UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE UN MÓDULO DE EVALUACIÓN PARA PRACTICAS DE LABORATORIO USANDO EL MODELO DE DESARROLLO DEL CÓDIGO ABIERTO.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

ALBERTO ENRIQUE RASO CANO

DIRECTOR DE TESIS: ING. ALEJANDRO VELAZQUEZ MENA

MÉXICO, D.F. 2006

AGRADECIMIENTOS:

A Dios

Por mantenerme siempre de pie
y lograr ser alguien en la vida.

Gracias.

A mi facultad

Por permitirme conocer
la humildad de la grandeza
y el valor del conocimiento.

Gracias.

A mi director de tesis
Por su paciencia, por guiarme correctamente,
y llegar a la meta con su apoyo.

Gracias.

A mi madre

Por tu cariño, tu esfuerzo y tolerancia,
los consejos, las palabras y el amor.

Gracias.

A mi esposa

Por cerrar junto a mí esta etapa.
Mi compañera en mis desvelos y
El motivo de mis anhelos.
TE AMO

Gracias

A mi hermano

Por tu apoyo durante mi carrera
Y tus palabras de aliento.

Gracias.

ÍNDICE GENERAL

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

Resumen.

La evolución tecnológica marca la pauta en los métodos de aprendizaje que se desarrollan con éste. Sin embargo la tecnología es cara, y se encuentra mantenida como nicho de negocio. Se han implementado toda clase de modelos que permitan sacarle siempre el mejor provecho, ¡todo se vende!

En los años noventa, con la disponibilidad cada vez mayor de Internet, la filosofía del Open Source (Código Abierto) y el nacimiento de comunidades de personas muy afines a esta nueva forma de pensar, la generación de la era tecnológica, se expande y multiplica por toda la tierra.

Los paradigmas en el desarrollo tecnológico, particularmente en la electrónica y la computación, sufren cambios esenciales como resultado del trabajo de estas comunidades.

El enfoque del negocio también esta cambiando, las inversiones pasan de los productos desarrollados (propiedad intelectual) hacia el capital humano.

El presente trabajo es una muestra que representa la aplicación de los modelos de desarrollo consolidados, bajo la perspectiva de la filosofía de Código Abierto. En principio la metodología basada en los procesos nos permitirá la creación de un diseño original y su implementación, mientras que la liberación del código hará que el trabajo en éste mejore y perfeccione su calidad y utilidad, brindándole un espacio mas extenso de acuerdo a la utilidad que los interesados les quieran dar, que en este caso es académico y con la posibilidad de incorporación a un LMS (Learning Managment System). Porque si en algún lugar vale la pena invertir tiempo, esfuerzo o dinero es precisamente en el aprendizaje y la capacitación de las personas.

CAPÍTULO 1

Introducción

1.1.-Conceptos Básicos

1.1.1 Antecedentes de Educación a Distancia en la Universidad.

La Educación a Distancia no es una estrategia educativa nueva, sus orígenes se remontan al año 1840, cuando Sir Isaac Pitman inicia su sistema de enseñanza por correo para impartir cursos de estenografía por correspondencia en Gran Bretaña, dato con el que coinciden la mayoría de los autores.

A fines del siglo XIX instituciones particulares en Estados Unidos y en Europa ofrecían cursos por correspondencia dedicados a la enseñanza de temas y problemas vinculados a oficios de escaso valor académico. Es probable que este origen de la educación haya significado una apreciación desvalorizada de muchas de sus propuestas. El hecho de que, además, se transformase en una segunda oportunidad de estudio para algunas personas que fracasaron en una instancia juvenil, no evitó esa desvalorización, sino que le imprimió un nuevo sello. Transcurrieron varias décadas hasta que la educación a distancia se instaló en el mundo de los estudios como una modalidad competitiva frente a las ofertas de la educación presencial.

En 1892, la Universidad de Chicago estableció un curso por correspondencia, incorporando los estudios de la modalidad en la universidad. A principios del siglo XX, otras instituciones -por ejemplo, La Calvert, en Baltimore- desarrollaron cursos para la escuela primaria. En 1930 reconocemos treinta y nueve universidades norteamericanas que ofrecen cursos a distancia.

Sólo en la década de 1960, con la creación de universidades a distancia que competían con las de modalidad presencial, se lograron vencer muchos de los prejuicios de la educación a distancia. La Universidad de Wisconsin, creada para estudios a distancia, marca un punto importante en los desarrollos de esta modalidad en la educación norteamericana. La Universidad Abierta de Gran Bretaña, más conocida como Open University, mostró al mundo una propuesta de diseño complejo que, utilizando medios impresos, televisión y cursos

intensivos en períodos de receso de otras universidades convencionales, logró generar cursos académicos de calidad. Los egresados competían por los puestos de trabajo con los graduados de universidades presénciales.

Cuando se incorpora un nuevo paradigma de la audiencia tomada de los medios de educación masiva, particularmente la radio, el cine y la televisión, su uso sirvió para proveer educación a grandes grupos mediante difusión de mensajes, esto trajo como consecuencia la acuñación del término tele-educación, el cual tenía como característica la *unidireccionalidad* y el uso de un de medio a la vez. En la década de los años 70 aparece el paradigma de la enseñanza multimedia o modular que trae como innovaciones la combinación de medios impresos, video, audio laboratorio etc., en función de los objetivos instruccionales y la utilización de tutores locales en lugar de los tutores por correspondencia.

1.1.2 El Aporte de la Tecnología.

Los rápidos progresos de las nuevas tecnologías de la información y la comunicación seguirán modificando la forma de elaboración, adquisición y transmisión de los conocimientos. También es importante señalar que las nuevas tecnologías brindan posibilidades de renovar el contenido de los cursos y los métodos pedagógicos, y de ampliar el acceso a la educación superior. La nueva tecnología de la información no hace que los docentes dejen de ser indispensables, sino que modifica su papel en relación con el proceso de aprendizaje, y que el diálogo permanente que transforma la información en conocimiento y comprensión pasa a ser fundamental.

Desde el origen de la educación a distancia, las diferentes tecnologías incorporadas a la enseñanza contribuyeron a definir los soportes fundamentales de las propuestas. Libros, cartillas o guías, redactados especialmente fueron las propuestas iniciales; la televisión y la radio constituyeron los soportes de la década de 1970; los audios y videos, los de la década de 1980. En los años noventa, la incorporación de redes satelitales, el correo electrónico, la utilización de Internet y los programas especialmente diseñados para los soportes informáticos aparecen como los grandes desafíos de los programas en la modalidad.

1.1.3 Generaciones de la EAD.

La Educación a Distancia (EAD) es un sistema de enseñanza-aprendizaje que ocurre independientemente del tiempo y espacio. Para ello se requiere desarrollar estructuras educativas y establecer métodos que permitan el estudio y el diálogo, los cuales pueden estar basados en el uso de tecnología u otros medios tradicionales. La EAD no siempre fue concebida con un soporte tecnológico. Se pueden encontrar sus antecedentes en el estudio por correspondencia, en 1833 en Suecia fue registrado un curso de contabilidad donde se enviaba por correo el material impreso. No se contaba con soporte del profesor, por lo que se estudiaba de manera independiente. Esto es considerado la primera generación de la Educación a Distancia; en la segunda generación, el aprendizaje estaba combinado con la utilización de la radio y la televisión, donde se dictaban programas especiales. En general, este tipo de aprendizaje no incluía ningún tipo de retroalimentación o comunicación con otros participantes.

En la tercera generación, se incluyó la transmisión de televisión, casettes de videos, televisión por cable y satelital. En la última generación de Educación a Distancia incorporó el uso de las computadoras y la Web. Algunos dividen esta última generación en dos: una cuarta en la que se utilizan las redes de área local de forma aislada y una quinta que incluye las computadoras y redes con conexiones a Internet.

La Educación a Distancia se podía desarrollar como un programa de estudio por cuenta propia o podía tener interacciones entre el profesor y el estudiante. Este diálogo entre los participantes del proceso de aprendizaje podría ocurrir en el mismo tiempo (síncrono) o en otro momento (asíncrono). En cualquier caso, para generar esta comunicación, algunas tecnologías adicionales llamadas "*computer media communication*" (CMC) se deben aplicar para transformar la experiencia en un "aprendizaje colaborado".

Innovar en la educación es un tópico que se maneja con pretendida univocidad. Profesores, alumnos, directivos parecen estar de acuerdo en buscar, defender, propugnar y exigir innovaciones. El problema aparece en el momento de precisar en qué consiste innovar.

El desarrollo actual de la tecnología favorece la creación y el enriquecimiento de las propuestas en la educación a distancia, en tanto permite abordar de manera ágil numerosos tratamientos de temas, así como generar nuevas formas de encuentro entre docentes y alumnos, y de alumnos entre sí. Las modernas tecnologías resuelven el problema crucial de la educación a distancia, que es la interactividad.

1.1.4 Computer-Based Training (CBT)

Aquí se destina al enfoque tecnológico en la educación para el aprendizaje dentro del contexto de la informática educativa (Computer Based Training o CBT). La evolución tecnología, implica cambios y avances constantes y dinámicos aplicables al ámbito educativo y por ello deben operarse continuamente revisiones y ajustes. El uso de la tecnología implica para la educación inversiones que deben sustentarse en metodologías y criterios emanados de la realidad en estudio, mediante la estructuración de criterios representativos, es decir, cada vez se hace más necesario contar con información actualizada y confiable. CBT (Computer Based Training) se refiere a la capacitación a través de sistemas multimediales, generalmente entregados en CD-ROM.

1.1.5 Computer-Based Learning (CBL)

Computer Based Learning (CBL), se refiere al uso de la tecnología computacional como punto clave dentro del ambiente educacional. Mientras que se puede clasificar como el uso de las mismas dentro de un salón de clases, el término CBL se refiere más a un ambiente estructurado en el cual se utilizan para propósitos de enseñanza. Este concepto es generalmente visto como una forma diferente al uso de las computadoras donde el aprendizaje es al menos un elemento aparte de la experiencia, tal como, juegos o incluso el navegar en Internet.

1.1.6 Computer-Based Instruction (CBI)

Por lo general esta categoría de enseñaza es determinada por la presencia de un auxiliar o profesor que asiste a los estudiantes dentro de un salón o laboratorio. La

computadora se convierte en una herramienta que guía dentro de alguna práctica o tema concreto, generalmente a través de presentaciones gráficas, o representaciones de modelos que se asisten de la capacidad de cómputo del equipo.

El uso de software específico es instalado y provee lo necesario para cubrir los temas que se presentan. El uso de tutoriales facilita mucho el aprendizaje y brinda de mayor autonomía al estudiante frente a los problemas representados.

El contar con aulas equipadas que permitan representar problemas reales a través de modelos computacionales, es muy particular de este género. En las universidades por lo general se emplean en las practicas de ciencias (física, matemáticas, química, biología, etc...) así como en las áreas de economía y antropología social.

1.1.7 Internet-Based Training (IBT)

El desarrollo de la infraestructura de redes convierte a este particular en un nuevo actor frente al CBI, que aportando bajo la misma estrategia de trabajo, los elementos para lograr la capacitación del personal y de los estudiantes, consolida para las organizaciones grandes, un mismo material de enseñanza, asegurándose que esta sea de calidad homogénea en cualquier lugar que se presente.

Por poner un ejemplo, CISCO establece sus cursos sobre capacitación de redes y manejo de sus equipos con el mismo material que provee para los EE.UU. que para Europa, el material siempre se conserva actualizado y al alcance de los estudiantes.

Esta capacidad de concentración y administración del material y de los contenidos da pie para que se desarrollen los primeros sistemas de administración de aprendizaje (LMS) y en base a los IBT las universidades concentran los demás recursos que existen en los servicios de Internet para poder ofrecer esta concentración de recursos en un solo estándar con miras hacia la enseñanza estructurada.

1.1.8 E-Learning.

El e-learning es un término que se impuso como sinónimo de Educación a Distancia ó Educación en Línea. Para muchas personas es más cercano el e-learning que la EAD, sin embargo, este es un subconjunto, una técnica más para la aplicación de la Educación a Distancia. En este marco de cosas, el e-learning, es decir, la educación electrónica es una de las formas que la Educación a Distancia puede adoptar en la práctica, en tanto y en cuanto utilice recursos electrónicos para mediatizar los contenidos. Las facilidades con las que el desarrollo tecnológico dota a la formación no presencial se traducen de hecho en la preparación y oferta de cursos de la más diversa índole, la generación de plataformas en tal sentido y la oferta de avales mediante la creación de instituciones educativas propias. El e-learning es una herramienta que va más allá de que un estudiante curse una materia a través de Internet. El e-learning permite ofrecer información, capacitación y entrenamiento a todas aquellas personas que lo necesiten, en línea, en el momento y lugar más conveniente. El e-learning no es más que la utilización de Internet para revolucionar la manera en que la gente aprende.

1.1.9 Beneficios asociados con el E-Learning

Al eliminar barreras de tiempo, distancia, económicas y sociales, los individuos pueden tomar las propias riendas de su vida educativa. En la era de Internet, las destrezas y conocimientos tienen que ser actualizados constantemente. Las nuevas tecnologías, junto con el e-learning, ayudarán a las compañías de todos los tamaños, y a los países, a adaptarse a las demandas de la nueva economía. Al unificar todos los conceptos en versiones electrónicas, el empleado puede obtener la información que necesita en el momento que la necesita, a través de un sistema que registra el progreso de cada uno de los individuos, a la medida de sus propias posibilidades.

Con el e-learning, los desarrolladores de contenidos, los expertos y la comunidad de gente que busca aprender, están interconectados. Los empleadores pueden entregar contenido

en formas múltiples, administrar la experiencia de aprendizaje y crear una comunidad en red de gente que aprenda, desarrolladores de contenido y expertos. Quien recibe la educación puede practicar sus habilidades de manera individual o a través de equipos virtuales alrededor de áreas específicas de interés. El E-learning entrega un aprendizaje superior a costos reducidos; un mayor acceso al aprendizaje y un método de medición claro a todos los participantes en el proceso. En la actual cultura que marcha a pasos rápidos, las organizaciones que implantan procesos de e-learning le entregan a su fuerza de trabajo la habilidad de convertir el cambio en una ventaja competitiva. El e-learning cuenta con lo mejor de las soluciones de los dos mundos: e-learning efectivo que combina el probado método de enseñanza tradicional con los ricos recursos de la enseñanza basada en computadora para crear una solución de capacitación atractiva y motivadora.

Flexibilidad en línea: enseñar y retener la información crítica que necesita, cualquiera sea y donde sea que la necesite, sin tener que abandonar nunca la oficina o a las personas que lo necesitan.

Interactividad en el mundo real: motivando a los estudiantes a ponerse manos a la obra en los laboratorios de simulación, les permite probar sus habilidades en un entorno simulado perfecto, incrementando la probabilidad de que recuerden lo que aprendieron y puedan aplicarlo posteriormente en el trabajo.

Aprendizaje personalizado: a través de una prueba de pre-ingreso que mide el conocimiento y el nivel de habilidad, cada experiencia de aprendizaje es personalizada para asegurar que recibió sólo la información que necesita.

- *El perfil del profesional docente en e-Learning.*

La figura “docente” en e-Learning sufre importantes modificaciones, y en parte está todavía por definir o, al menos, por consensuar.

Sin embargo, el desarrollo de contenidos académicos y científicos específicos para un contexto de formación en red es uno de los ámbitos profesionales emergentes entre los nuevos perfiles docentes.

La figura del *Gestor Académico de e-Learning* no interviene de manera directa en el proceso de enseñanza-aprendizaje una vez iniciado, sin embargo es fundamental que, detrás de toda experiencia formativa, exista una planificación eficaz de todos los elementos que se implicarán en el proceso formativo. Además, en cuanto responsable de la iniciativa, el Gestor ha de coordinar todos los elementos que toman parte en el proceso, tanto para lograr el mayor éxito de la intervención como para evaluar y mejorar constantemente la calidad de las iniciativas de e-Learning bajo su responsabilidad.

1.2 Software Libre y Software Comercial

1.2.1 Software Libre

Software libre (en inglés *free software*) es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del costo de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. Análogamente, el software gratis o gratuito (denominado usualmente Freeware) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de software no es libre en el mismo sentido que el software libre, al menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

No se debe confundir software libre con software de dominio público. Este último es aquel por el que no es necesario solicitar ninguna licencia y cuyos derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software es aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es dominio público. En resumen, el software de dominio público es la pura definición de la libertad de usufructo de una propiedad intelectual que tiene la humanidad porque así lo ha decidido su autor o la ley tras un plazo contado desde la muerte de éste, habitualmente 70 años.

Ventajas e inconvenientes del Software Gratuito

El Software gratuito muchas veces es suministrado de forma complementaria a algún otro programa, muchas veces comercial, por ejemplo las bibliotecas de entorno de java, se ponen a disposición de los usuarios sin costo alguno, pero el código fuente nunca es revelado. Este tipo de estrategias que suelen usar las casas comerciales persiguen muchas veces un fin

mas allá del mero software, que se descarga (como el ejemplo de los visores de java, flash, etc...), y se basa en vender el software de producción.

. En definitiva, el software gratuito es una excelente idea como una opción complementaria de otro tipo de software. Pero el objetivo del software gratuito sólo y exclusivamente para no tener que pagar nada por algo que cuesta hacer tampoco tiene sentido. A menos que se este pensando que el Estado se encargue de ello (es decir, que se financie con los impuestos de todos). En este sentido, la historia ha demostrado que ésta no es precisamente la mejor opción para la creatividad, innovación y desarrollo, por ello se debe la coexistencia de distintos tipos de software (libre, gratuito, abierto y comercial). Se deben potenciar todos, y que la competencia y el mercado hagan su tarea de seleccionar lo mejor, dentro de unas reglas de coexistencia.

1.2.2 Software Comercial

Gracias al modelo de Software comercial la gente puede construir negocios basándose en sus propias ideas y novedades. Sus 30 años de existencia avalan la vitalidad de este modelo en todo el mundo como generador de novedades y posibilidades económicas. Este modelo nace de un sistema creado por el propio sector TI junto a gobiernos de todo el mundo. Los líderes del sector y los administrativos han puesto en marcha una de las épocas más productivas en avances tecnológicos al elaborar, de una forma eficaz, un sistema que promueve la transparencia y la colaboración tecnológica al tiempo que premia a las empresas que crean productos innovadores.

El sector del software comercial ha transformado el mundo empresarial al ofrecer productos que pueden hacer casi cada tarea del negocio de una manera más rápida, inteligente y con mejores resultados En algunos casos, estas ideas se diseñaron primero a través de métodos de desarrollo de código fuente abierto. Como estas ideas recibieron licencias gracias a las permisivas licencias de código fuente abierto, las empresas pudieron conocer esas ideas y mejorarlas y finalmente lanzar productos mejorados al mercado.

Los beneficios aportados por el sector del software comercial surgen mayoritariamente debidos a que el sector TI ha adoptado nuevos modelos sostenibles de negocio y a que las administraciones públicas adoptaron las medidas necesarias para promocionar un mercado de trabajos digitales. La consecuencia ha sido un sector dinámico conocido por sus impresionantes innovaciones y su notable colaboración técnica.

