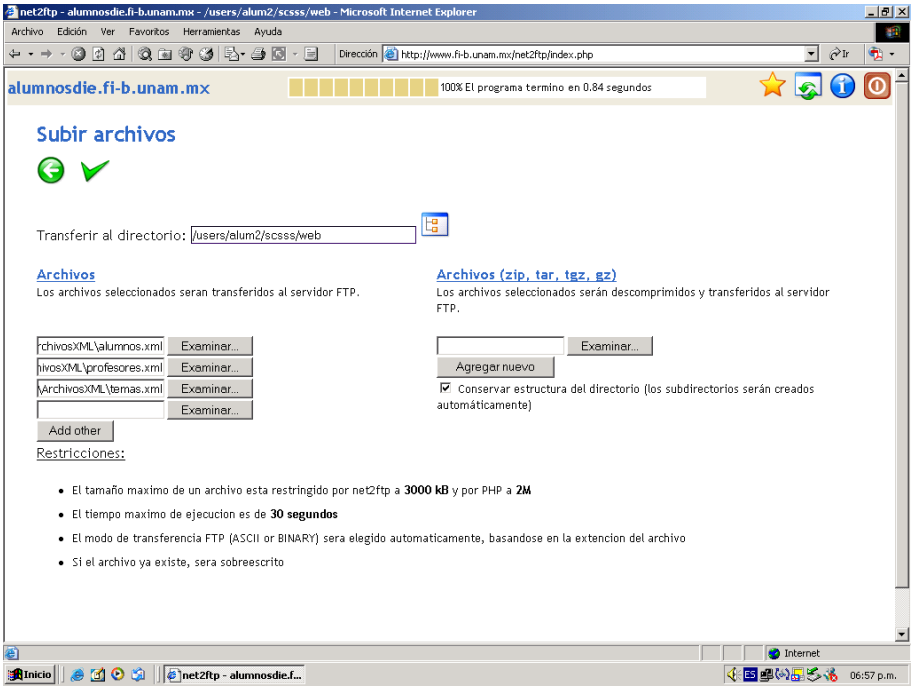
A continuación aparecerá la siguiente sub-ventana en la cual se deberá seleccionar el archivo xml (en este caso alumnos.xml) ubicado en la carpeta Archivos XML del Escritorio.



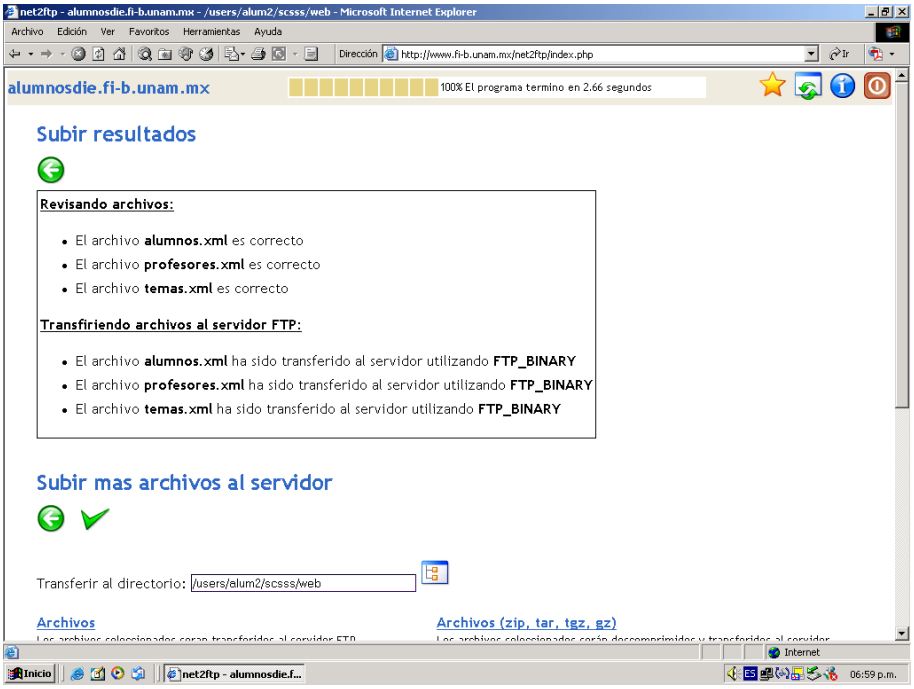
Una vez seleccionado el archivo se deberá dar clic sobre el botón “Abrir”, con lo que habremos indicado la ruta del archivo en cuestión.