Ventajas e inconvenientes del Software Comercial

El software comercial es típicamente cerrado es decir, que el código no está disponible y, como su nombre indica, no es gratuito. Además suele ser licenciado, o sea que se le da a uno permiso de usarlo, después de que hace el pago correspondiente; en el caso de las empresas y negocios, necesitan hacer aportaciones anuales, pues el licenciamiento tiene fecha de caducidad y para no cometer un delito, es necesario renovar dicho licenciamiento, todas esas características pueden ser inconvenientes en este tipo de programas; sobre todo si el precio es elevado. No obstante, existe el llamado software de prueba (“tryware” o “shareware”) y las versiones de demostración (“demo”), limitadas de alguna forma o totalmente funcionales durante un periodo de tiempo dado. El objetivo es que se pueda probar la aplicación antes de decidirse a comprarla.

Probablemente la mayor ventaja del software comercial es que en general tiene calidad, o eso se supone, si bien existen casos que demuestran que un software comercial puede no ser lo suficientemente bueno para determinado uso.

Por último, la coexistencia de los tres modelos es interesante y puede facilitar el desarrollo de nuevos productos de calidad. Así, por ejemplo, un programa de código abierto puede resultar conveniente cuando interese modificarlo y adaptarlo a unas necesidades determinadas, como sucede con el sistema operativo Linux. Esto es particularmente relevante en grandes empresas, instituciones o en la Administración regional o estatal. En otros casos lo ideal puede ser un programa gratuito, porque no existan recursos para adquirirlo, como puede suceder en una escuela o en países en desarrollo. Pero en otros casos lo ideal puede ser un programa o sistema operativo comercial, porque ofrezca un valor añadido extraordinario.

1.2.3 Software Abierto (Open Source)

El software abierto es aquel en el que contamos con el código, independientemente si pagamos o no por él, así entonces el software gratuito plantea como una ventaja importante ser, desde luego gratis, sin embargo, la desventaja principal es que no cuenta con soporte alguno del desarrollador (persona o empresa) que lo ofrece, también es valioso mencionar que el software gratis y de código abierto (como Linux) nos da la posibilidad de adaptar o corregir el problema que se nos presenta.

La principal ventaja del software abierto es que su código fuente es público, y por tanto puede ser conocido y modificado por cualquiera, pero ello no implica necesariamente que se trate de software gratuito.

El mayor inconveniente del software abierto puede ser que, si no existe una fuente directa de financiación por la venta del mismo, se resientan aspectos de control de calidad, interfaz intuitiva y desarrollo. Por eso en algunos casos el software abierto suele adornarse con alguna ventaja añadida, siendo entonces vendido a un determinado precio, que no suele ser elevado, alguien habrá tenido que pagar, aunque sea con su tiempo, el esfuerzo de desarrollar algo, y no se puede ser egoísta, pretendiendo que los programadores trabajen para los demás y aporten siempre el software gratis.

Así que, el software abierto puede ser financiado por instituciones públicas como las universidades (aunque en última instancia se estará pagando con los impuestos), particulares o entidades privadas, puede y debe tener un espacio en el mundo del software, pero probablemente lo mejor es que exista junto a otras alternativas.

1.2.4 Licenciamiento en el Software Libre

Inicialmente, las computadoras eran herramientas que servían para procesar datos, y los programadores se ayudaban entre sí compartiendo el código que escribían, sin embargo, poco a poco las empresas decidieron convertir los programas informáticos en un producto comercial y prohibir su libre copia y modificación, lo que llevó al desmembramiento de la comunidad de programadores.

Origen de la Fundación para el Software Libre.

Richard Matthew Stallman, del Laboratorio de Inteligencia Artificial del MIT (Massachusetts Institute of Technology), luchó durante varios años contra la disolución de su comunidad, pero finalmente se quedó solo. Entonces se planteó crear una nueva comunidad, en la que compartir y ayudar a los demás no fuera ilegal. Para ello decidió escribir un nuevo sistema operativo completo, compatible con Unix (un potente sistema) pero libre para todos. Bautizó a su proyecto como GNU (GNU is Not Unix).

En 1985 publicó el "Manifiesto GNU", que define y explica sus objetivos y motivaciones, y poco tiempo después fundó la organización sin ánimo de lucro Free Software Foundation (Fundación para el Software Libre: http://www.fsf.org) para coordinar el proyecto, al que poco a poco se iba uniendo más gente. La influencia de Stallman ha sido esencial para establecer el marco de referencia moral, político y legal del movimiento del software libre como alternativa al desarrollo y distribución de software privativo. Un mérito tan importante o más que sus impresionantes logros como programador fue el inventar el concepto de copyleft (izquierdos de autor), que implementó en la Licencia Pública General de GNU (conocida generalmente como la GPL).

Hacia 1990 el sistema GNU estaba casi completo; el único componente esencial que faltaba era lo que se llama kernel o núcleo, al que denominaron Hurd. La Free Software Foundation decidió escribirlo siguiendo un diseño tan innovador como complejo. A día de hoy, el Hurd es funcional, pero todavía le faltan varios años para alcanzar la madurez

necesaria. Afortunadamente, no ha hecho falta esperar a la publicación del Hurd para poder disfrutar de un sistema completamente libre, gracias a la aparición de Linux.

GNU/Linux

Al ser el código de Unix secreto, los estudiantes de informática tenían difícil estudiarlo y aprender cómo se escribía un sistema operativo. Ante esta situación, el profesor Andrew Stuart Tanembaum escribió un sistema operativo de tipo Unix llamado Minix, y un libro en el que explicaba todos sus secretos. Al ser de carácter pedagógico, el sistema era deliberadamente sencillo y con pocas funcionalidades. En 1991, el estudiante finlandés Linus Benedict Torvalds decidió aplicar lo aprendido y escribir un nuevo núcleo que superase las limitaciones de Minix. Lo hizo por mera diversión, y aprovechando las herramientas del proyecto GNU. Sin embargo, la verdadera genialidad de Linus fue que, aunque en principio no pasaba de ser un entretenimiento privado, decidió enviar un mensaje a Internet informando de su proyecto (que se llamó Linux), poniéndolo a disposición de quien quisiera jugar con él, y solicitando la ayuda de todo el que quisiera colaborar. Lo revolucionario de Linux no está en su diseño (que no es especialmente innovador) ni en su filosofía (que la Free Software Foundation llevaba años predicando), sino en su metodología. Efectivamente, hasta entonces el software se escribía en grupos cerrados y de carácter vertical, mientras que Linus inauguró un nuevo modelo, distribuido y muy abierto, en el que cualquiera podía participar. A estos métodos tan diferentes se les ha denominado modelo catedral y modelo bazar, respectivamente.

Linus no tardó en adoptar la licencia GPL, y al unir su núcleo con las herramientas del proyecto GNU, se obtuvo finalmente un sistema operativo funcional totalmente libre, que se conoce como GNU/Linux y que hoy usan millones de personas en todo el mundo.

Desde entonces, el software libre no ha dejado de crecer y multiplicarse, y el modelo bazar ha demostrado ser más eficiente y producir programas de más calidad. Surgió también un nuevo movimiento, que no defendía este tipo de software por motivos éticos, sino

únicamente por la superioridad técnica de su modelo. Este movimiento y el software creado con esta perspectiva se denomina Open Source o código abierto. En la práctica, el software libre y el de código abierto son lo mismo, diferenciándose únicamente en su filosofía.

No tardaron en surgir personas que empezaron a empaquetar en un conjunto de disquetes el núcleo, los programas de GNU y diverso software libre de otras partes, con lo que se facilitaba mucho su instalación. A estos conjuntos de disquetes (hoy CD o DVD) se les denomina distribuciones de GNU/Linux. Algunas están desarrolladas por empresas, como Red Hat o SuSE, mientras que otras las preparan organizaciones de voluntarios, como Debian o Gentoo.

1.2.5 Herramientas de Software (libre y código abierto)

El software libre es un fenómeno que está llamado a revolucionar modelos de negocio de la industria del software. Los criterios básicos que deben cumplir las herramientas son:

- ofrecer el código fuente de la aplicación.
- distribuirse bajo alguna de las licencias consideradas de referencia.
- poder ser modificadas, copiadas y distribuidas libremente, respetando los términos establecidos en la licencia respectiva.

Una revisión somera de las herramientas para gestión de contenidos que son distribuidas bajo alguna de las licencias consideradas para software libre muestra la existencia de un gran número que ofrecen variadas prestaciones y orientaciones. Brevemente, su arquitectura técnica se fundamenta en la tercia *servidor web*, *intérprete de lenguaje de programación y gestor de base de datos*. A este esquema responde el conocido acrónimo LAMP (Linux, Apache, MySQL, PHP o Perl), o su versión Windows, WAMP. Precisamente han sido PHP, Perl y MySQL las herramientas más extendidas entre los sistemas libres para gestión de contenidos, ya que la mayoría de las soluciones se basan en ellos.

Plataformas para desarrollo de gestión de contenidos.

Se trata de soluciones que ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Ofrecen un entorno y unas herramientas de desarrollo. En consecuencia, su interés radica en la posibilidad de construir soluciones adaptadas a cada caso.

- *Servidor de aplicaciones*

Un servidor de aplicaciones es una computadora servidor en una red de computadoras, dedicado a ejecutar ciertas aplicaciones de software. El término también hace referencia al software instalado en tal computadora para facilitar la ejecución de otras aplicaciones.

  - Typo3
  - Zope.
  - Midgard Project.
  - OpenCMS.

- *Portales*

Un portal de Internet es un sitio web 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..., es un sitio que recibe un alto tráfico y que está dirigido a resolver necesidades específicas de un grupo de personas. Los portales normalmente tienen programación que requiere muchos recursos computacionales y por su alto tráfico generalmente se hospedan en servidores dedicados. El portal es considerado un intermediario de información que tiene como fuente de ingreso la publicidad de empresas que ahí se anuncian. La creación y mantenimiento de portales, como soporte y herramienta básica de los servicios de información web, es una de las funciones principales que desempeñan los sistemas de gestión de contenidos para portales. Su funcionalidad, administración y

mecanismos de control están especialmente orientados a ofrecer a sus usuarios un portal con diferentes tipos de contenidos y de servicios, Los más extendidos son:

- PHP Nuke
- Drupal
- Mambo
- Plone (requiere Zope)

- *Aula Virtual*

Entornos que ofrecen las prestaciones necesarias para crear contenidos para aprendizaje en línea, y ciertos mecanismos de interacción, como foros, chats, evaluación interactiva, etc, Las plataformas más conocidas son:

- Claroline
- Moodle
- Atutor

- *Bibliotecas Digitales.*

El paradigma para los servicios de información de finales del siglo XX y comienzos del siglo XXI, organizadas alrededor del grupo que forman los usuarios, las colecciones, y los servicios de valor añadido, se configuran como un espacio altamente especializado para la gestión de contenidos. Este tipo de herramientas es más exigente, en sus requerimientos, que los otros tipos anteriores.

- Dspace
- Greenstone

- *Publicaciones Digitales*

Publicaciones digitales: son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc.

- Cofax

- Open Journal Systems
- ePrints

- *Blogs o Bitácoras*

Los blogs son el fenómeno, ya consolidado, que ha dado un verdadero potencial democrático a la web, a pesar de los inconvenientes que pueda conllevar. Los blogs muestran un modelo de gestión de contenidos bastante simplificado, ya que suelen ser monousuario, y con un sencillo flujo de trabajo, lo que ha facilitado su expansión entre amplios grupos de usuarios sin conocimientos técnicos profundos.

- WordPress.
- MediaWiki.
- Jaws.

- *Herramientas de programación*

Las herramientas de programación para plataformas libres también compiten con sus contrapartes comerciales, resultando de igual calidad, dentro de las cuales existen compiladores (C, C++, Fortran, Pascal, Basic,…), modeladores UML (gaphor), depuradores (GDB, Valgrind, etc…) y podemos seguir así, consultando todo el amplio repositorio que existe en Sourceforge.net.

Podemos concluir que existen vastas herramientas para la creatividad y desarrollo de software en el mundo del Open Source del que se puede echar mano para crear aplicaciones de calidad, montar servidores de aplicaciones, portales y un sinfín de soluciones que se ajustarán a las necesidades de los individuos, grupos o empresas.

1.3 Estándares en la ingeniería de software

1.3.1 Modelos de estimación de costos

Modelo COCOMO (1981), COCOMO II(2000).

COnstructive COst MOdel, es un modelo que permite realizar estimaciones y planificaciones de proyectos de sistemas de información. Fue desarrollado por **B. W. Boehm** a finales de los 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "*Software Engineering Economics*"[1]. COCOMO es una jerarquía de modelos de estimación de costos de software que incluye submodelos básico, intermedio y detallado.

Cocomo puede ser aplicado a tres tipos de proyectos de software. Esto nos da una impresión general del proyecto.

- Proyectos Orgánicos – Son relativamente pequeños, con proyectos software sencillos en los que el equipo tiene mucha experiencia y tienen pocos requisitos estrictos.

- Proyectos Medios – son intermedios (en tamaño y complejidad) Proyecto software en los que no tienen la misma experiencia todos los miembros del equipo. Hay requisitos más y menos rígidos.
- Proyectos embebidos – Son proyectos software que se deben desarrollar con unos requisitos hardware, software y de operación.

Modo Básico

La ecuación de COCOMO en este modo básico es:

$E=a_i S^b im(X)$

[1] Boehm, B.W. "Software Engineering Economics" (Prentice-Hall, 1981) pp 23,24,25

- **S** el número de miles de líneas de código fuente
- **m(X)** es un multiplicador que depende de 15 atributos

en la siguiente tabla se muestran los coeficientes para los diferentes modos

Tipo de Proyecto	básico		Intermedio	
	a_i	b_i	a_i	bi
Orgánico	2.4	1.05	3.2	1.05
Medio	3.0	1.12	3.0	1.12
Embebido	3.6	1.20	2.8	1.20

Tabla 1.- Coeficientes ecuación COCOMO

Modo orgánico.

En este modo, un pequeño grupo de programadores experimentados desarrollan software en un entorno "familiar." El tamaño del software varía de unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles de líneas (medio), mientras que en los otros dos modos el tamaño varía de pequeño a muy grandes (varios cientos de miles de líneas). En este modo, al igual que en los otros, el costo se incrementa a medida que el tamaño lo hace, y el tiempo de desarrollo se alarga.

Se utilizan dos ecuaciones para determinar el esfuerzo de personal y el tiempo de desarrollo. El costo es:

$K_m=2.4\ S_k 1.05$

donde K_m se expresa en personas-mes y S_k es el tamaño expresado en miles de líneas de código fuente. El tiempo de desarrollo se da por:

$t_d=2.5\ K_m 0.38$

donde K_m se obtiene de la ecuación anterior y t_d es el tiempo de desarrollo en meses. Estas ecuaciones se han obtenido por medio de ajustes de curvas realizado por **Boehm** en TRW sobre 63 proyectos.

Modo Embebido.

En este modo, el proyecto tiene unas fuertes restricciones, que pueden estar relacionadas con el procesador y el interfase hardware. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Las estimaciones de tiempo y costo se basan en las mismas ecuaciones que en el modo orgánico, pero con diferentes constantes. Así, el coste se da por

$K_m = 3.6\ S_k 1.20$

y el tiempo de desarrollo por
$t_d = 2.5\ K_m 0.32$

Modo Semiencajado.

Es un modo intermedio entre los dos anteriores. Dependiendo del problema, el grupo puede incluir una mezcla de personas experimentadas y no experimentadas.

Las ecuaciones son:
$K_m = 3.0\ S_k 1.12$

y el tiempo de desarrollo por
$t_d = 2.5\ K_m 0.35.$

COCOMO básico es un forma rápida y sencilla de estimar la magnitud de los costos de un proyecto software. Podemos observar que a medida que aumenta la complejidad del

proyecto, las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que usar con cuidado el modelo básico, pues este alcance está necesariamente limitado porque hay muchos factores sin contabilizar, como son las diferencias de requisitos hardware, la calidad y experiencia del personal, utilización de técnicas y herramientas más sofisticadas, y otra serie de atributos conocidos que tiene mucha influencia en los costes de un proyecto.

Modo Intermedio

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

Modelo SLIM (Software Life Cycle Management)

SLIM es uno de los primeros algoritmos de modelo de costos para el desarrollo de software. Está basado en la función de Norden/Rayleigh y es conocido generalmente como un macro modelo de estimación (para proyectos grandes). SLIM habilita un estimador de costo basado en las siguientes funciones:

Calibración: Es un ajuste minucioso del modelo que representa el desarrollo de una aplicación de software a través de la interpretación de la base de datos de desarrollo de proyectos anteriores.

Construcción: Un modelo de información de la aplicación de software, recabando datos sobre sus características funcionales de la aplicación, los atributos del personal, de las computadoras, etc...

Tamaño del software: SLIM usa una versión de 'Técnicas de costo" automatizada de líneas de código (SLOC)

La cantidad de trabajo que se encuentra en cualquier producto se puede ver como el producto del esfuerzo realizado en un periodo de tiempo, y se puede escribir como:

Producto = (Constante) x Esfuerzo x Tiempo

Donde:
Producto: representa cierta medida sobre la funcionalidad del mismo, y se cree proporcional al producto **Esfuerzo x Tiempo**.

La medida SLOC (Counting Source Lines of Code) suele ser una medida habitual de la funcionalidad.

Esfuerzo: representa el trabajo humano, medido en personas-mes o personas-año.

Tiempo: representa la duración del trabajo, medido en meses o años.

La Constante es un factor de proporcionalidad. Una vez establecidas las otras tres variables, esta constante permite igualarlos. Sin embargo parece que la cantidad de producto depende también de "*cómo se hacen las cosas*", puesto que con el mismo esfuerzo y tiempo, y dependiendo del entorno de trabajo, podremos conseguir mayor o menor cantidad de producto.

Productividad del Proceso.

Mayor sentido tiene la ecuación anterior si se expresa como:
Producto = Productividad x Esfuerzo x Tiempo

De donde podemos obtener el valor de **Productividad**, si conocemos los otros términos, como:

Productividad = Producto/(Esfuerzo·x Tiempo).

Aunque de esta manera la Productividad no está definida precisamente, se supone que incluye un conjunto de factores que afectan a toda la organización, incluyendo:

- la gestión del proyecto.
- la utilización de buenos requerimientos
- diseños, codificaciones, inspecciones y pruebas
- el nivel del lenguaje de programación
- el estado de la tecnología
- la experiencia de los miembros del grupo
- la complejidad de la aplicación.

En un determinado momento del proyecto, el valor de la Productividad es fijo. Sin embargo, la evolución en el tiempo y la acción de la dirección puede hacer cambiar su valor. Esta productividad es un conjunto de factores y se puede denominar *"parámetro de productividad del proceso"*.