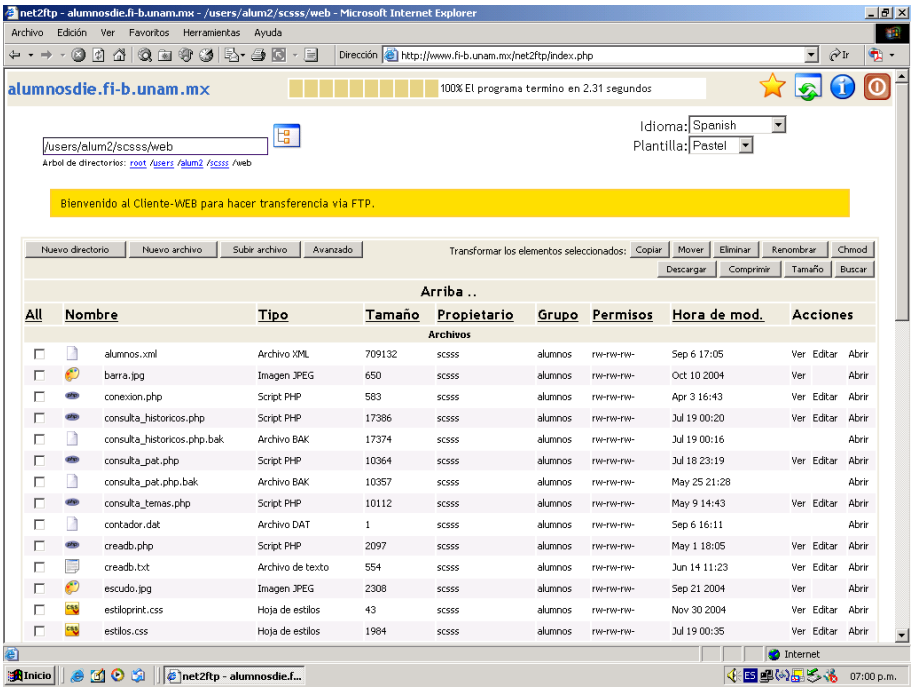
Se deberán repetir estos mismos pasos para los 2 archivos restantes, con lo que tendremos la pantalla siguiente. Para proceder a subir los archivos se dará clic en la “paloma”.



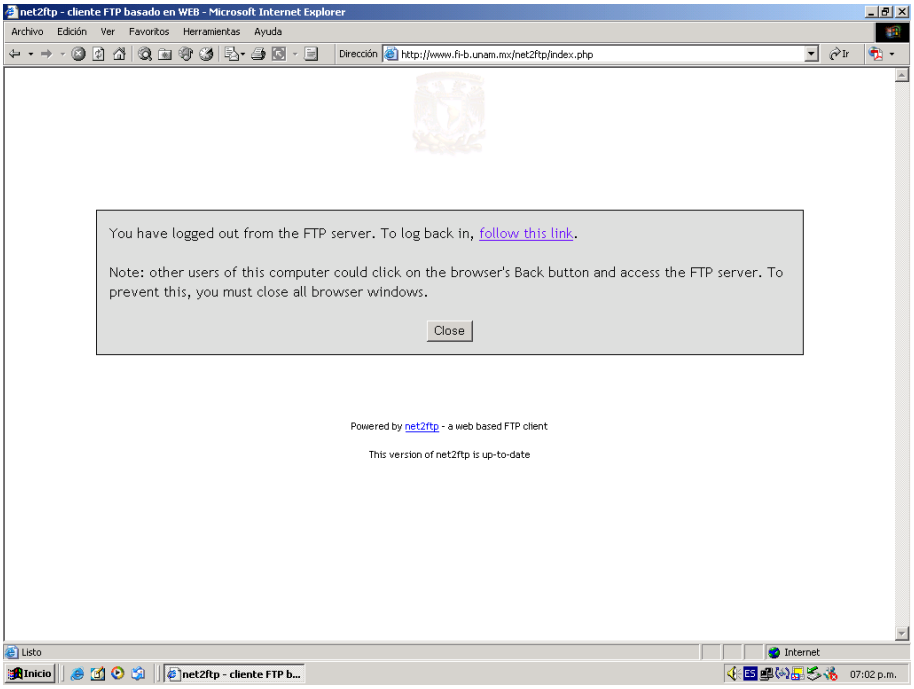
El proceso de subir los archivos tomará alrededor de 3 minutos, al término de los cuales deberá aparecer la siguiente pantalla indicando que se han transferido correctamente al servidor. Se dará clic en la flecha para regresar a la pantalla principal.



Ya en la pantalla principal se deberá cerrar la sesión, dando clic en el botón rojo que se encuentra en la parte superior derecha.



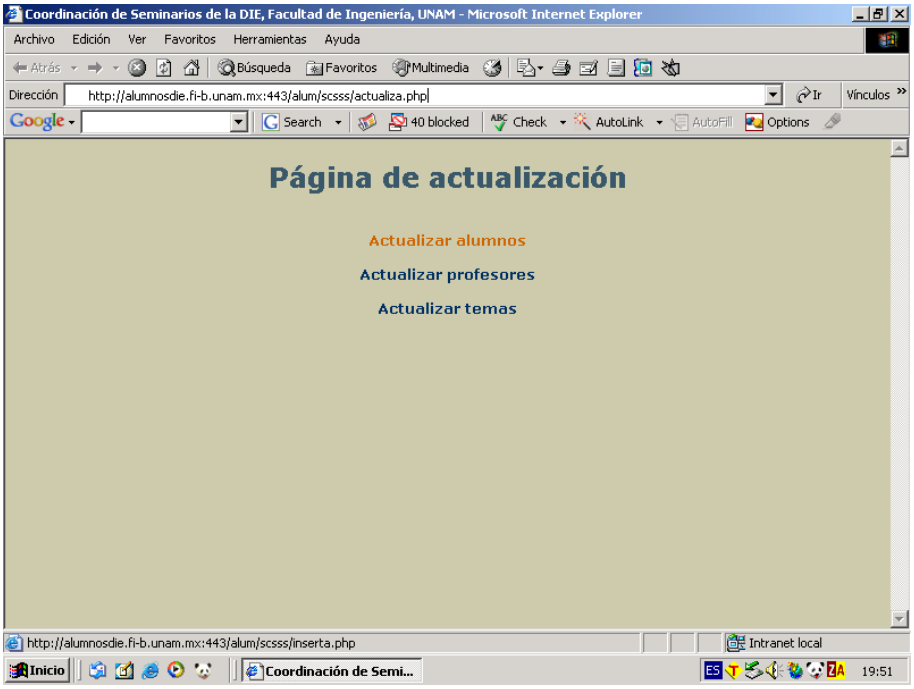
Por último aparecerá la pantalla siguiente en la que se dará clic en el botón “Close”, con la cual se habrá concluido este proceso.



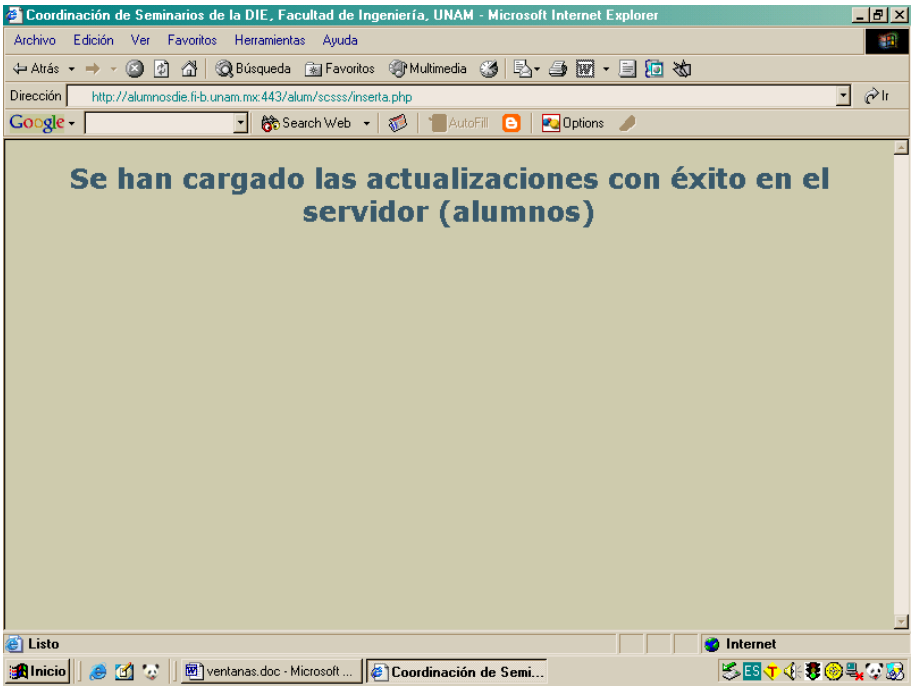
Al dar “Close”, aparecerá una sub-ventana preguntando que sí efectivamente queremos cerrar la ventana, por seguridad se deberá dar clic en “sí”, esto con el objetivo de que aseguremos que la sesión se ha cerrado y que no hay forma de que alguna persona ajena al sistema pueda hacer uso de la página de administración.