1.3.2 Modelos de evaluación de procesos

Los métodos los podemos diferenciar entre métodos ágiles y métodos pesados, actualmente podemos considerar que los procesos de desarrollo de software están establecidos de ésta manera. Los primeros aportan rapidez y flexibilidad al proceso de desarrollo pues nos permiten obtener resultados en las primeras fases de desarrollo del proyecto. Los segundos se toman más tiempo y se genera mucha documentación, que dependiendo de la naturaleza de proyecto, podrían ser relevantes o no.

Método en cascada

En muchos años este fue el método más empleado, es un clásico de la ingeniería de software, y en muchos casos los profesionistas lo destacan como el único proceso aplicable. En la bibliografía referente a la Ingeniería de Software este proceso es continuamente mencionado, de hecho en las universidades es el método de desarrollo inicial y a veces perdura durante la instrucción de los alumnos a lo largo de la carrera. Al día de hoy este método se sigue utilizando, sobre todo cuando los proyectos suelen tener una proyección de un año o más en su desarrollo y con mucha gente involucrada. Está definitivamente considerado entre los métodos pesados.

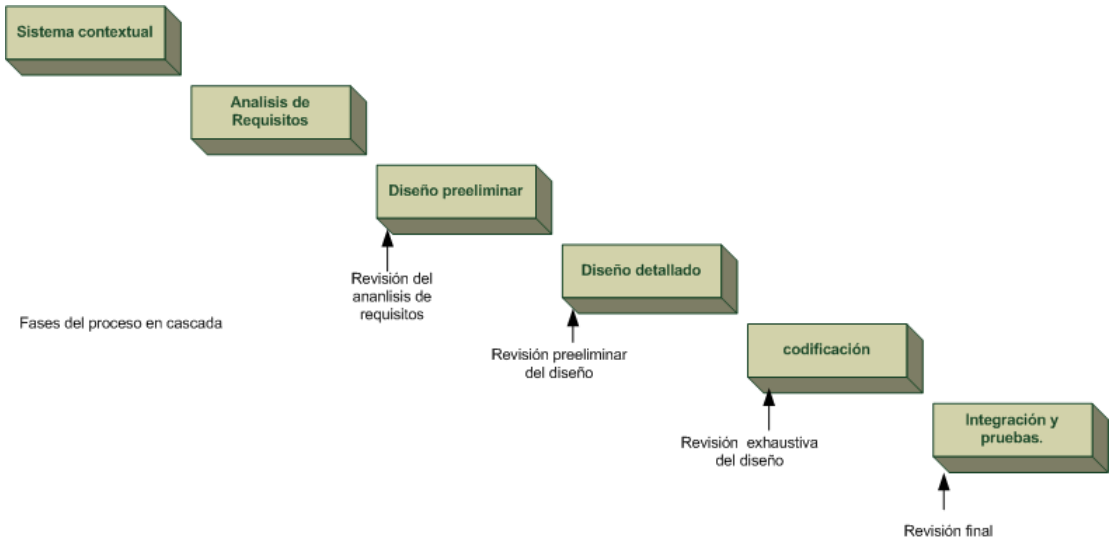


Ilustración 1.- Ciclo de vida del modelo en cascada

Básicamente, el método en cascada consiste en dividir un proyecto en fases, actividades, tareas y procesos jerarquizados. De entrada se divide el proceso en fases. Hasta que no se completa una fase, no se puede pasar al la otra, y en cada una de ellas se realizan todos los controles necesarios para asegurar la calidad del resultado final. Además esta revisión nos puede servir para tomar determinadas decisiones como reacción anta las eventualidades que pudieran surgir, como por ejemplo el cambio de un determinado requisito funcional de parte del cliente.

Las fases que suelen utilizarse para dividir un proyecto a utilizar esta metodología son las siguientes:

- Sistema Contextual.
- Análisis de requisitos.
- Diseño preliminar.
- Diseño al detalle.
- Codificación y pruebas Unitarias.
- Integración y pruebas del sistema.

RUP (Rational Unified Process).

Esta es también una metodología pesada, con el propósito de adaptación en mente, para cualquier tipo de proyecto. En sus inicios fue desarrollada por la empresa Rational, que hoy pertenece a IBM. ¿Como funciona? divide el proyecto en fases independientes, en las cuales se ejecuta un flujo de trabajo iterativamente de acuerdo con los diagramas de actividades a realizar, en resumen las actividades de RUP son las siguientes:

- Puesta en marcha.
- Definición Inicial, análisis y diseño preliminar.
- Implementación (diseño detallado y codificación).
- Puesta en producción.

RUP utiliza el lenguaje de modelado UML para plasmar los requisitos del proyecto y además propone la realización de vasta y detallada documentación. Como RUP es un proceso muy amplio y flexible, se recomienda adaptarlo a cada proyecto en particular. Algunos equipos de desarrollo suelen utilizar una versión reducida y personalizada de este proceso en sus proyectos.

Entre sus características principales se cuentan:
Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)

- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

CMMI (Capability Maturity Model Integration).

Originalmente este Método se llamó tan solo CMM pero desde el año 2000 de cambio a su actual denominación. En español se traduce como "Modelo de capacidad de Madurez". Esta metodología fue introducida por el SEI (Software Engineering Institute) y su objetivo es valuar la madurez de un proyecto en base a la capacidad de sus procesos.

El CMMI se puede utilizar de dos formas:

1 Por etapas: La valoración de la madurez se realiza por niveles de la capacidad de procesos.
2 Continua: Dicha valoración se realiza según el proyecto.

Esta metodología contiene los procesos de un proyecto en diferentes áreas:

- Toma de requisitos.
- Gestión de requisitos.
- Solución técnica.
- Integración del producto.
- Verificación.
- Validación.

El CMMI fue ideado para aplicar en cualquier área no solo de Ingeniería de Software, incluso puede utilizarse para mejorar la eficiencia de la organización de una empresa.

La integración del Modelo CMM provee de una oportunidad para evitar o eliminar las barreras que eliminan los modelos integrados que trascienden disciplinas. La integración del CMM consiste en la mejor práctica dirigida al desarrollo y mantenimiento del producto, esta se refiere a la cobertura del ciclo de vida del producto desde su concepción hasta su entrega y mantenimiento. Hay un énfasis en ambos sistemas de ingeniería e ingeniería en software así como la integración necesaria para construir y mantener el producto íntegro. EL CMM se

enfoca en mejorar el procedimiento en una organización, contiene los elementos esenciales del proceso efectivo para una o mas disciplinas y describe el camino a seguir para un revolucionario mejoramiento llevándolo desde su proceso de inmadurez con una disciplina hacia el proceso de madurez con una calidad efectiva de mejoramiento.

Mark Paulk y otros en el Instituto de Ingeniería de Software crearon el primer Modelo CMM de la organización de Software y la publicaron en un libro que nombraron "El Modelo de Capacidad de Madurez. El proyecto de Integración del Modelo de Capacidad de Madurez fue formado para separar el problema de uso múltiple. El resultado de la misión de trabajar en equipo fue la combinación de 3 modelos.

- El Modelo de Capacidad de Madurez de Software (SW-CMM) v2.0
- El Modelo de Sistema de Capacidad de Ingeniería (SECM[2]).
- La Integración de Productos desarrollados con el Modelo de Capacidad de Madurez (IPD-CMM[3]) v098

Estos 3 modelos fueron seleccionados por su adaptación en software y en sistemas de ingenierías, así mismo por sus acercamientos a los diferentes procesos de mejoramiento en una organización.

Usando información de estos modelos tan populares y considerados como una fuente de material. Como resultado, el equipo de la integración del Modelo de Capacidad de Madurez creó un cohesivo conjunto de modelos integrados que pueden ser adoptados tanto por los modelos actuales como por los nuevos. La Integración del Modelo de Capacidad de Madurez es resultado de la evolución del SW-CMM, el SECM y el IPD-CMM.

El intento de la integración del CMM es proveer de un modelo que cubra el desarrollo y mantenimiento de productos y servicios pero también que provea de un extenso armazón al cual se puedan adherir los nuevos cuerpos de conocimiento, actualmente hay 4 cuerpos de conocimiento y están disponibles para el mejoramiento del proceso de plantación usando CMMI:

[2] Systems Engineering Capability Assessment Model
[3] Integrated Product Development Capability Maturity Model

- Sistema de Ingeniería.
- Ingeniería en Software
- Integración de productos y desarrollos del proceso.
- Suministrados Viable

La definición del CMM permite a la comunidad desarrollar modelos con diferentes características, tanto como un modelo contenga los elementos esenciales de un proceso efectivo para una o mas disciplinas y describa un camino hacia un mejoramiento desde su proceso de inmadurez con una disciplina, hacia un proceso de madurez con una efectiva calidad de mejoramiento, esto es considerado un CMM.

Extreme Programming (XP).

Se trata de un método ágil que se basa en una serie de reglas y practicas orientadas directamente a alcanzar el objetivo final del proyecto. Sus creadores, Kent Beck y Ward Cunningham dicen que estamos frente a una metodología que se aplica a proyectos de riesgo, donde los requerimientos se van dando poco a poco, o inclusive varían a lo largo del ciclo de vida del proyecto. Esta metodología divide al proyecto en cuatro fases distintas:

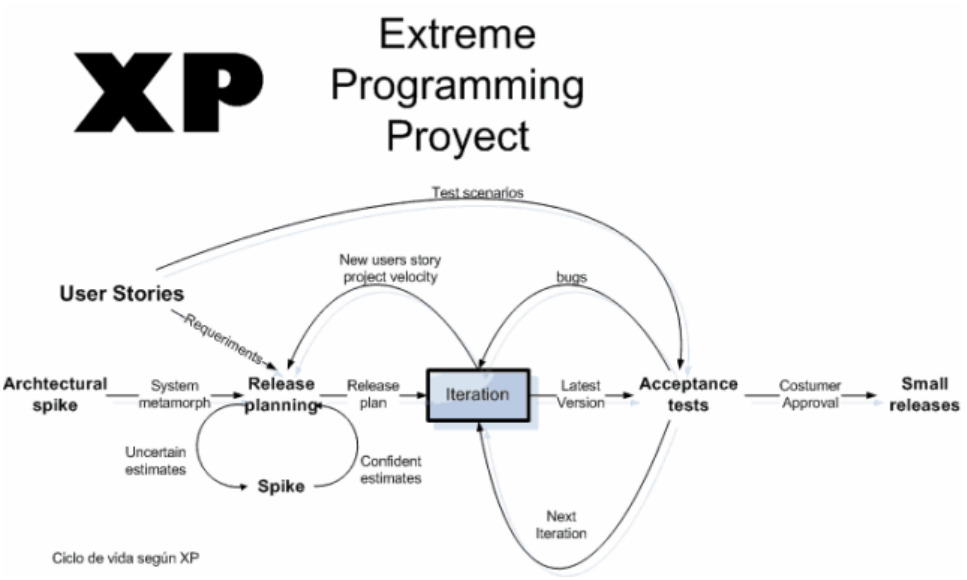


Ilustración 2.-Ciclo de Vida del Extreme Programming

Planificación.

Es la fase de inicio del análisis del proyecto. La primera regla que se aplica es la bitácora de usuarios (User stories), que sirven para realizar las estimaciones de tiempo y obtener de forma gráfica los requisitos del proyecto, remplazando la extensa documentación generada por las metodologías pesadas. En esta etapa también se crea un plan para realizar pequeñas versiones del proyecto, que van descubriendo en cada iteración nuevas funcionalidades. Este "plan de iteraciones" resulta muy practico si los requisitos cambian muy a menudo, pues en cada una de las iteraciones se agrega, quita o cambia dicha funcionalidad.

Diseño.

La simplicidad es el objetivo de esta fase, de forma que los requisitos se satisfagan correctamente. Con un diseño simple, la codificación resulta más rápida y el costo final del proyecto será menor como versión de esta fase, la refactorización que nos ayuda a mejorar la calidad en cada nueva iteración, también quedará prohibido añadir funcionalidades que no hayan sido planificadas.

Codificación.

Aquí se encuentra una regla fundamental para la correcta aplicación del eXtreme Programming:

El cliente siempre estará disponible para el equipo de trabajo. De hecho, a veces hasta se toma al cliente como parte del equipo de trabajo. Si se asegura la disponibilidad del cliente, siempre se puede contar con información adecuada en los momentos precisos. También propone que la codificación se realice en parejas de programadores, para mantener un alto nivel de desarrollo con menos errores, evitando que esto impacte en el tiempo total para el desarrollo.

Pruebas.

Todo el código desarrollado debe tener pruebas unitarias y debe pasarlas satisfactoriamente. Para cada error (bug) encontrado debe de realizarse una prueba unitariá hasta que este sea solucionado.

MoProSoft (Modelo de Procesos para la industria del Software).

Modelo de desarrollo de Moprosoft

Publicado por la Secretaría de Economía del Gobierno Federal Mexicano como base para la Norma Oficial Mexicana (NOM) y desarrollado por la facultad de Ciencias de la Universidad Nacional Autónoma de México (U.N.A.M).

Tiene por objetivo la presentación de un modelo de procesos para la industria del software que sea capaz de estandarizar la operación a través de mejores prácticas en gestión e Ingeniería de Software.

Según promete el documento, la adopción del modelo elevará la capacidad de ofrecimiento de servicios en calidad y alcanzará niveles internacionales de competitividad.

En relación con el actual trabajo, será el modelo base en el desarrollo de la aplicación, ejecutando la operación de dichas prácticas en cuanto a la creación de software se refiere.

MoProSoft es una Metodología Pesada por herencia, pues recopila en su propuesta de ingeniería de software organizada en áreas de aplicación, equivalencias puntuales con CMMI (áreas clave de procesos). Sin embargo pretende ser flexible para su adopción sobre todo en las PyMES con los siguientes requisitos:

1) fácil de entender.
2) fácil de aplicar.
3) No costoso en su adopción.
4) Ser la base para alcanzar evaluaciones exitosas con otros modelos o normas, tales como ISO 9000:2000 o CMM V1.1.

Moprosoft está, al igual que CMMI no solo con alcance sobre la ingeniería de software, si no de toda una entidad organizativa.

MoProSoft es un modelo integrado donde las salidas de un proceso están claramente dirigidas como entradas a otros; las prácticas de planeación, seguimiento y evaluación se incluyeron en todos los procesos de gestión y administración; por su parte los objetivos, los indicadores, las mediciones y las metas cuantitativas fueron incorporados de manera congruente y práctica en todos los procesos; las verificaciones, validaciones y pruebas están incluidas de manera explícita dentro de las actividades de los procesos; y existe una base de conocimientos que resguarda todos los documentos y productos generados.

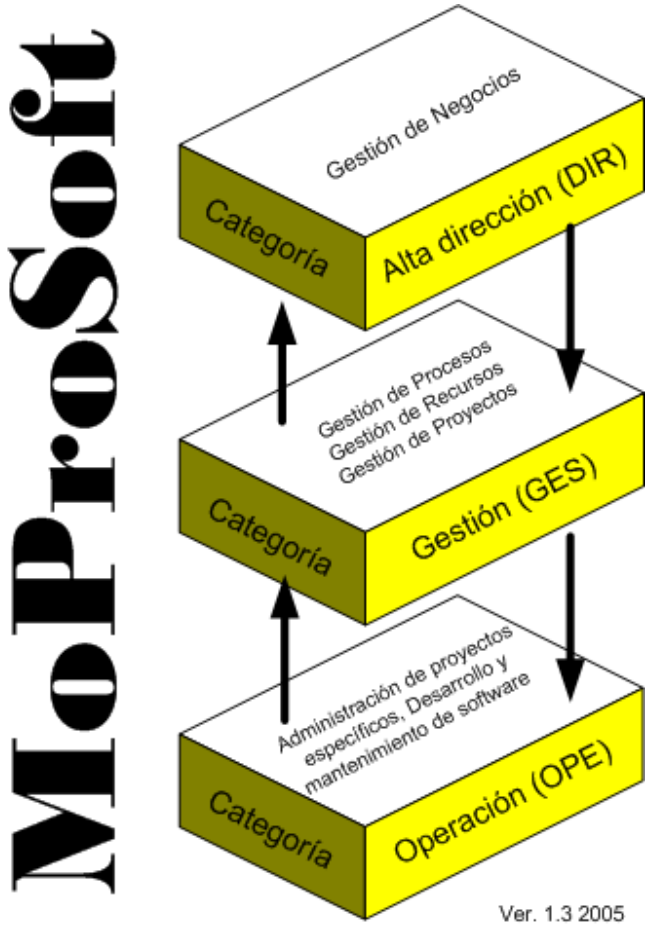


Ilustración 3.- Modelo de desarrollo MoProSoft.

La característica principal de MoProSoft es precisamente un “Enfoque basado en procesos", donde el desarrollo y mantenimiento del software es llevado a cabo a través de una serie de actividades que realizan equipos de trabajo.

En cuanto a la producción de software, el papel de los recursos humanos esta bien definido y establecido en “roles” para las prácticas de cada actividad dentro de los procesos:

Rol	Abreviatura	Capacidad
Responsable de la administración del proyecto especifico	RAPE	Capacidad de liderazgo con experiencia en la toma de decisiones, planeacion estratégica, manejo de personal y desarrollo de software
Responsable de desarrollo y mantenimiento de software	RDM	Conocimiento y experiencia en el desarrollo y mantenimiento de software
Analista	AN	Conocimiento y experiencia en la obtención, especificación y análisis de los requerimientos.
Diseñador de interfaz de usuario	DIU	Conocimiento en el diseño de las interfaces de usuario y criterios ergonómicos
Diseñador	DI	Conocimiento y experiencia en el diseño de la estructura de los componentes de software.
Programador	PR	Conocimiento y/o experiencia en la programación, integración y pruebas unitarias.
Responsable de pruebas	RP	Conocimiento y experiencia en la planeacion y realización de pruebas de integración y de sistema.
Revisor	RE	Conocimiento en las técnicas de revisión y experiencia en el desarrollo y mantenimiento del software.
Responsable de Manuales	RM	Conocimiento en las técnicas de redacción y experiencia en el desarrollo y mantenimiento del software.
Revisor de documentación	RD	Revisar y concentrar la documentación necesaria, para aporte de la base del conocimiento
Equipo de Trabajo	ET	Conocimiento y experiencia de acuerdo a su Rol.
Cliente	CL	Interpretación del estandar de la especificación de requerimientos.
Usuario	US	Ninguna.

Tabla 2.- Roles involucrados en el desarrollo y mantenimiento de Software.

Según la directora del grupo de desarrollo (Dra. Hanna Oktaba), en el alcance del documento, es posible adaptar el modelo, por lo que este trabajo, debido a su naturaleza, retoma y adapta la categoría de operación en la construcción y desarrollo del Módulo de Evaluación de Practicas de Laboratorio.

1.3.3 Modelos de contenido

1.3.3.1 Sharable Content Object Reference Model (SCORM).

En español se puede pronunciar como "modelo del objeto de referencia de contenido compartido". SCORM es un conjunto de estándares técnicos que permiten a sistemas basados en web el encontrar, importar, compartir, reutilizar, y exportar el contenido; aprender de una manera estandardizada.

Asume la existencia de un conjunto de servicios llamado por algunos "sistema administrador para aprender" y por otros "sistema de administración de contenido para el aprendizaje", antes también se le conoció como un sistema "de la instrucción asistida por computadora".