3. Actualización de la base de datos del servidor (descarga de los archivos xml en la base de datos).

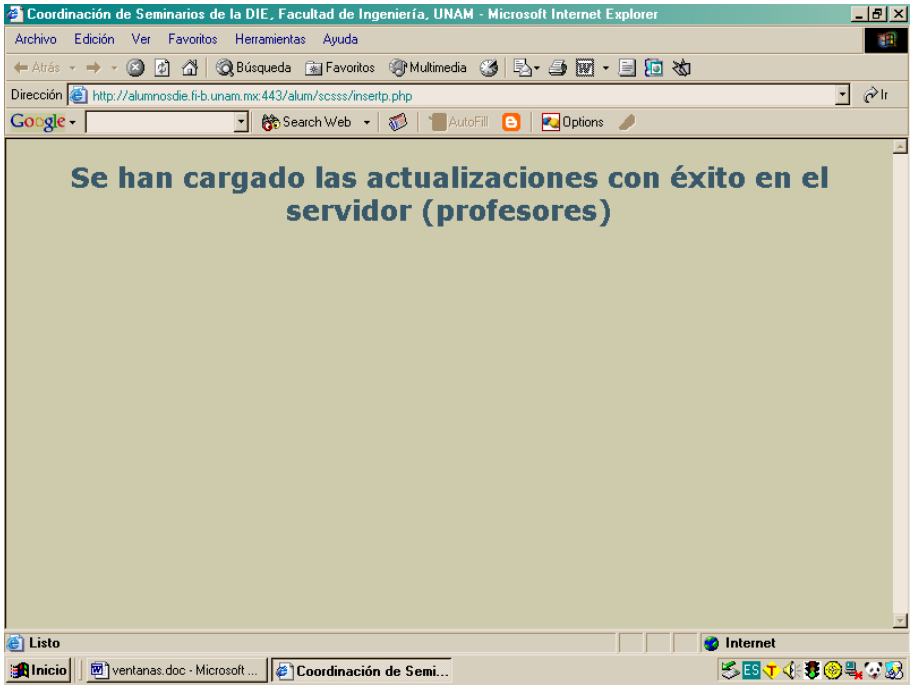
Una vez que hemos subido los 3 archivos xml al servidor, se procederá a descargarlos en la base de datos del servidor. Para esto ejecutaremos 3 scripts (uno por cada archivo xml) accediendo a la siguiente dirección: **http://alumnosdie.fi-b.unam.mx:443/alum/scsss/actualiza.php**, donde aparecerán tres links (uno por cada archivo xml) como se muestra en la siguiente ventana:

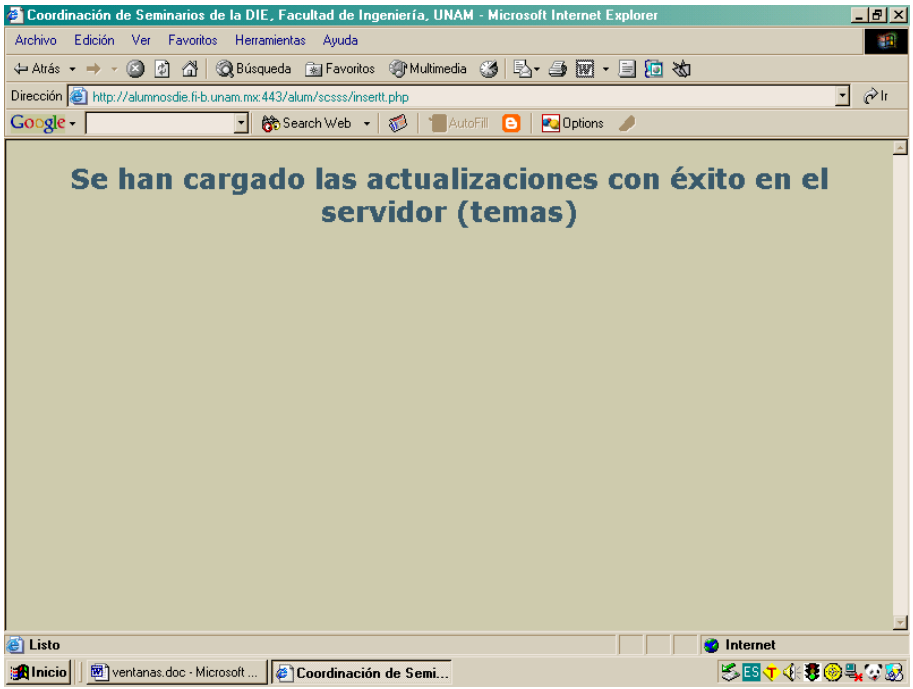


Haciendo clic en el link “Actualizar alumnos”, para el caso del archivo alumnos.xml, la ejecución habrá terminado una vez que aparezca en una nueva ventana la siguiente pantalla indicando que la actualización a la base de datos se ha llevado a cabo con éxito.



Se deberá seguir el mismo procedimiento para los archivos profesores.xml y temas.xml.





Glosario

ADO.- Del acrónimo en inglés Activex Data Objects. Es uno de los mecanismos que usan los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas.

Algoritmo.- Conjunto finito de instrucciones, operaciones o pasos no ambiguos que sirven para ejecutar una tarea o resolver un problema.

API.- Del acrónimo en inglés Application Programming Interface (Interfaz de Programación de Aplicaciones). Se refiere a una serie de funciones que están disponibles para realizar programas para un cierto entorno.

Array.- Conjunto de variables o registros del mismo tipo que puede estar almacenados en memoria principal o en memoria auxiliar.

Benchmark.- Anglicismo traducible al castellano como comparativa y utilizado en computación.

Bitmap.- Mapa de bits. Un tipo de imágenes para ordenador, en las que se almacena información sobre los puntos que las componen y el color de cada punto.

Bucle.- Serie de instrucciones que se repiten.

CGI.- Del acrónimo en inglés Common Gateway Interface (Interfaz Común de Pasarela). Interfaz de intercambio de datos estándar en WWW a través del cual se organiza el envío y recepción de datos entre visualizadores y programas residentes en servidores WWW.

Clase.- Las clases son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, se programa una clase.

Cliente.- Quien recibe el servicio. Programa o computadora que accede a recursos y servicios brindados por otro llamado servidor, generalmente en forma remota.

Cookie.- Fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas.

DOM.- Document Object Model (Modelo de Objetos de Documento). Es una forma de representar documentos estructurados (tales como una página web HTML o un documento XML) que es independiente de cualquier lenguaje orientado a objetos. Su finalidad es definir el conjunto de objetos que pueden componer documentos HTML

(páginas web) o XML, así como las estructuras que se definen dentro de él, sus propiedades y sus métodos, independientemente del lenguaje de programación utilizado, con el fin de evitar problemas de compatibilidad entre navegadores.

Feedback.- Retroalimentación.

Front-end.- En muchos casos, los terminos front-end y back-end se refieren al principio y final de un proceso. En diseño de software, el front-end es la parte del software que interactúa con el usuario y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en front-ends y back-ends es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end es el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas y procesarlas de una manera conforme a la especificación que el back-end pueda usar. La conexión del front-end y el back-end es un tipo de interfaz.

GPL.- Del acrónimo en inglés General Public License (Licencia Pública General). Es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.