Sin importar la terminología, en el mundo de la educación, SCORM es un conjunto de servicios que provee contenido de enseñanza, sin perder de vista el progreso del aprendizaje, tomando en cuenta la secuencia en que los "objetivos" son dispuestos, controlando el avance del estudiante a través del proceso de enseñanza.

Es su mayoría, el contenido web está "ligado" de una página a otra; dentro de un "**LMS**" estas ligas son "inteligentes" y saben que es lo será dispuesto al "alumno". Es decir, que cuando un alumno ha completado un nivel u objetivo, el sistema sabrá cual contenido será el correspondiente en consecuencia, según del resultado del nivel terminado (SCORM Ver. 2004).

Cabe resaltar dos cosas importantes:

1. SCORM es necesario para estandarizar la entrega y seguimiento de la experiencia del aprendizaje dirigido, para definir el comportamiento y la lógica de una experiencia compleja en donde el contenido puede ser re-usado, movido, cambiado, buscado e incluso re-contextualizado.

2. SCORM habilita de una forma compleja y con un objetivo especifico el aprendizaje, que va mas allá de lo que pueda lograrse con un simple contenido de hipertexto ligado.

1.4 Arquitecturas de sistemas

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Cabe señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento.

1.4.1 Arquitectura StandAlone

Una Arquitectura Standalone se caracteriza por contener todos sus elementos funcionales en si mismos. Es decir, un programa da cuenta de si mismo, su funcionalidad, seguridad y cada uno de los demás elementos convergen siempre dentro de ella misma.

Dentro de las vistas fundamentales de la arquitectura monolítica del software, los componentes de cada aspecto funcional no interactúan mas que consigo mismo, la dinámica funcional es completamente interior, lo que nos permite representar a estas aplicaciones como un solo bloque en el cual las entradas producen las salidas finales.

Esta construcción de software se da de acuerdo a una arquitectura de modelo de dominio específico para aplicaciones particulares, que son además representaciones reales del proceso que se desea automatizar.

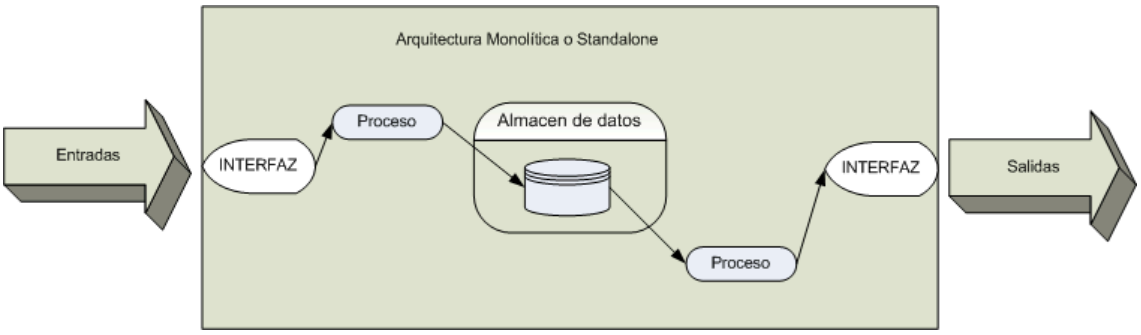


Ilustración 4.- funcionamiento de una arquitectura monolítica

Dentro de las arquitecturas de dominio específico existen dos vertientes:

- Modelos genéricos que son abstracciones de sistemas reales, conteniendo dentro de ellos mismos sus características principales.
- Modelos de referencia que son modelos abstractos y describen una clase mayor de sistemas, es decir que son una manera de informar a los diseñadores de la estructura general de esta clase de sistemas

1.4.2 Arquitectura Cliente/Servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como "servidores" estos últimos responden a la demanda del cliente que la produjo.

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 la Internet.

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.

IBM define al modelo Cliente/Servidor. *"Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas"*. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o clientes, resultan en un trabajo realizado por otros computadores llamados servidores.

Cualquier combinación de sistemas que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde esta ubicada. Es una arquitectura de procesamientos cooperativa donde uno de los componentes pide servicios a otro. También es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.

El término *cliente/servidor* es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante. "Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información."

Elementos principales

Los elementos principales de la arquitectura cliente servidor son justamente el elemento llamado cliente y el otro elemento llamado servidor. Por ejemplo dentro de un ambiente multimedia, el elemento cliente seria el dispositivo que puede observar el vídeo, cuadros y texto, o reproduce el audio distribuido por el elemento servidor.

Por otro lado el cliente también puede ser una computadora personal o una televisión inteligente que posea la capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audio, fotografías digitales y texto y los distribuye bajo demanda de ser una maquina que cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brinda éstos al cliente.

Que es un cliente:

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

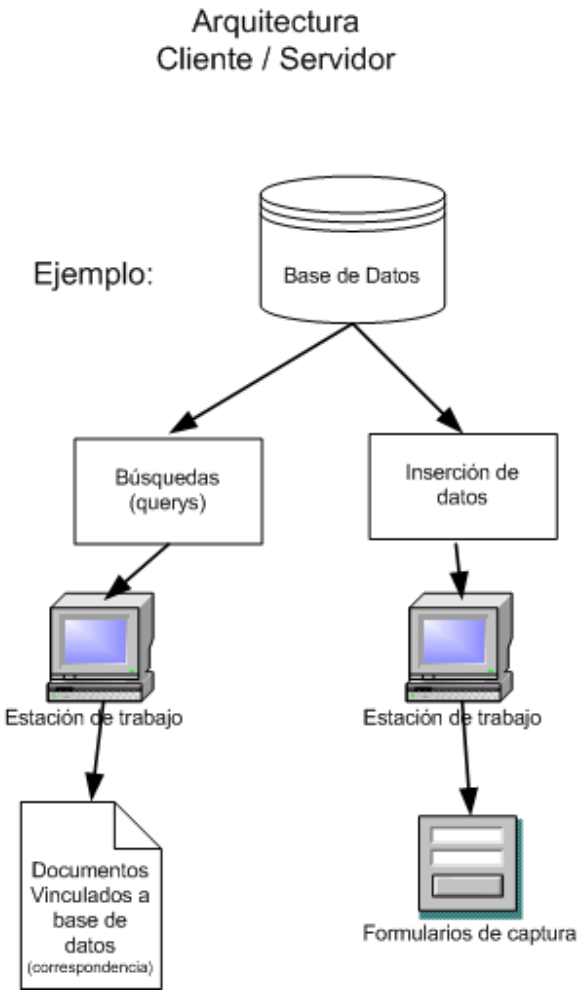


Ilustración 5.- Modelo con servicios distribuidos

Que es un servidor:

Será cualquier recurso de cómputo (software) dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de

redes LANs o WANs, para proveer de múltiples servicios a los clientes tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

En esta aproximación, y con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente/Servidor, debemos identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Presentación/Captación de Información
- Procesos
- Almacenamiento de la Información

Y se integran en una arquitectura Cliente/Servidor en base a los elementos que caracterizan dicha arquitectura, es decir:

- Puestos de Trabajo
- Comunicaciones
- Servidores

Características

En el modelo CLIENTE/SERVIDOR podemos encontrar las siguientes características:

1 El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
2 Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
3 Un servidor da servicio a múltiples clientes en forma concurrente.
4 Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

5 La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

6 Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

7 Además se constituye como el nexo de unión mas adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadoras, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

8 Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos de este host central y otros sistemas de la organización ponen a su servicio.

En conclusión, el modelo Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de protocolos: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

1.4.3 Arquitectura 3 Capas

La arquitectura de tres capas es muy usada para los desarrollos de soluciones empresariales, en especial aquellas propuestas por Microsoft, que suelen utilizar la Web como medio de transporte de datos. Estas soluciones se descomponen en componentes que forman unidades operacionales por si mismas, que se pueden conceptualizar bajo el modelo de la caja negra. Estos componentes se ubican, por su actividad en estratos o capas. La arquitectura de Aplicaciones en 3 capas Se ha convertido en el estándar para el software empresarial.

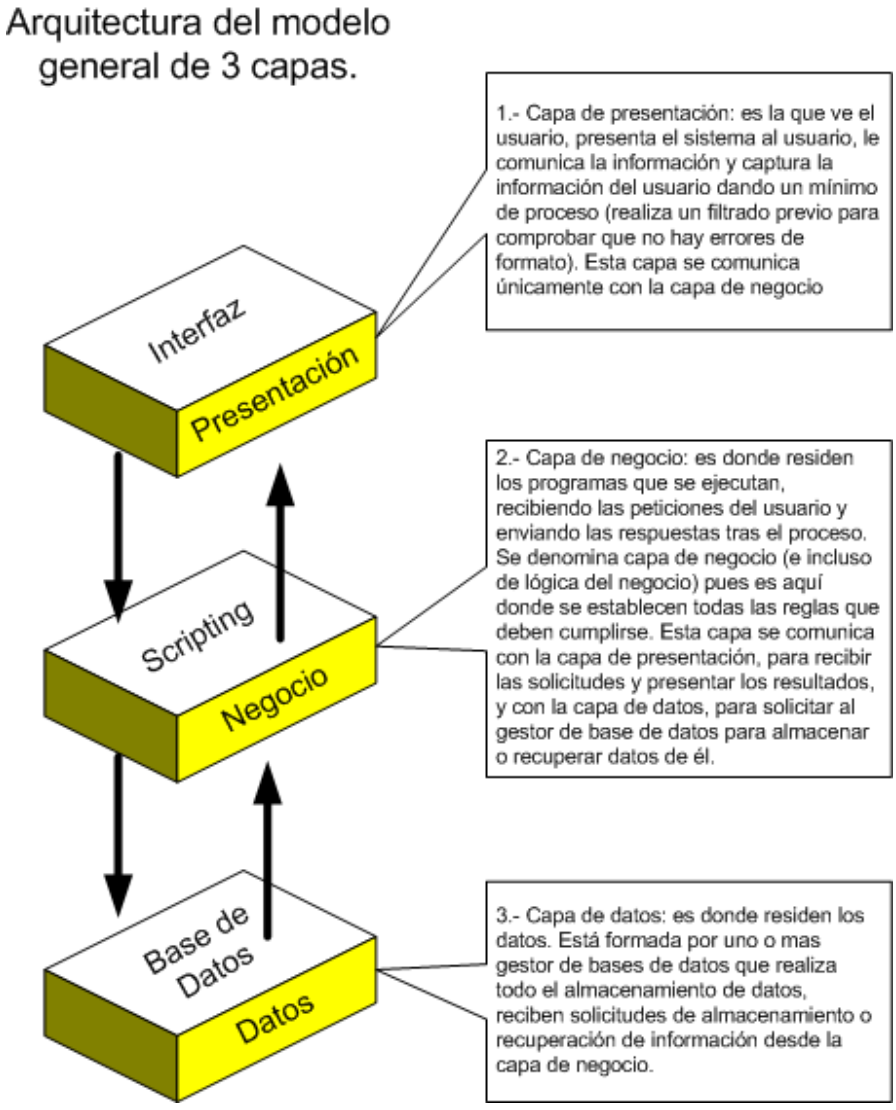


Ilustración 6.- Modelo General de 3 Capas

Todas estas capas pueden residir en una única computadora, si bien lo más usual es que haya una multitud de computadoras donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en la misma computadora, y si el crecimiento de las necesidades lo aconseja, se pueden separar en dos o más computadoras. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varias computadoras las cuales recibirán las peticiones de la computadora en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en una o más computadoras que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de computadoras sobre las cuales corre la capa de datos, y otra serie de computadoras sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación / Lógica de Negocio/ Datos.

En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física.

Por ejemplo:

- Una solución de tres capas (presentación, lógica, datos) que residen en una sola computadora (Presentación+lógica+datos). Se dice, que la arquitectura de la solución es de tres capas y un nivel.
- Una solución de tres capas (presentación, lógica, datos) que residen en dos computadoras (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

- Una solución de tres capas (presentación, lógica, datos) que residen en tres computadoras (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

1.4.4 Solución del sistema

Durante la construcción del proyecto del “Módulo de Evaluación de Practicas de Laboratorio”, se empleara el modelo básico de tres capas y la arquitectura de cliente / servidor.

Será una aplicación basada en Web, bajo el Modelo de Procesos de la Industria del Software (MoProSoft) en su desarrollo y construido únicamente con herramientas de código abierto (Open Source).

El diseño a través de diagramas de hipervínculos (diagramas “H”) aportará una visión global de los procesos y el flujo de datos. Además los diagramas de hipervínculos son una herramienta que permite representar en forma compacta varios aspectos de un sistema:

- Comunicación entre entidades Web.
- Especificación de rutas de navegación.
- Definición de las secuencias de interfaces.
- Representación jerárquica de los módulos que integran el sistema.
- Representación de arquitecturas para otros módulos extendidos.
- Representación de las entidades Web que integran el sistema de información y las entidades Web estáticas caracterizadas por mostrar contenido específico.

Además aportará una buena pista en la construcción del prototipo y de la estructura de la base de datos en el diagrama de entidad relación.

A partir del diseño del prototipo en los diagramas “H” se desarrolla la interfaz de usuario, que corresponde a la capa de presentación. Las validaciones en la entrada de datos el formato de la interfaz, el diseño gráfico de la interfaz de usuario son elementos esenciales de esta capa.

Las rutas de navegación nos periten la identificación de la relación entre los procesos, el flujo de datos, el manejo de las transacciones y en general toda la programación necesaria para resolver la aplicación, es la capa de negocio dentro de la arquitectura.

De igual forma, a partir de nuestra capa de presentación, se pueden encontrar los datos para formar el modelo conceptual de entidad relación para la base de datos a utilizar, también los datos necesarios para el funcionamiento de las transacciones que quedan fuera de la interfaz del usuario y su forma de almacenarlos, complementan la estructura, es la última capa, la capa de datos.

La nomenclatura que usaremos para los diagramas H será de nivel, como observaremos en la figura siguiente:

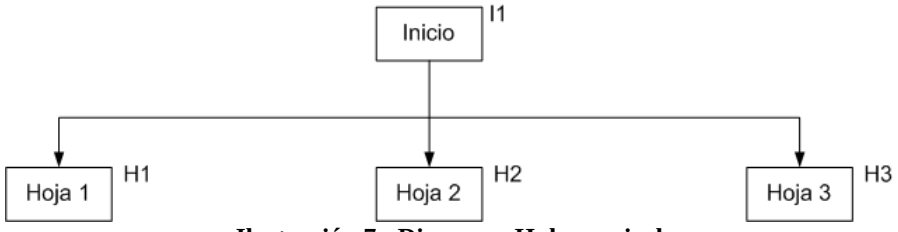


Ilustración 7.- Diagrama H de un nivel

La disposición de cada recuadro indica una “pantalla” es decir una representación al usuario en cada fase del proceso en el que la aplicación solicita y devuelve datos. Es importante tomar en cuenta que la asignación de nombres para cada pantalla (esq. Sup. Izq. a menos que el espacio lo impida) se determinan en forma incremental de izquierda a derecha.

A parte de la hoja de inicio, todas las paginas subsecuentes debe de conservar como su nombre la página predecesora y la subsecuencia correspondiente. Esto nos ayuda a identificar puntualmente cada una de las posibles pantallas (independientemente de la cantidad de datos

expuestos) con las que el usuario debe encontrarse, como deben de ser formateadas y todas las peculiaridades de las imágenes, letras, colores, etc… deben de aceptarse.

Cuando se utilizan muchos niveles (jerarquías) la representación se ve como en el siguiente ejemplo:

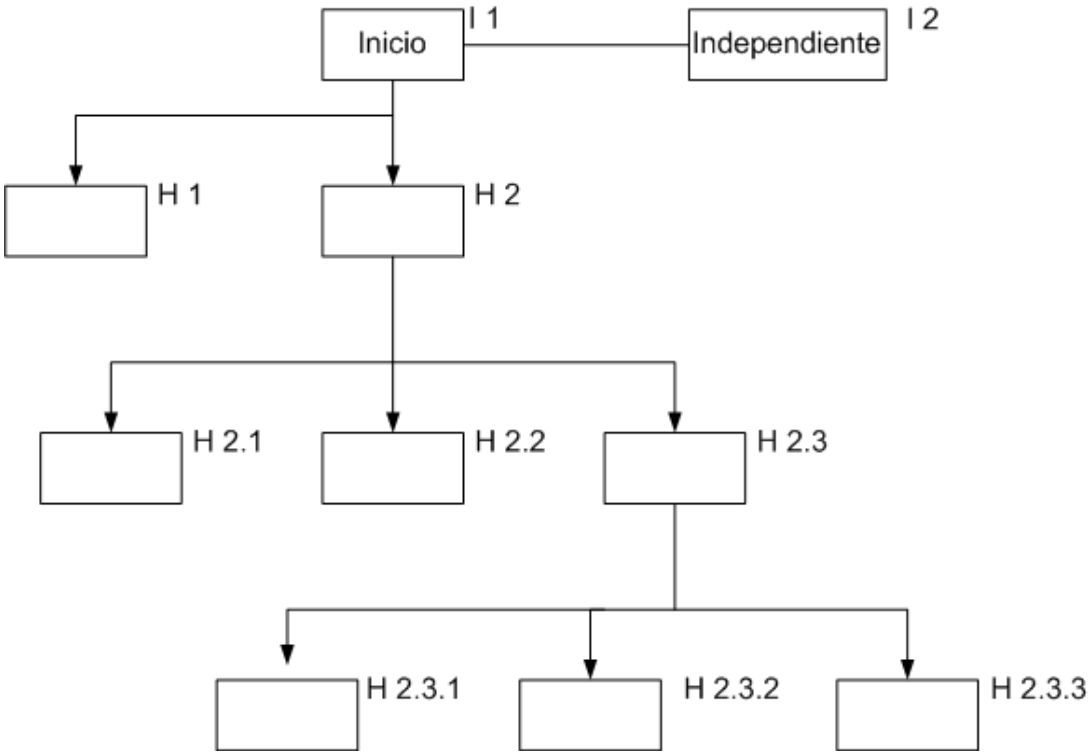


Ilustración 8.- Diagrama H 3 niveles

De acuerdo con el modelo de Moprosoft se lleva un control en el proceso de diseño, verificando en una tabla la descripción de la interfaz, su nomenclatura y el proceso o subproceso al que pertenece:

Nomenclatura	Descripción	Proceso
I1	Inicio	P1
I2	Ayuda	P1
H2	Inscripciones	P2
H2.3	Reporte de Inscritos	P2

Tabla 4.-Ejemplo de guía de descripción de la Interfaz.

La aplicación se desarrollará de acuerdo a estas dos herramientas que trabajan en el nivel de presentación de nuestro modelo de 3 capas.

CAPÍTULO 2

Definición del Proyecto

2.1.-Análisis del proceso

2.1.1 Revisión del producto.

Para llevar a cabo este proyecto, se utilizará el modelo de procesos para la industria del software (MoProSoft). Es la categoría de operación la que aplica en este apartado sobre el análisis del proyecto. La intervención de RDM y AN (Tabla 2) son los roles que participan durante este capítulo.