Groupware.- Programa Informático Colaborativo. Se refiere a los programas informáticos que integran el trabajo en un sólo proyecto con muchos usuarios concurrentes que se encuentran en diversas estaciones de trabajo, conectadas a través de una red. Programas de aplicación que incrementan la cooperación y la productividad conjunta de pequeños grupos de colaboradores.

Hardware.- Conjunto de elementos materiales que componen una computadora.

Herencia.- Mecanismo de la programación orientada a objetos que permite derivar una clase de otra, de manera que extienda su funcionalidad.

HTTPS.- Versión segura del protocolo http.

HTTP.- Del acrónimo en inglés Hyper Text Transmission Protocol (Protocolo de Transmisión de Hipertexto). Es un protocolo usado para la transferencia de documentos WWW.

IA.- Inteligencia Artificial. Campo de la ciencia de la computación cuyo propósito es mejorar las computadoras para tratar de dotarlas con algunas características asociadas con la inteligencia humana, como la capacidad de entender el lenguaje natural y razonar bajo condiciones de incertidumbre.

Interfaz.- Conexión entre 2 dispositivos de hardware, entre 2 aplicaciones, o entre diferentes secciones de una red de computadoras. Porción de un programa que interactúa con el usuario.

Objeto.- Módulo de programa completo que contiene datos y procedimientos necesarios para que los datos sean útiles. Las características principales de un objeto son su identidad, su estado y su comportamiento. La primera permite distinguirlo de otros objetos y es fácilmente identificable gracias a los nombres que éstos poseen. El estado, por su parte, describe las propiedades (o datos) del objeto mientras que el comportamiento señaliza qué tipo de interacción puede comprender el objeto (sus métodos). También se puede considerar que los objetos son instancias de una clase en particular.

ODBC.- Del acrónimo en inglés Open DataBase Connectivity. Es un estándar de acceso a bases de datos desarrollado por Microsoft, el objetivo de ODBC es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué sistema gestor de bases de datos almacene los datos.

OLTP.- Del acrónimo en inglés OnLine Transaction Processing. Es un tipo de procesamiento de transacciones a través de una red de computadoras.

Paradigma.- Ejemplo que sirve de norma. Totalidad de ideas, percepciones y valores que constituyen una determinada visión de la realidad. Representan un enfoque particular o filosofía para la construcción del software.

Plataforma.- Se refiere al sistema operativo.

Portal.- Conjunto de páginas de Internet reunidas bajo una marca, dirección, tema, asunto o interés cuyo objetivo es ofrecer al usuario, de forma fácil e integrada, el acceso a una serie de recursos y de servicios, entre los que suelen encontrarse buscadores, foros, compra electrónica, etc. También se le conoce como sitio web.

Procedures.- Procedimientos. Serie de instrucciones almacenadas en una base de datos.

Proceso.- Programa en ejecución. Un proceso padre puede tener uno o más procesos hijos que llevan a cabo tareas adicionales.

Programa.- Secuencia de instrucciones que una computadora puede interpretar y ejecutar.

Red.- Conjunto de dos o más computadoras o dispositivos conectados entre sí y que comparten información.

RNA.- Redes Neuronales Artificiales. Paradigma de aprendizaje y procesamiento automático donde el objetivo es conseguir que las máquinas den respuestas similares a las que es capaz el cerebro que se caracterizan por su generalización y su robustez.

Rollback.- Operación que regresa la base de datos a algún estado previo (integridad de la base de datos).

RPC.- Del acrónimo en inglés Remote Procedure Call (Llamada a Procedimiento Remoto). Es un protocolo que permite a un programa de computadora ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

Sistema.- Conjunto de elementos interconectados que realizan funciones independientes a partir de datos de entrada y condiciones iniciales para generar información de salida.

Sistema experto.- Son aquellos programas que se realizan haciendo explícito el conocimiento en ellos. Los sistemas expertos trabajan con inteligencia artificial simbólica, es decir, a nivel de símbolos; como pueden ser ideas, imágenes, conceptos, etc.

Sistema operativo (SO).- Es un conjunto de programas o software destinado a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos.

SQL.- Del acrónimo en inglés Structured Query Language. Se refiere al lenguaje estándar de consulta a bases de datos.

TCP/IP.- Del acrónimo en inglés Transmission Control Protocol/Internet Protocol. Es el protocolo de comunicaciones estándar en Internet.

Teorema.- Afirmación que puede ser demostrada como verdadera dentro de un marco lógico.

Trigger.- Un trigger o disparador en una base de datos es un evento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

UDP.- Del acrónimo en inglés User Datagram Protocol. Protocolo de red.

URL.- Del acrónimo en inglés Uniform Resource Locator (Localizador Uniforme de Recurso). El URL es la cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet.

WebApp.- Aplicación web.

WYSIWYG.- Del acrónimo en inglés What You See is What You Get (Lo que ves es lo que tienes). Técnica que ofrece la reproducción exacta en pantalla de un texto tal como aparecería después en formato impreso.

World Wide Web.- La web o WWW, es un sistema de hipertexto que funciona sobre Internet.

Referencias

Bibliografía:

[1] Roger S. Pressman, 1998, *INGENIERÍA DEL SOFTWARE. Un enfoque práctico*, McGraw-Hill.

[2] Ibarra Obando Alma, Canales Domínguez Erick, Talavera Rosales Alejandro, 2000, *Guías y Textos de Cómputo. Páginas para Internet*, Cómputo Académico UNAM.

[3] Atkinson Leon, 1999, *Core PHP programming: using PHP to build dynamic Web sites*, Prentice-Hall.

[4] Gilmore W. J., 2001, *A programmer's introduction to PHP 4.0*, Apress.

[5] Lerdorf Rasmus, 2000, *PHP pocket reference*, O'Reilly.

[6] Medinets David, 2000, *PHP3 programming browser.based applications*, McGraw-Hill.

[7] Ratschiller Tobias, 2000, *Web application development with PHP 4.0*, New Riders.

[8] Welling Luke, 2001, *PHP and MySQL Web development*, Sams.

[9] Williams Hugh E., 2002, *Web database applications with PHP and MySQL*, O'Reilly.

[10] Bryan Ptaffenberger, 1999, *Diccionario de Términos de Computación*, Prentice Hall.

Mesografía:

Facultad de Ingeniería, UNAM:
http://www.ingenieria.unam.mx/

Programa de Apoyo a la Titulación (PAT):
http://www.mineria.unam.mx/pat.php

Instituto de Ingeniería:
http://pumas.iingen.unam.mx/infins/semina1.html

PHP (documentación y downloads):
http://www.php.net/

Desarrollo web:
http://www.desarrolloweb.com/

PosgrestSQL (documentación y downloads):
http://www.postgresql.org/

MySQL (documentación y downloads):
http://www.mysql.com/

La calidad y la ingeniería del software:
http://www.idg.es/computerworld/articulo.asp?id=7747

Arquitectura Cliente/Servidor:
http://sistemas.dgsca.unam.mx/publica/pdf/clienteservidor.PDF

Modelo Entidad-Relación:
http://www3.uji.es/~mmarques/f47/apun/node83.html

Modelo Relacional:
http://www.chochurro.com/archivos/000052.html

RDBMS (Relational Data Base Manager System):
http://es.wikipedia.org/wiki/Sistemas_Gestores_de_Bases_de_Datos

Manejadores de Bases de Datos PostgreSQL y MySQL:
http://www.netpecos.org/docs/mysql_postgres/

Diseño, usabilidad y arquitectura web:
http://www.proyectoweb.org/boletin/074-enero05.html
http://www.htmlweb.net/diseno/estructura/creacion_3.html

Visual Basic:
http://perso.wanadoo.es/tutoriales/cursos/vb/1.htm
http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/vbcn7/html/vbconintroductiontofilesystemobjectmodel.asp

Base de Datos ODBC:
http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/vccore/html/_core_what_data_sources_can_i_access_with_dao_and_odbc.3f.asp

Microsoft XML:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/xmmscXML.asp

Consulta de términos informáticos:
http://es.wikipedia.org

UML:
www.dcc.uchile.cl/~psalinas/uml/
www.creangel.com/uml/casouso.php
http://www.fi-b.unam.mx/pp/profesores/carlos/aydoo/toc.html
http://www.cs.ualberta.ca/~pfiguero/soo/metod/requerimientos.html#act1
http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x194.html