Definición General.

El "Módulo de Evaluación para Practicas de Laboratorio" es una aplicación versátil que pertenece al grupo de las WBT (Web Based Training) y puede integrarse a los desarrollos de e-learnig. Tiene como propósito ser capaz de permitir a los usuarios un manejo ágil y sencillo, sin que deban mediar grandes conocimientos sobre como funciona la aplicación. Se divide en dos subprocesos relacionados, el primero tiene que ver con los usuarios "Docentes"; quienes desempeñan este rol tienen dentro de sus actividades crear los reactivos/practicas y registrar a los "alumnos" que son quienes desempeñan el segundo rol, y a ellos corresponde contestar los reactivos. Entre otras cosas, se espera construir este proyecto usando puras herramientas de código abierto.

Catalogo de procesos

Contiene los procesos involucrados categorizados por actividad dentro de la aplicación en general. Aquí se describen las actividades correspondientes a cada rol.

Proceso	Descripción	US Involucrado
P1	Autenticación	Profesores y Alumnos
P2	Registrar alumnos	Administrador
P3	Registrar grupos	Adminstrador
P4	Registrar materias	Profesores
P5	Registrar Practicas	Profesores
P6	Registrar reactivos	Profesores
P7	Consultar evaluaciones	Profesores y alumnos
P8	Consultar practicas	Alumnos
P9	resolver reactivos	Alumnos
P10	Impresiones	Alumnos y profesores
P11	Evaluar y Guardar calificación	Automático

Tabla 3.- Catalogo Gral. de procesos

Actividad

Alumno:

1) Consultar Calificaciones

2) Resolver prácticas

- Pantalla de prácticas a resolver
- Escoger practica
- Desplegar practica
- Contestar Reactivos
- Guardar
- Ver evaluación

Profesor:

1) Consultar:
- Listado de grupos
  - Listado de Alumnos
    - Imprimir Listado
  - Calificación por alumnos
    - Imprimir Calificaciones
- Listado de Prácticas
  - Listado de reactivos

2) Crear registro de práctica
- Subir archivo.
- Crear reactivo(s).
- Editar status de la práctica

3) Registrar usuarios
- Registrar alumnos.
- Editar Alumnos.
- Registrar Profesores.
- Editar Profesores.
- Registrar Grupos.
- Editar Grupos.
- Registrar Alumnos x grupo.
- Editar Alumnos x grupo.
- Registrar Asignatura.
- Editar Asignatura.
- Registrar Laboratorio (aula).
- Editar Laboratorio (aula).

Diagrama de relación de procesos

En el diagrama de relación de procesos se muestran los procesos y su conexión entre ellos, en base a su funcionamiento y a partir del diagrama general se descompone en los subprocesos y su comportamiento.

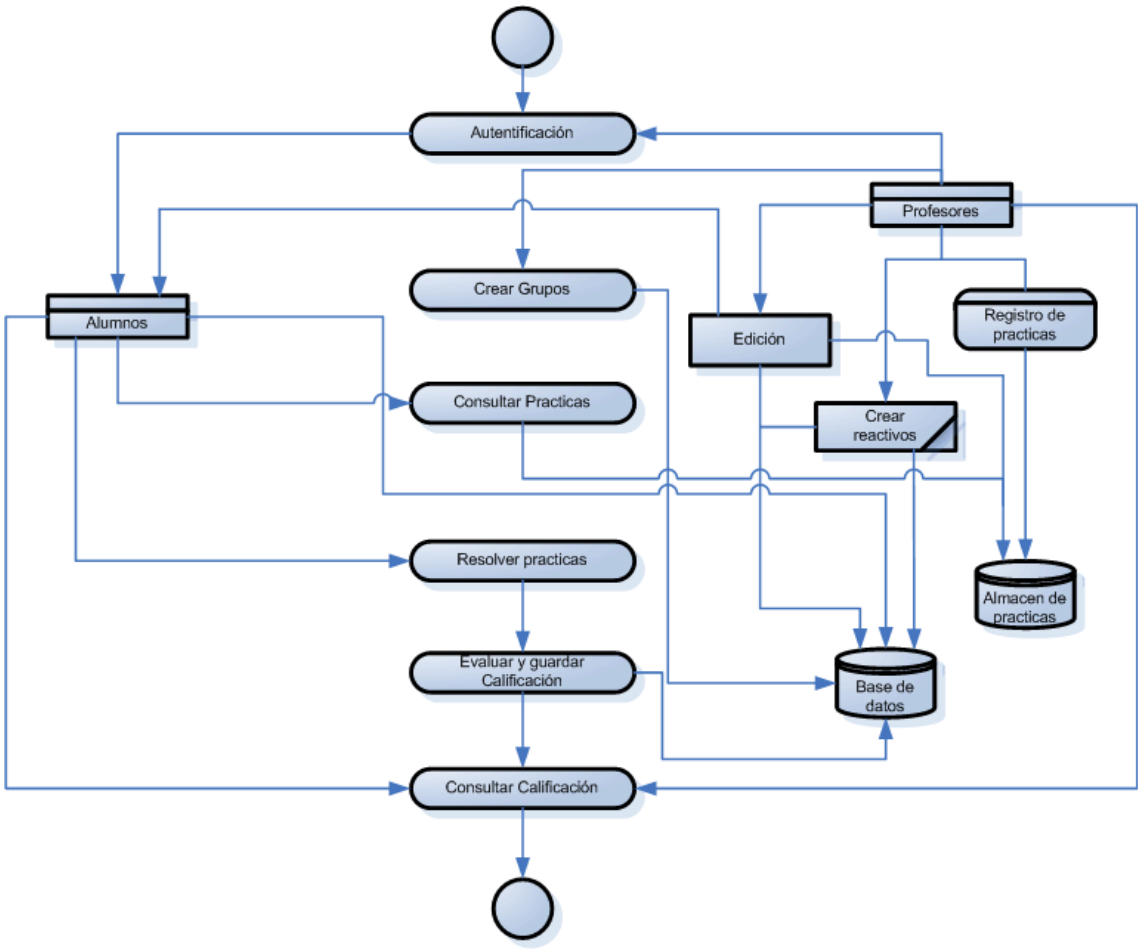


Ilustración 9.- Relación de procesos

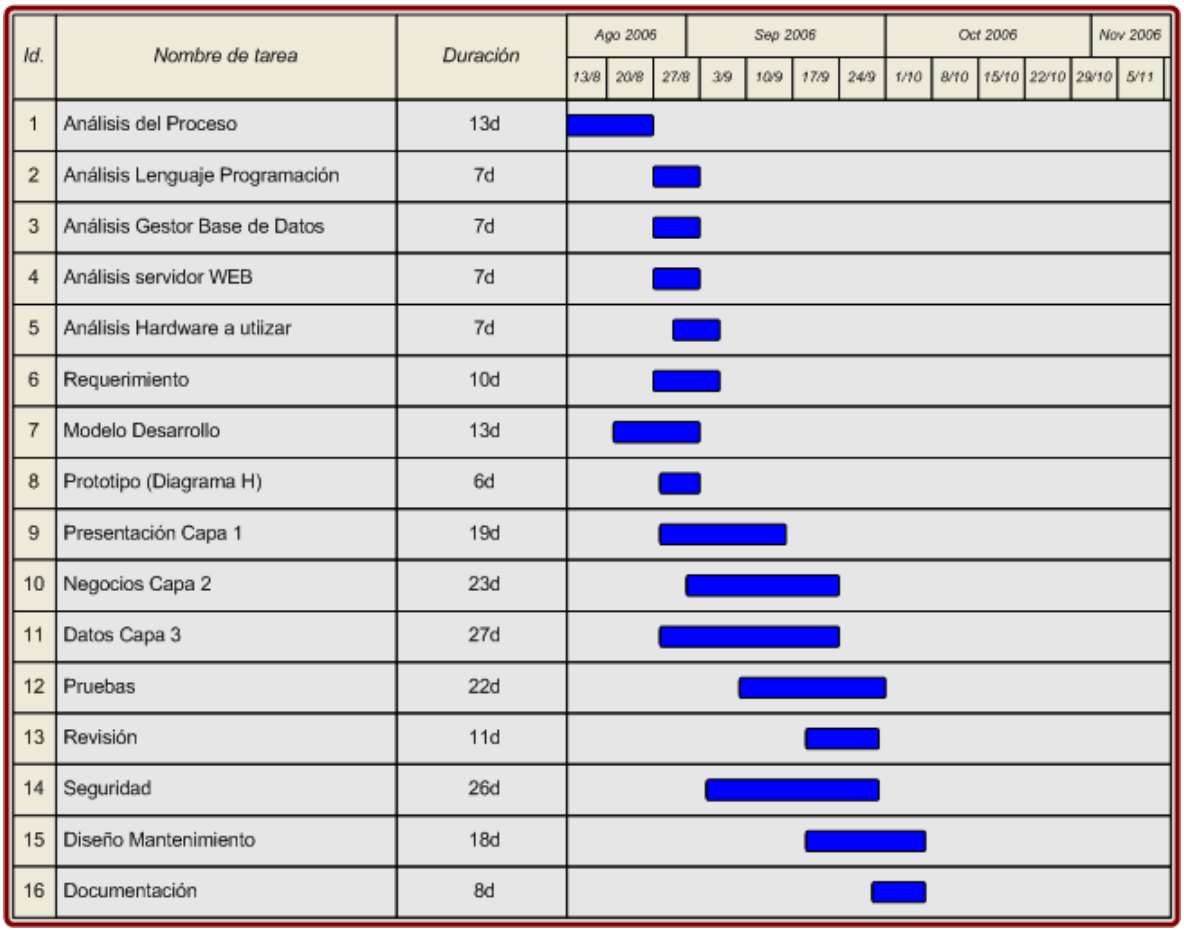
El papel de Analista en esta parte es relevante, pues identifica de manera general la relación que hay entre cada proceso y hace posible la separación en subprocesos. Para trabajar dentro de la arquitectura de tres capas, la identificación de cada proceso permite comprender el flujo de datos que se construirá en la capa de negocio.

El catalogo de procesos permite controlar el desarrollo hacia los subprocesos y se convierte en una herramienta útil que el diseñador de la estructura percibe para las demás capas de la arquitectura.

La capa de presentación, es la que marca la pauta sobre que datos se utilizaran o requerirán y como deben ser presentados de cara al usuario en esta aplicación.

2.1.2 Plan de trabajo

Esta es la tarea más importante del Responsable del Desarrollo y Mantenimiento de Software (RDM) pues da asignación de las actividades, tiempos y estrategias en el seguimiento y calendario durante el desarrollo y mantenimiento del software.



Id.	Nombre de tarea	Duración
1	Análisis del Proceso	13d
2	Análisis Lenguaje Programación	7d
3	Análisis Gestor Base de Datos	7d
4	Análisis servidor WEB	7d
5	Análisis Hardware a utlizar	7d
6	Requerimiento	10d
7	Modelo Desarrollo	13d
8	Prototipo (Diagrama H)	6d
9	Presentación Capa 1	19d
10	Negocios Capa 2	23d
11	Datos Capa 3	27d
12	Pruebas	22d
13	Revisión	11d
14	Seguridad	26d
15	Diseño Mantenimiento	18d
16	Documentación	8d

Ilustración 10.- Calendario del plan de desarrollo.

Responsabilidades.

Estas son distribuidas de a cuerdo a las capacidades u competencias de los involucrados en el desarrollo del proyecto. La siguiente tabla de practica muestra de que forma son integrados los diferentes roles en el desarrollo del Módulo de Evaluación de Prácticas de Laboratorio.

Responsable	Actividad
RDM AN, DI	Análisis del proceso
AN PR, RDM	Análisis del lenguaje de programación
RDM AN, PR	Análisis del gestor de base de datos
RDM AN	Análisis del servidor WEB
RDM AN, CL	Análisis del Hardware a utilizar
RDM AN, DI	Requerimientos
RDM AN, CL	Modelo de Desarrollo
AN, DI, PR DIU, RP	Prototipo de Software (1) (Presentación)
DI PR, RP	Prototipo de Software (2) (Base de datos)
DI PR, RP	Prototipo de Software (3) (Negocios)
RE RP	Revisión de cambios
ET, AN, DI RP, RE	Seguridad
DI, PR, RE	Diseño de mantenimiento
RP PM	Documentación

Tabla 4.- Distribución de trabajo.

2.1.3 Evolución del Software en desarrollo.

Por supuesto que la etapa de mantenimiento es donde la aplicación, comienza la evolución integral, después del lanzamiento inicial. Es este es proceso marcado para todo el software como soluciones basadas en la programación de aplicaciones dinámicas que parten del modelo de tres capas, pudiéndose realizar independientemente o en conjunto en cada una de ellas.

2.2.- Análisis del Lenguaje de Programación

Lenguajes

Como parte esencial en el desarrollo de aplicaciones, los lenguajes son el medio y la forma para materializar las acciones de la computadora de acuerdo a los deseos y proyecciones de los desarrolladores. A veces se del lenguaje de programación y del lenguaje informático como si fueran la misma cosa, sin embargo no es del todo correcto, pues los lenguajes de programación son mas bien un subconjunto del vasto universo de lenguajes informáticos

2.2.1 SQL

(Structured Query Language o Lenguaje de consultas estructurado) este es un lenguaje utilizado para organizar, gestionar y recuperar datos de bases de datos. Como características, tenemos que es declarativo, de alto nivel (de no procedimiento), que permite una alta productividad en codificación gracias a su orientación al manejo de conjuntos de registros, en vez de registros individuales y una buena base teórica. Con SQL se explota la flexibilidad y potencia de los sistemas relacionales permitiendo una baraja de operaciones sobre los mismos.

Antecedentes

Surgió de IBM a finales de los anos 70, cuando Edgar Frank Codd propone el modelo relacional, definiéndolo como lenguaje SEQUEL (Structured English QUEry Language). Posteriormente en 1986 es adoptado por ANSI (American National Standards Institute), naciendo a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO. La última versión se llama SQL-99, y todas las bases de datos comunes son compatibles.

Este Lenguaje sirve de interfaz con los Sistemas Gestores de bases de datos (o también llamados Data Base Management Systems DBMS); del que el primero también fue aportado por IBM. sus funciones se extienden mas allá de una consulta o recuperación de datos, define

los datos mismos, la presentación de estos y su manipulación, capaz de ejercer operaciones sobre los datos contenidos

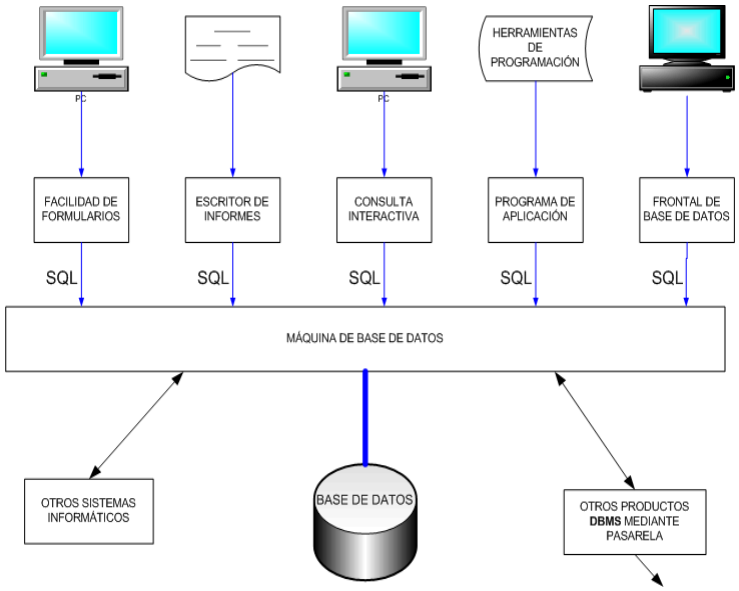


Ilustración 11. - Acciones del lenguaje SQL

Sentencias

Dentro del Lenguaje SQL que MySQL utiliza para tratar la información, suelen distinguirse varias categorías de sentencias (o comandos). El Lenguaje de consulta de datos DQL (Data Query Language) permite básicamente obtener datos de tablas y especificar la forma en que se presentarán. La sentencia esencial es SELECT. Se trata de la más simple de las categorías y es la primera en abordarse.

Existen dos categorías de sentencias SQL como Lenguaje de Modificación de Datos DML (Data Modification Lenguaje) cuyas sentencias se utilizan en la interrogación y manipulación de datos en esquemas de bases de datos ya existentes. Estas sentencias características son INSERT, UPDATE, DELET y DROP.

Otras categorías de sentencias SQL son el Lenguaje de definición de datos DDL (Data Definition Language) y Lenguaje de Control de Datos DCL (Data Control Language). Las sentencias del Lenguaje de definición de datos se utilizan para crear, alterar o borrar objetos de la base de datos tales como tablas, columnas y secuencias CREATE, ALTER y DROP. Las sentencias de control de datos se utilizan en el control de acceso a datos en la base de datos.

Como ejemplos característicos tenemos los comandos GRANT y REVOKE, estas últimas son muy usadas en la administración de la base de datos.

2.2.2 HTML

HTML de Ingles (Hyper Text Markup Languaje) es el lenguaje de etiquetado que se utiliza para las "publicaciones" a través del Internet. La estructura de construcción de textos es Hipertexto, esto significa que el texto puede recurrir a otros puntos de referencia dentro de si mismo o hacia otros documentos, se basa en un formato no propietario llamado SGML ("Standard Generalized Markup Language") y que sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial. Html cuenta con un registro en el estándar internacional ISO 8879.

Antecedentes

HTML ha sufrido una constante evolución, desde su primera publicación, ahora se encuentra en una version 4.0 y es en la se que basa otras derivaciones y extensiones, como el XHTML (eXtended HTML) que funciona como una aplicación basada en XML ver. 1.0 [eXtensible Markup Language (lenguaje de marcado ampliable o extensible) desarrollado por el World Wide Web Consortium (W3C).]

Para escribir documentos con formato HTML no hace falta mas que un editor de texto plano como en "Block de notas" en Windows, Emacs o vi en Unix/Linux, etc.... utilizando la notación de "etiquetado". Para su publicación, al tratarse de un estándar internacional, la gran mayoría de los navegadores populares son capaces de interpretar correctamente todas las etiquetas HTML, XHTML. que se utilizan en los documentos.

Las Etiquetas

Estas se identifican por aparecer entre los símbolos de mayor y menor que (<>). Generalmente describen las características de formato del documento que ha de interpretar el navegador. Algunas marcan principio y fin del formato con <algo> </algo>; otras disponen de algún elemento en particular y no generan la etiqueta final (o de cierre) como que dispone de una línea horizontal <hr>.

Sin embargo existen unas pocas etiquetas indispensables para que el documento sea reconocido e interpretado por el navegador como documento HTML y estas son:

```
<html>
       <head>
             <title>Titulo en el marco del navegador </title>
      </head>
      <body>
         Cuerpo de toda la página
    </body>
</html>
```

HTML nos brinda la oportunidad de publicar nuestros trabajos, desarrollos, aplicaciones y casi cualquier cosa que sea susceptible de realizarse en una computadora de forma tal que cualquiera pueda verlo, es decir, su portabilidad en cuanto a despliegue de archivos es universal.

Por lo general el despliegue dinámico de html se hace a través de algún lenguaje de script como perl o php, es decir, que el script "construye" la pagina html de acuerdo a las entradas que recibe, ya sea a través de un formulario o de la respuesta de algún otro programa.

HTML fue escogido para la generación de la interfaz del proyecto precisamente por buscar la adaptación a la plataforma que los usuarios pudiesen tener previamente instalada y causara el mínimo impacto en cuanto a los recursos necesarios y el mayor alcance en cuanto a cobertura de clientes.

2.2.3 JavaScript

Es un lenguaje creado por Netscape con el objetivo de integrarse a HTML y facilitar la creación de páginas interactivas sin necesidad de usar CGI o Java. Por esta integración, se toma a JavaScript como una “extensión” del HTML, está orientado a objetos diseñado para el desarrollo de aplicaciones cliente-servidor a través de Internet.

El código de programa de Javascript, llamado script, se introduce directamente en el documento html y no necesita ser compilado, es el propio navegador quien se encarga de traducir el código, es así que se pueden desarrollar los programas de manera que se puedan efectuar determinadas operaciones o tomar decisiones sin necesidad de acceder al servidor. Por ejemplo podríamos usarlo en verificaciones de campo para que las validaciones en las entradas sean siempre las correctas.

El problema inherente de que sea un lenguaje hecho para trabajar del lado del cliente, es que éste lo soporte y de cómo lo soporte. Por ejemplo, el Internet Explorer de Microsoft soporta una especie de JavaScript llamado jscript, que si bien es similar no es 100% compatible, lo que nos lleva a revisar la ejecución del código en diferentes navegadores para verificar su correcto funcionamiento.

En el desarrollo del proyecto se ha previsto llevar a cabo pruebas para 5 navegadores que son los clientes más usados:

1. Internet Explorer
2. Netscape
3. Mozilla Firefox
4. Konqueror
5. Safari

2.2.4 PHP

PHP es el acrónimo recursivo de "PHP: Hypertext Preprocessor", (Personal Home Page Tools). PHP fue originalmente diseñado en Perl, seguidos por la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador Danés-Canadiense Rasmus Lerdorf en el año 1994. En el año 1997 en Israel, se escribió el analizador gramatical (parser)y nació el php3. En 1999 Suraski y Gutmans reescribieron el código de PHP, creando lo que hoy se conoce como Zend Engine o motor Zend. En mayo de 2000 PHP 4 fue lanzado bajo el poder del motor Zend Engine 1.0.

Su uso es sencillo y es bastante parecido a los lenguajes más comunes de programación estructurada, como Perl y C. Les permite involucrarse a la mayoría de los programadores con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas. También es posible crear aplicaciones complejas con una curva de aprendizaje muy suave.

Características

- Su interpretación y ejecución se da en el servidor, esto permite que el cliente sólo reciba el resultado de la ejecución del script.
- Es posible utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos.
- Permite la conexión a diferentes tipos de servidores de bases de datos.
- Tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos.
- Esta una muy bien documentado en su página oficial[1].
- Código fuente abierto (Open Source).
- Gratuito.

Cuando las páginas contiene exclusivamente código HTML se pueden desarrollar y probar sin la necesidad de un servidor Web, porque el código es interpretado por el navegador del usuario. Esto así es inclusive si las páginas incluyen código JavaScript, ya que también es una tecnología que trabaja del lado del cliente (navegador). Sin embargo PHP se ejecuta en el servidor antes de que la página se envíe al usuario que realiza la petición.

El funcionamiento de PHP puede realizarse de 3 diferentes formas:

- Como un interprete externo (modo CGI).
- Como una extensión del servidor (via ISAPI y NSAPI) 4
- Como un módulo interno del servidor

La interfaz CGI (Common Gateway Interface) dota de capacidades de comunicación a los servidores Web con otros programas. Esta interfaz de comunicación es común en el sentido en que es exactamente igual para todos los servidores independientemente de la plataforma en la que trabajen. Esta interfaz establece una forma clara y precisa de

♦[4] Usando la API (*application program interface*) del servidor.

transferencia de información entre el servidor Web y los demás programas que se ejecutan en la misma máquina.

Usar PHP como una extensión del servidor (a través de su API), solo está disponible a partir de la versión 4 de PHP. Como modulo integrado al servidor es la forma mas aconsejable de uso, pues no requiere de ningún programa adiciona mas allá que el mismo servidor Web. Dependiendo de la plataforma donde se instala el intérprete de PHP, se debe decidir por alguno de los modos aquí descritos. Los factores determinantes en esta decisión serán entonces el sistema operativo, la carga prevista y el servidor Web que se usará.

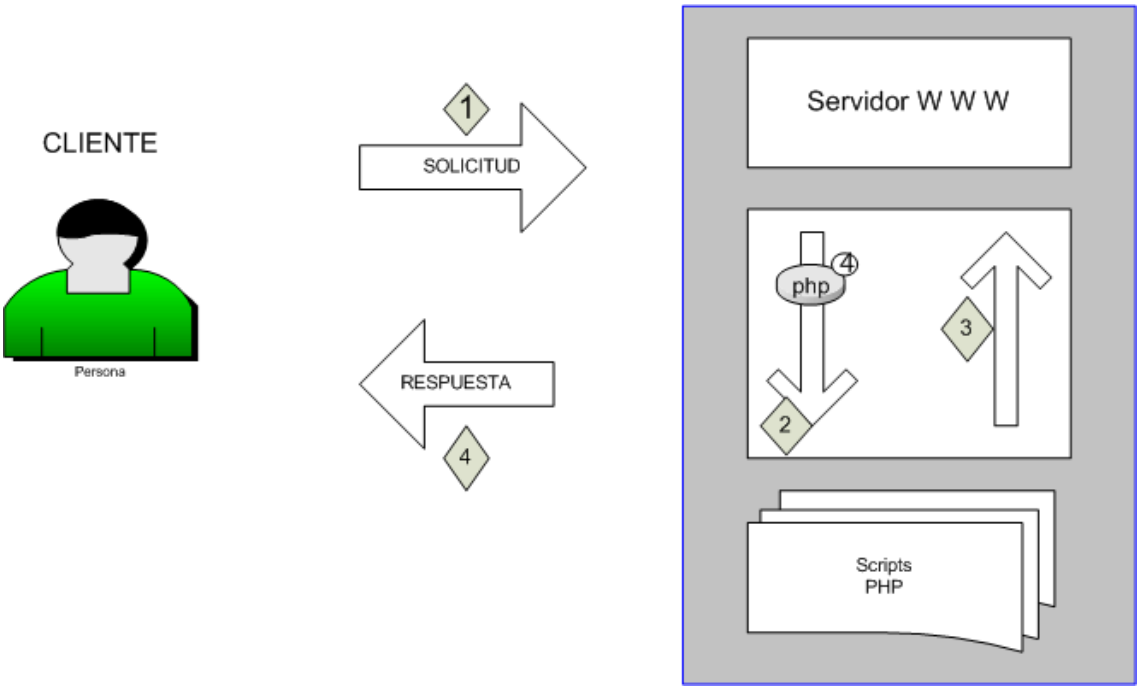


Ilustración 12.- Diagrama de funcionamiento de PHP

Motor ZEND

PHP4 incluye características propias de esta versión. Como el soporte de sesiones, nuevas instrucciones de bucles, tratamiento mas extenso en los arreglos (arrays), etc… pero el cambio mas importante esta la implementación de un motor de scripting que ha mejorado el rendimiento, hasta en 200 veces en algunos casos.

2.3 Análisis del Gestor de Bases de Datos.

2.3.1 MySQL

El DBMS que vamos a utilizar será MySQL. Porque en primer lugar se busca cumplir con el "estándar LAMP" que es importante tomar en cuenta en el mundo del código abierto, además de las características generales que requerimos para el desarrollo del módulo de evaluación, MySQL es un servidor de bases de datos relacionales muy rápido, multiusuario y multihilo.

Licencia

La licencia GPL de MySQL obliga a distribuir cualquier producto derivado (aplicación) bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero no desea distribuirlo bajo licencia GPL, puede adquirir la licencia comercial de MySQL que le permite hacer justamente eso.

Características

En un principio, MySQL no tenía los elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. Sin embargo, atrajo a los desarrolladores de páginas web con contenido dinámico, precisamente por su simplicidad; aquellos elementos faltantes fueron llenados por la vía de las aplicaciones que la utilizan. Poco a poco los elementos faltantes en MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

Disponibilidad.

También es un producto que generalmente viene, por antonomasia en las distribuciones mas populares, se le reconoce como paquete básico en las distribuciones Linux, aunque existen los portes a los demás Unix y también en Windows, en un intento por desplazar el caro MSQL. El desarrollo está bien soportado por la documentación y una empresa que le respalda (MySQL AB). También se puede descargar el código fuente de la página de MySQL AB e instalarse en prácticamente cualquier UNIX.[2].

Operación.

La sabida estabilidad de la reciente versión de producción de MySQL resulta tan confiable que no por nada empresas de prestigió han intentado comprarla [3], sin embargo lo que mas importa para este desarrollo es su valoración dentro del estándar LAMP, el cual recomienda usar la versión 4.0 en adelante. Existen numerosísimos Administradores gráficos para el manejo de la base de datos, en plataformas Linux, Windows y hasta Web (phpMyAdmin) con los cuales podremos revisar y verificar que nuestro desarrollo esta haciendo precisamente lo que deseamos.

Versión.

El Gestor empleado para el proyecto ya viene en formato binario (rpm) instalable desde la misma distribución, sin embargo el producto final debe de funcionar independientemente de la distribución Linux a utilizar. Las pruebas que se llevaran a cabo en la construcción del prototipo se harán en dos distribuciones diferentes con distintas versiones del DBMS.

Distribución	Versión DBMS
Fedora Core 3	MySQL 3.23.58
Centos 4.2	MySQL 4.1.12

Soporte

El soporte a usuarios y administradores que trabajan con este gestor de base de datos es principalmente la comunidad de desarrollo.

2.4.- Análisis del Servidor WEB

2.4.1 Apache

El servidor Apache es un servidor Web (HTTP) producto del esfuerzo y colaboración de desarrolladores voluntarios en diferentes partes del mundo. Es el mas claro ejemplo del modelo de desarrollo de la comunidad de código abierto, dando como resultado la fundación APACHE(Apache Software Fundation). El código y su documentación son elaborados utilizando como medio de comunicación el Internet y utilizando herramientas (como los newsgroups, sourceforge, cvs, etc...) que fueron concebidas también dentro del abanico de licencias del mundillo del código abierto (Open Source); la idea de utilizar software libre para desarrollar mas aplicaciones, es que puedas tener la libertad de utilizarlas, revisarlas y enmendar errores (bugs). La fundación APACHE es ahora una entidad orgánica, donde aquellos que usan ese software, también tienen la oportunidad de mejorarlo.

Licencia.

Los proyectos Apache se caracterizan por un modelo de desarrollo basado en el consenso y la colaboración, y en una licencia de software abierta y pragmática [4].

Características.

El servidor HTTP Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) [5] y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor parcheado). Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Versión.

El núcleo 2.x de Apache tiene varias mejoras clave sobre el núcleo de Apache 1.x. Estas mejoras incluyen threads de UNIX, mejor soporte para plataformas no Unix (como

Windows), un nuevo API, y soporte de IPv6. Su trabajo es modular, lo que permite dosificar la carga en el servidor, según se requiera par las aplicaciones hospedadas. El prototipo también será probado en dos versiones del servidor Web. Cada una de estas viene en su distribución de forma binaria, asi que los más importante es como quede la configuración del archivo.

Distribución	Versión Apache
Fedora Core 3	MySQL 3.23.58
Centos 4.2	MySQL 4.1.12

Documentación:

Es un proyecto de software muy bien documentado. En su instalación se provee de un muy buen manual de instalación y de configuración; viene con un archivo de configuración igual además cuenta con diversos sitios que documentan cada un de sus módulos (ej: http://www.apacheweek.com/) y de las actualizaciones que van ocurriendo dentro del proyecto.

Niveles de seguridad:

Hoy por hoy se posiciona como uno de los mas versátiles en cuanto a seguridad se refiere, en comparación al IIS, iPlanet y otros de su competencia, apache ya no solo cuenta con autenticación a través de su propia base de datos, ahora se integra a las bases de datos mas comunes (MySQL, Postgres, LDAP, etc...). Es capaz de filtrado a nivel dirección IP así como integración en los lenguajes de script, como php, perl, etc...

Disponibilidad

El servidor Web ya forma parte de las distribuciones mas populares hoy en día, sin embargo se puede obtener de la pagina de la ASF, así como su código fuente para generar los binarios. De hecho esta portado hacia mas plataformas Unix e incluso Windows, compitiendo fuertemente con IIS.

Operación.

Al instalar nuestra distribución, el servidor ya viene habilitado, solo se hacen unos pequeños ajustes para usar los certificados y el https en servicio en el pto. 445, la configuración de nuestro servidor se encuentra registrada aquí, la habilitación de los módulos para nuestra configuración es esta.

Lo más destacado de Apache es su capacidad de operación, que con una buena configuración realiza un alto desempeño. Las estadísticas indican que el año pasado Apache se utilizaba en más del 70% de los sitios Web a nivel mundial (fuente Netcraft [6]).

Para ilustrar de manera concreta la configuración propuesta, es trabajo del Responsable de Pruebas realizar un sobre el rendimiento de este servidor web. Se hace sobre el destinado a alojar la aplicación.

Prueba de rendimiento:

Se ejecuta una prueba de simulación de carga sobre el servidor, es decir del numero de usuarios conectados a este simultáneamente. Para esto recurrimos a Jmeter, una aplicación basada en Java disponible gratuitamente desde el sitio de la Apache Software Fundation.

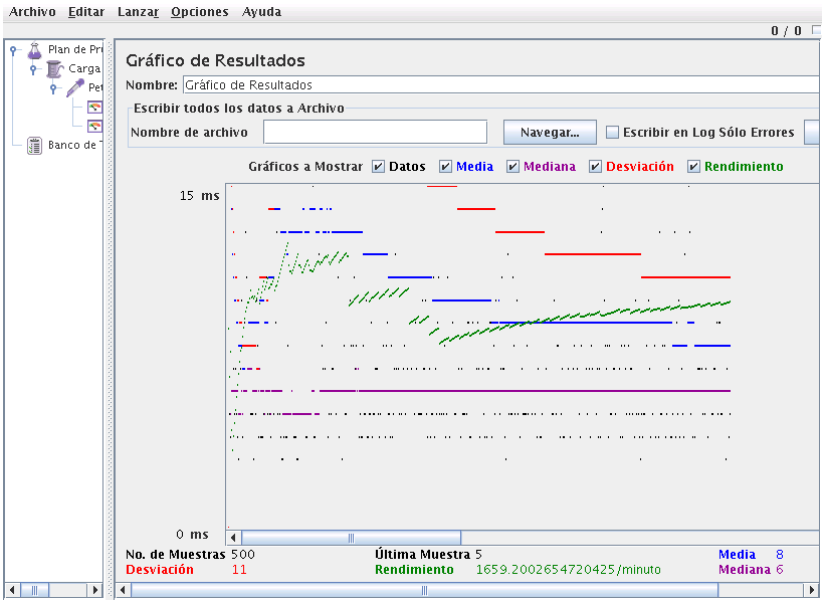


Ilustración 13.- Resultado gráfico de la prueba de Carga

2.5.- Análisis de Hardware a utilizar.

El hardware es un aspecto importante, se ha de tener en cuenta tanto el hardware para realizar el desarrollo de la aplicación (que se verá en el siguiente capítulo), como el hardware de producción.

El hardware de producción es donde se correrá la aplicación, por lo tanto, la proveeduría del hardware de desarrollo que el responsable de la Gestión de recursos facilite, deberá guardar una relación estrecha en funcionalidad con el equipo de producción.

Realizando un estudio preeliminar en cuanto a los equipos que el "cliente" mantendrá como equipo de producción, se ajustará a los parámetros del requerimiento del proyecto.

- Teniendo en cuenta:
- Arquitectura.
- Seguridad.
- servidor(es).
- Maquinas de usuarios.

El líder de proyecto y los programadores es sin lugar a dudas el punto de partida en cuanto a administración de los recursos para la disposición del proyecto.

2.5.1 Arquitectura

Esta muestra la infraestructura necesaria que el sistema necesita para operar y aportar la "experiencia del usuario" que se define en la descripción del proyecto. Generalmente se define por los consultores y el equipo técnico que trabaja durante la especificación de la aplicación. En un proyecto pequeño la arquitectura podría constar solo de paginas HTML dentro de un servidor Web, en este caso la arquitectura solo constará de los archivos y la estructura de directorios; pero conforme este crece, se convierte mas en un diagrama de redes

mostrando como la interfaz se integra con el "middleware" y los sistemas de "back-end" y el como de su conectividad y seguridad.

En el caso del proyecto de el "módulo de evaluación de practicas de laboratorio", la arquitectura básica de 3 capas es aplicada para la definición necesaria del hardware a utilizar.

2.5.2 Seguridad

En este apartado, gracias a la intervención de los consultores técnicos responsables (categoría de operación), se establece que debe protegerse y como, de acuerdo al catalogo de procesos del proyecto (2.1).

Se establecen las rutas de acceso a los recursos, como se introduce la información, las tecnologías de seguridad utilizadas (PGP, SSL, etc...) y el hardware que las protegerá; como en algunos casos existen tarjetas criptográficas o firewalls dedicados...

De acuerdo con el catálogo de procesos, es preciso prestar atención en las páginas donde la autenticación y seguimiento de sesiones, la custodia del servidor y la protección de la base de datos, como se muestra ne la figura.

2.5.3 Servidor(es)

Aquí se perfila el ambiente operativo del servidor, su hareware y el software, que permita anticipar las cargas de trabajo (en particular al mantener a varios usuarios en conexiones simultáneas) que brinde un nivel rendimiento aceptable dentro de los procesos en cuestión.

Es importante precisar si la aplicación la alojará el cliente o un ISP (proveedor de servicios de Internet), en general es tomada la segunda opción por varios motivos, entre ellos la disponibilidad, la velocidad de conexión, el factoraje del alojamiento o "hosting", y así por

varias ventajas, para aplicaciones comerciales suele utilizarse. En el caso del módulo de evaluación para practicas de laboratorio" este se alojará por el cliente mismo, dada la naturaleza del proyecto. Además el responsable de proyecto decide el sistema operativo en este caso es Linux, de acuerdo al estudio de costos y definición general (2.1).

Como regla general, entre mas poderoso sea un servidor, mas se incrementará su costo, pues el incremento de ancho de banda es un factor que el ISP observa. Mas procesadores y memoria extra darán mejor balance en la administración entre la carga y el rendimiento.

Tener en cuenta que si la aplicación debe compartir los recursos del servidor junto a otras aplicaciones es importante según el tamaño de ésta y de como podría crecer, pues el hecho de mantener o no un ancho de banda y demás recursos en una misma maquina junto a la aplicación podría bien disminuir su rendimiento. La propuesta de un servidor dedicado a la aplicación, aunque elevase el costo total, se pueden conseguir ventajas que permitirían validar dicha propuesta:

- Control total de forma remota.
- Instalación de cualquier software adicional que se requiera.
- Reiniciar la máquina.
- Configuración del servidor Web, del FTP y otros servicios exactamente como se desean.
- Por último, la configuración de los discos duros, los elementos del sistema operativo, creación de reportes de los logs del servidor y la creación de áreas de seguridad protegidas con password mediante las herramientas del propio sistema operativo.

Estas ventajas se logran sin el riesgo de padecer las limitaciones que imponga el ISP o interferir con los procesos de otros sitios particulares. Existen servidores virtuales que brindad un muy buen nivel de control de los recursos de la máquina, pero al final siempre quedará supeditado al límite que imponga el ISP.

También podemos mencionar las tecnologías del servidor y las interfaces que se deseen o no soportar, esto incluye ASP (Active Server Pages) conectividad a bases de datos (que bases de datos), Java Servlets, JavaScript, XML, JSP, CGI/Perl, PHP, CORBA, etc...

2.5.4 Especificación de la máquina del Usuario

La existencia de una variedad de configuraciones de software y hardware que pueden ser accedidos de la red hacia la aplicación del proyecto particular, es necesario definir los requerimientos mínimos del hardware cliente para poder brindar una mejor experiencia al usuario.

De la misma forma que se brindan los requerimientos mínimos, es preciso mencionar los requerimientos recomendados para producir el diseño de la interfaz, que sea capaz de ajustarse las necesidades que el mismo proyecto plantea en la ejecución de sus procesos, la codificación de forma adecuada y se puedan cubrir los objetivos principales. Entre las capacidades principales que se deben señalar tenemos:

- **Resolución de pantalla, profundidad de color y tamaño del monitor**.

La resolución de pantalla es una de las especificaciones más importantes a definir. La denominación común mas baja para la visualización es de 640 x 480, así que si lo que se necesita es asegurar que "todos" los visitantes sean capaces de tener una correcta visualización, será la que debiera de proponer, sin embargo este tamaño nos da muy poco espacio en el cual diseñar. Hoy en día los recursos en los usuarios, por lo general se ha visto favorecidos y la gran mayoría puede ser capaz de presentar una resolución de 800 x 600.

Existen actualmente pocos sitios diseñados para resoluciones de de 1024 x 786 o superiores, pues en algunos casos esta resolución supera la capacidad de los monitores mas antiguos que aun se usan en las máquinas de los usuarios (superVGA). La gran mayoría de los equipos nuevos vienen con monitores y tarjetas gráficas que son capaces de cuando menos lograr una resolución de 800 x 600. De la misma forma hay que pensar en el manejo del color y las imágenes que se incluyan, pues el usuario podría contar con un buen monitor pero una tarjeta gráfica de bajo desempeño, no pudiendo mostrar la intensidad de los colores pretendidos, o en su caso inverso, la presentación de la aplicación de forma extraña, sobre todo cuando se trata de tablas que pueda llenar el espacio de la pantalla.

- **Sistema operativo del Usuario.**

Este es otro aspecto importante, y es necesario que se le diga al usuario con que sistema operativo la aplicación funciona, no bastará con decir que versión del sistema operativo, sino que soporta versiones de 16-bit, 32-bit o 64-bit y que derivación (release). En algunos casos, la derivación incluye parches (patches) particulares o corrección de errores (bug fixes).

- **Audio.**

La mayoría de los sitios que manejan contenidos educativos suelen hacerse de recursos multimedia, entre estos se tiene los archivos de audio. Por lo general se usa algún formato universal, sin embargo existen diversos factores que nos puedan impedir usar los (como el copyright), lo que hace necesario advertir al usuario sobre los recursos con los que deberá contar para poder reproducir nuestros archivos. Esto también le advierte al usuario sobre la necesidad de contar con una tarjeta de procesamiento de audio y un par de bocinas.

- **Navegadores.**

La importancia de las pruebas que se lleven a cabo de la aplicación con diferentes navegadores y su correcta interpretación en cada uno de ellos, ampliará las posibilidades de que los usuarios comprendan lo que se transmite de forma correcta. Los navegadores están muy relacionados con el sistema operativo que los instala, como el explores en Windows o el safari de Macintosh, sin embargo existe otros mas que son independientes de la plataforma (o multiplataforma) como Opera.

- **Plug-ins.**

Forma parte de la estrategia en que se quiera enfocar a la audiencia, una experiencia multimedia dentro del desarrollo y presentación haría necesario pensar en estos "extensores" de funcionalidad para los navegadores. Nuevamente este tema también esta muy relacionado con el anterior, pues la existencia de uno u otro extensor (plug-in) dependen del navegador.

- **JavaScript, Java, marcos, cookies y scrolling.**

Puede un usar o no, alguno o todos estos elementos en el desarrollo de la aplicación, pero habrá que definir si esto será adecuado o no. Por ejemplo las validaciones de entrada de datos puede hacerse con estas herramientas en la capa de presentación, o en el scripting de la capa de negocio. Se debe mencionar porque no siempre los navegadores son capaces de ajustarse por completo a los estándares y usan un juego propio de interpretación del script, por ejemplo, jscript; o peor aun, un juego particular de scripting como VBscript.

2.5.5 Nota sobre el sistema operativo.

El sistema operativo planteado será Linux, es importante que se aclare que cualquier distribución mantiene una configuración por defecto (default) en la instalación que valiera la pena ajustar, tanto para la seguridad, como por el rendimiento. Configurar la seguridad en la capa de transporte (SSL) es cosa que debe de hacerse manualmente, pues la creación de certificados es responsabilidad del administrador de la máquina.

Sin embargo, el funcionamiento de la aplicación puede trabajar, con algunos ajustes, en otras plataformas Unix o incluso Windows, gracias a la portabilidad de las herramientas esenciales.

Las distribuciones en las que se llevará el desarrollo y probará la aplicación son Fedora 3 y Centos 4.2, que son versiones Libres y actualizadas de RedHat.

REFERENCIAS:

[2] http://dev.mysql.com/downloads/mysql/5.0.htm
[3] http://www.mysql-hispano.org/index.php?m=read&id=424
[4] http://www.apache.org/licenses/
[5] http://www.faqs.org/rfcs/rfc2616.html
[6] http://news.netcraft.com/archives/web_server_survey.html

Capítulo 3

Diseño y creación de la Aplicación.

3.1.- Requerimientos

Dentro del modelo de MoProSoft los participantes principales serán:

Responsable	Actividad
RDM, AN	Distribución de las tareas al equipo de trabajo según su rol, según el plan de desarrollo
AN, CL US	Documentar y/o modificar las especificaciones de los requerimientos: Fuentes de información. Requerimientos identificados para delimitar el alcance y su factibilidad, según las posibilidades y restricciones del cliente.
RP	Elaboración de un plan de pruebas para la validación de los requerimientos.
RD	Documentación sobre los requerimientos propuestos
RE	Verificación sobre la documentación del los requerimientos.

Tabla 5.- Requerimientos (responsables y actividades)

3.1.1 Software

Las herramientas que se escogieron para llevar a cabo este proyecto, son todas de naturaleza Open Source y la dinámica de modelo de trabajo es colaborativa.. De entrada, para que la aplicación funcione, es necesario contar con el hardware ya revisado en el apartado 2.5 y como resultado de la intervención del analista con el cliente y los usuarios se especifica el software a utilizar para el desarrollo y las pruebas de implantación:

Maquina servidor.

+ Software instalado:

- S.O. = Fedora 3 [1]
- DBMS = MySQL [2]
- Scripting = PHP [3]
- Web Server = Apache [4]
- Base del conocimiento = Web del proyecto -MediaWiki-[5]
- Contenedor de versiones = CVS [6]
- Administración de la base de datos = PhpMyAdmin [7]

Las estaciones de trabajo (desarrollo)

Las estaciones de trabajo usarán para:

- Desarrollo de interfaz web, phpscript y javascript = Bluefish [8] y Quanta [9].
- La creación y manipulación de Imágenes = The Gimp [10].
- Diagramas UML = Gaphor [11]
- Validación de hojas de estilo (en linea) [12]
- Estimación de costos (COCOMO II) [13]

3.1.2 Hardware.

De acuerdo con MoProSoft, correspondería a Categoría de gestión (GES) proporcionar los elementos para el funcionamiento de los procesos de la categoría de operación.

Servidor.

El servidor deberá ser una maquina en la que se pueda correr completo el estándar LAMP, sin embargo es importante tomar en cuenta que en un medio de producción se tendrá que evaluar sobre todo la carga de trabajo y almacenamiento, en caso de decidir alojar la base de datos en la misma maquina.

Para la base del conocimiento se utilizó una segunda máquina conectada igualmente a Internet, de forma permanente. Este servidor proporciona los servicios de documentación de este proyecto, la base de conocimiento del desarrollo y la presentación final en formato html. Cuenta de igual forma con el estándar L.A.M.P. manteniendo la base del conocimiento bajo resguardo en un repositorio de acceso privado.

Configuración.

Los servidores deben de cumplir en su instalación con el estándar L.A.M.P.

Como requisitos mínimos dentro del estándar:

- Linux con kernel 2.4.
- Apache ver. 1.3 o superior.
- MySQL ver 3.24 o superior.
- PHP ver. 4.x

Es importante mencionar que durante el desarrollo de la aplicación se utilizará la configuración de php con las variables globales activadas, tomando en cuenta que para su uso final estas deberán ser desactivadas para evitar agujeros de seguridad.

Estación de trabajo.

Se utilizan dos estaciones de trabajo y un equipo portátil. Para la instalación de las herramientas de trabajo, una computadora se dedico a contener los editores de html y los clientes de la base de datos.

La otra estación se dedicó a la producción del material gráfico, esta en especial con un sistema operativo OS X, debido a la falta de controladores que se necesitaron para el uso de scanner y cámara fotográfica en Linux.

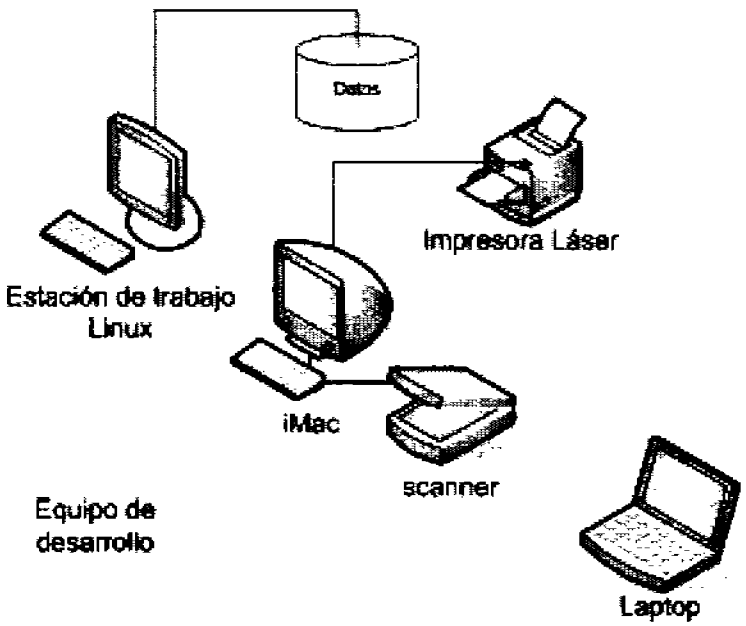


Ilustración 14.-equipo de trabajo

Por último se, ocupó una maquina portátil para la redacción de los documentos y probar el acceso a la aplicación desde cualquier punto. Esta maquina contó con un sistema operativo Windows preinstalado y además sirvió para revisar la prueba de visibilidad en los diferentes equipos y sus navegadores.

3.1.3 Enlace.

Configuración Física del Enlace

El enlace de desarrollo y pruebas es un servicio comercial adsl a 512 Mbps. conecta con el servidor Web y gobierna una red con un ruteador con punto de acceso inalámbrico.

Servicio.

El servicio esta disponible 24x24 cuenta con un ups powercom 570 con respaldo de energía para 15 min. Durante el desarrollo se habrán presentado algunas interrupciones en el servicio eléctrico, por lo que se tomó la iniciativa de conseguir ésta unidad de suministro de energía.

Configuración.

Configuración física.

- Gabinete Único. (MDF)
- Modem Speedstream 5200.
- Servidor L.A.M.P.
- Ruteador con punto de acceso inalámbrico (Linksys).
- Cableado horizontal (HCC) para estaciones de trabajo.

Configuración lógica.

- Estrella extendida (Ver figura)

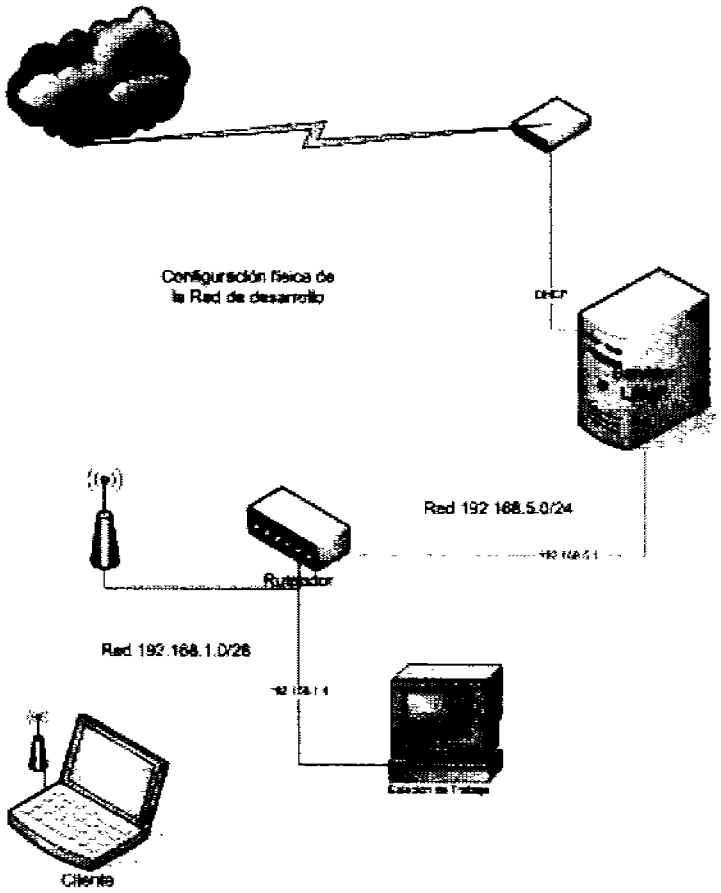


Ilustración 15.-Configuración física del enlace

Plan de pruebas.

(1) Es necesario identificar la cantidad de clientes concurrentes que el servidor web es capaz de atender.
(2) La facilidad de concentrar los recursos de software para el servidor.
(3) La velocidad de conexión.
(4) La compatibilidad entre los navegadores para la presentación de la aplicación.
(5) Compatibilidad de lenguaje del BDMS con el estándar ANSI para SQL.

Dispositivo	Prueba	Resultado	Valoración
SERVIDOR WEB	Conexiones concurrentes	500 (JMeter p. 66)	✔
SISTEMA OPERTIVO	FEDORA 3	Incluye LAMP en distribución	✔
	CENTOS 4.2	Incluye LAMP en distribución	✔
ENLACE [14]	Velocidad de subida	350 Kbps	✔
	Velocidad de bajada	190 Kbps	✔
BASE DE DATOS	Compatibilidad SQL	80 % sintaxis	✔
CLIENTES (HTML y JavaScript) Anexo I	Opera	Javascript 1.3, HTTP 1.1	✔
	MS Explorer	Javascript 1.6, HTTP 1.1, HTTP 1.1	✔
	Mozilla Firefox	Javascript 1.3, HTTP 1.1, HTTP 1.1	✔
	Safari	Javascript 1.3, HTTP 1.1, HTTP 1.1	✔

Tabla 6.- Resultado de las pruebas.

REFERENCIAS:

[1] http://ftp.ale.org/mirrors/fedora/linux/core/3/i386/iso/
[2] http://dev.mysql.com/downloads/
[3] http://www.php.net/downloads.php
[4] http://httpd.apache.org/download.cgi
[5] http://www.cecontec.com/proyecto/index.php/Portada
[6] http://www.cecontec.ath.cx/cvsroot
[7] http://www.phpmyadmin.net/home_page/index.php
[8] http://bluefish.openoffice.nl/index.html
[9] http://quanta.kdewebdev.org/
[10] http://www.gimp.org/
[11] http://gaphor.sourceforge.net/
[12] http://www.htmlhelp.com/tools/csscheck
[13] http://sunset.usc.edu/research/COCOMOII
[14] http://us.mcafee.com/root/speedometer/default.asp

3.2.- Modelo de desarrollo

De acuerdo al modelo de desarrollo de MOPROSOFT, Dentro de la Categoría de operación, encontramos la definición general del proceso concreto de "Administración de Proyectos Específicos".

Definiremos entonces:

3.2.1 Propósito

La realización del proyecto en tiempo y costo especifico, de acuerdo a los objetivos planteados.

3.2.2 Descripción

Desarrollo de un proyecto de Evaluación de Practicas de Laboratorio dentro del estándar L.A.M.P. mediante el uso de herramientas de código libre y el sistema colaborativo de desarrollo del open source.

3.2.3 Planeación

De acuerdo con el capítulo 2, la planeación se basa en las especificaciones de producto que, como se vió, consta de dos módulos principalmente, uno que interactúa con los usuarios de clase "profesores" y otro que atiende a los usuarios de clase "alumnos". Por su funcionalidad es necesario establecer a base de la información para su operación, es decir "las políticas de operación". Aquí también se generará la base del conocimiento en cuanto al proyecto en sí, de acuerdo al desarrollo y a los requerimientos del sistema.

Al tratarse de un equipo de desarrollo, se asignarán tareas específicas que permitan llevar a cabo el proyecto en el tiempo y forma convenidos.

Los requerimientos son expuestos en el apartado anterior de este capítulo, la responsabilidad de la Gestión del Proyecto (GES) asume la proveeduría material e intelectual de los recursos especificados. También establece las pruebas a que se deberá someter la

aplicación con la finalidad de que esta cubra las especificaciones del "cliente", de tal suerte que no quede duda alguna sobre la utilidad y funcionalidad de ésta y lo que se necesita.

3.2.4 Realización.

Aquí se lleva a cabo una minuciosa revisión de como se deberá presentarse la aplicación, que funcionamiento tendrá y como contendrá la información.

La presentación de un prototipo es necesaria para ir cohesionando cada idea dentro del componente con los requerimientos que debe cumplir. También se presentan la organización de los datos, los diagramas Entidad- Relación, diagramas de flujo de datos y la construcción de diagramas H que representan el diseño conceptual de la aplicación.

La construcción materializará a través de la codificación, integración de gráficos en cada componente, la implementación de la base de datos, el registro de actividades para mejores practicas, las pruebas de funcionalidad establecidas, etc... Los componentes del software probados según nuestro diseño.

3.2.5 Evaluación y control

Aquí se lleva a cabo una prueba integral de los componentes, haciendo uso de las herramientas de software para la verificación en la funcionalidad de nuestros componentes, haciendo una revisión completa en el cumplimiento de cada uno de los requerimientos que el cliente ha solicitado. Se suministra la configuración del entorno y de programa mismo con pruebas reales de ambas clases de usuario, el comportamiento de sistema y descartamiento de riesgos.

3.2.6 Cierre

Generación del:

- Manual de Usuario.
- Manual de operación.
- Manual de Mantenimiento.

3.2.7 Objetivos:

(1) Aplicar el sistema de desarrollo del open source dentro del modelo de procesos para la industria del software.

(2) Formular una Aplicación que sirva para la evaluación de prácticas de laboratorio a bajo costo y de utilidad en el ambiente académico y de capacitación de personal

(3) Mantener un control de cambios de acuerdo a lo observado por el director quien en cuyo caso actuaría como el "cliente" de acuerdo al modelo MoProSoft.

3.2.8 Indicadores

(4) (1) Aprobación del proyecto en base a una comparativa de costos representativa en cuanto al desarrollo de la aplicación con herramientas de Código Abierto.

(5) (2) Comprobar que el modelo (MoProSoft) es funcional en un desarrollo basado en Web y se puede integrar a la filosofía del Software de Código Abierto.

(6) (3) Cuantificación de cambios de acuerdo a las observaciones hechas durante cada ciclo de mejora.

3.3.- Prototipo del software (1)

3.3.1 Diseño de la aplicación

Es una parte de la preproducción del proyecto. Los prototipos son parte esencial para revisar las capacidades y habilidades del equipo de desarrollo, para consolidar la funcionalidad y creatividad al momento de la construcción de las ideas propuestas. En la etapa de desarrollo y mantenimiento del software se debe lograr que el producto de salida sea consistente con el producto de entrada a través de los controles de verificación, validación o pruebas.

Para la construcción de este prototipo, como se menciona en los capítulos anteriores, comenzará por la construcción de la interfaz del usuario, de acuerdo a nuestro modelo (MoProSoft) tenemos que:

Responsable	Actividad
RDM AN, DI	Distribución de las tareas al equipo de trabajo según su rol, según el plan de desarrollo
AN US DIU CL	Documentar y/o modificar las especificaciones del diseño. Creación del diagrama de hipervínculos. Realización de la propuesta de la interfaz de usuario según los procesos identificados en el análisis previo.
CL, RE	Validación del análisis y diseño de la interfaz de usuario.
RP	Preparación y realización de pruebas de navegación de acuerdo al seguimiento de los procesos de la aplicación
AN, DI	Reconocimiento del flujo de datos según el diagrama de procesos y la incorporación del diagrama de hipervínculos.
RE	Validación de las rutas de navegación de acuerdo al diseño basado en la interfaz y el diagrama de hipervínculos.

Tabla 7.- Plan de desarrollo, diseño y construcción capa 1 (responsables y actividades).

Diagrama de Hipervínculos.

El diagrama de hipervínculos se dividirá en dos, según el catalogo de procesos (tabla 3) para su mejor conceptualización, en módulo de profesores y los subprocesos derivados y módulo de alumnos de igual forma.

Ambos parten de un inicio común, el proceso de autentificación.

Profesores

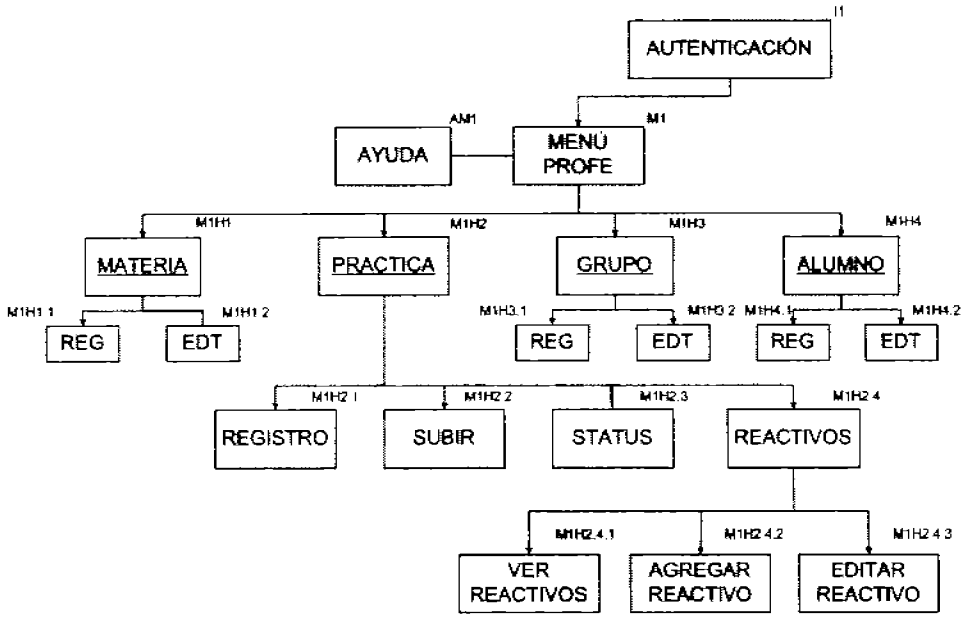


Ilustración 16.- Diagrama de hipervínculos (profesores).

Nomenclatura	Descripción	Proceso
I1	Pantalla de Autentificación	P1
M1	Pantalla del menú del profesor	
AM1	Ayuda descriptiva para el profesor.	
M1H1	Opción relacionada con las materias	
M1H1.1	Registro de materias y asignaturas	P4
M1H1.2	Edición de las asignaturas y materias	P4
M1H2	Opción relacionada con las prácticas.	
M1H2.1	Registro de prácticas en la db.	P5
M1H2.2	Subir los elementos de la práctica registrada.	P5
M1H2.3	Edición del estatus de la práctica registrada.	P5, P8
M1H2.4	Opción de reactivos para la practica seleccionada	P5, P6
M1H2.4.1	Listado de reactivos de la practica seleccionada	P6
M1H2.4.2	Registro de nuevos reactivos.	P6
M1H2.4.3	Edición de reactivos	P6
M1H3	Opción sobre grupos.	P3
M1H3.1	Registro de grupos	P3
M1H3.2	Edición de grupos	P3
M1H4	Opción sobre alumnos	P2
M1H4.1	Registro de alumnos	P2
M1H4.2	Edición de alumnos	P2

Tabla 8.- Descriptiva del diagrama de hipervínculos (profesores)

En la construcción del prototipo, el diagrama de hipervínculos es una referencia inicial que se vincula a cada proceso a partir de la tabla de descripciones.

Alumnos

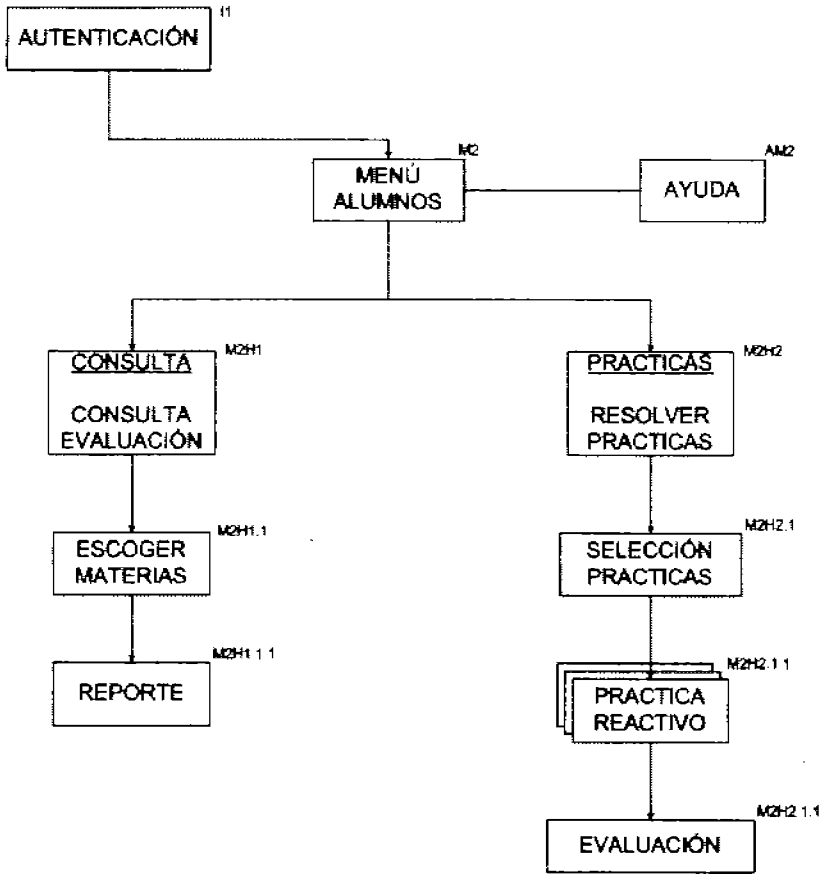


Ilustración 17.- Diagrama de hipervínculos (alumnos)

Nomenclatura	Descripción	Proceso
I1	Pantalla de Autentificación	P1
M2	Menú de la aplicación para los alumnos	
AM2	Ayuda en el menú para los alumnos	
M2H1	Hoja de consulta para las prácticas ya evaluadas con breve explicación.	P5
M2H1.1	Recoger, por las opciones, la(s) practicas(s) a reportear	P8
M2H1.1.1	Presentación en pantalla de las practicas escogidas (opción a impresión)	P8
M2H2	Hoja de consulta de practicas no evaluadas	P7
M2H2.1	Selección de práctica(s) para resolver.	P8
M2H2.1.1	Presentación de la practica y sus reactivos	P8, P9
M2H2.1.1.1	Presentación de la evaluación obtenida al terminar la solución de los reactivos.	P7, P11

Tabla 9.- Descriptiva del diagrama de hipervínculos (Alumnos)

El diagrama de hipervínculos inicial propone la estructura de navegación y permite ir encontrando nuevos módulos que mejoren el diseño del programa.

Interfaz de usuario

En la construcción de la capa de presentación es sin duda el elemento principal es la interfaz del usuario. Tratándose de una aplicación Web debemos hacer énfasis en el formateo de salida en los navegadores mediante el uso de hojas de estilo en cascada (CSS). Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Las ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad.
- En las páginas se puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

Para la especificación de la interfaz utilizamos Hojas de Estilo o CSS (Cascade Style Sheet) la cual nos permite darle formato a nuestra interfaz de usuario:

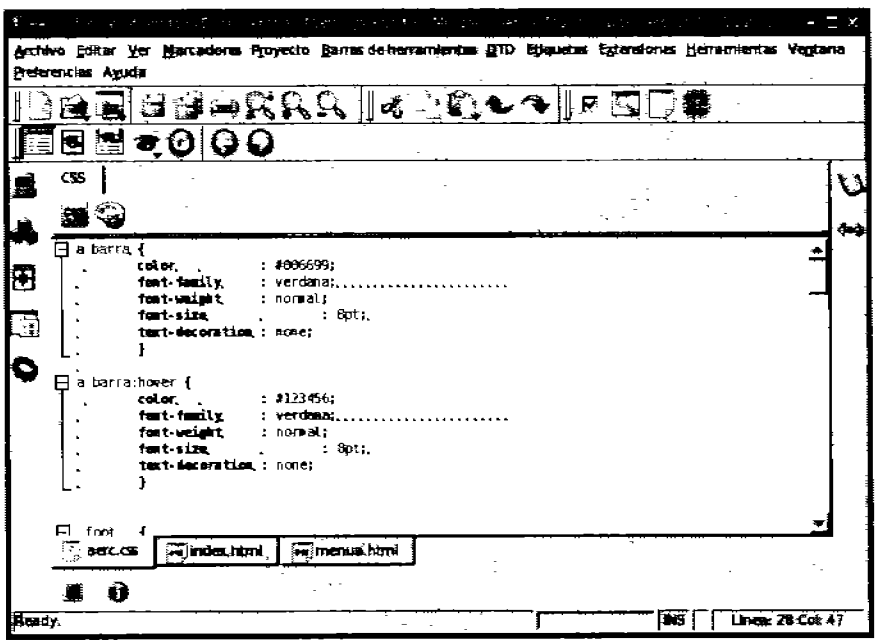


Ilustración 18.- Edición de la hoja CSS en el editor Quanta.

La generación de las vistas deben de ser siempre bajo las especificaciones que el cliente proporciona en las entrevistas con el diseñador de la interfaz y el analista, proporcionándole las características de cada elemento visible en la interfaz.

Elemento	Características
Tipografía del menú superior	Verdana, gris, normal 8 pts.
Tipografía del menú lateral	Verdana, negro, normal 10 pts.
Tipografía de los campos.	Predeterminada del navegador
Tipografía del pie de página	Verdana, tono azul, normal 8 pts.
Tipografía de la ayuda	Verdana, negro, normal 8 pts.
Cabecera	Imagen
Cuerpo principal	Fondo Blanco, Tipografía verdana, tono azul, normal 12 puntos. Títulos verdana 14 pts.

Tabla 10.- Ejemplo de tabla de resumen de características de la interfaz.

Como resultado obtendremos una vista que nos permita darle cohesión a la interfaz de la aplicación. Como ejemplo de vista de algunas pantallas de salida con la aplicación de la css tenemos:

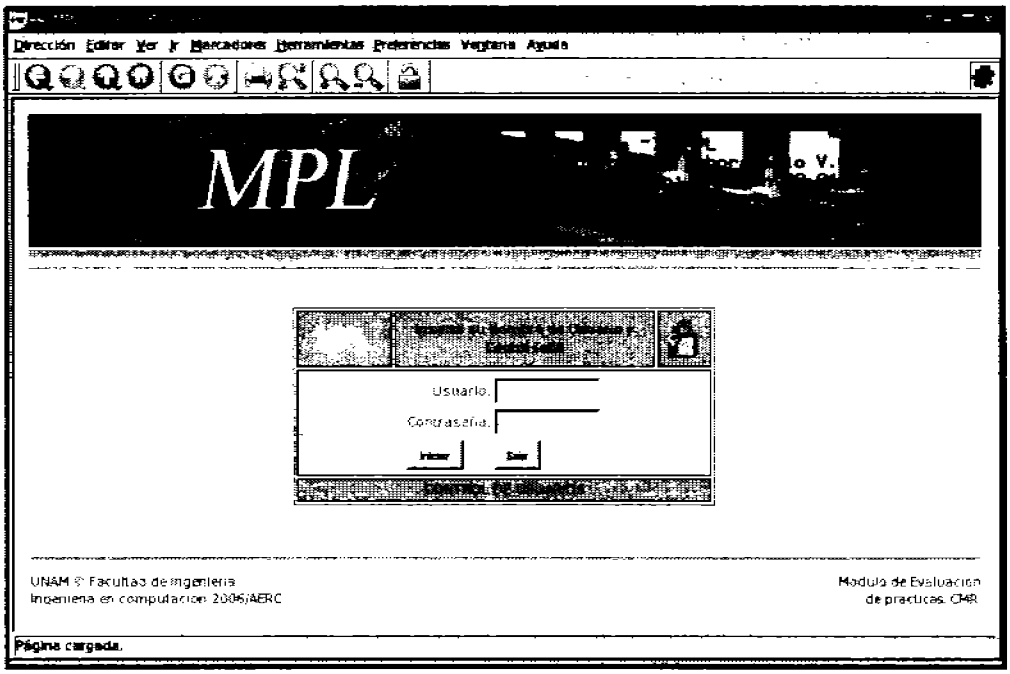


Ilustración 19.- Vista de la página de inicio desde el navegador.

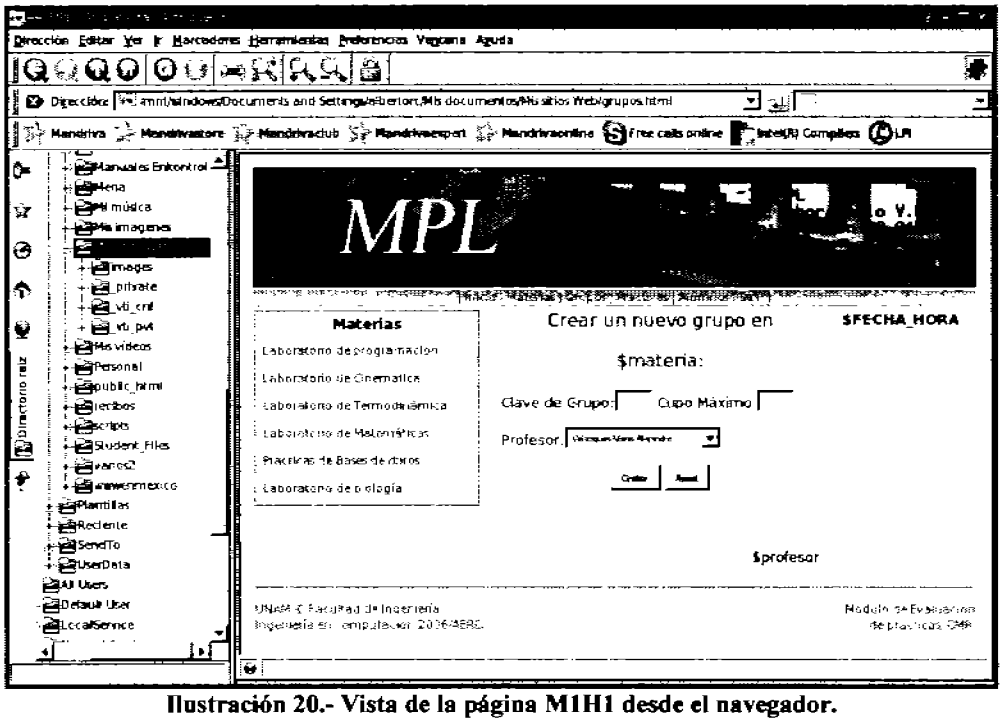


Ilustración 20.- Vista de la página M1H1 desde el navegador.

El diseñador de la interfaz además recoge todos los elementos gráficos proporcionados por el cliente para crea las imágenes de salida que se despliegan en cada página.

Con respecto a la capa de presentación, solo queda trabajar las restricciones de entrada, donde se validan los datos que ingresan los usuarios. Esto se realiza con JavaScript ver. 1.3, permitiendo que las validaciones funcionen con todos los navegadores previamente evaluados en la tabla de resultados (Tabla 6).

3.4.- Prototipo del software (2)

3.4.1 Estructura de la base de datos

Responsable	Actividad
RDM AN, DI	Distribución de las tareas al equipo de trabajo según su rol, según el plan de desarrollo
AN DI	Documentar y/o modificar las especificaciones del diseño. Creación del diagrama entidad relación a partir del modelo de bases de dato E-R.
RE	Validación del Diagrama entidad relación respecto a los datos requeridos en el diseño de la interfaz. Implementación de un registro de rastreo
PR	Creación de la base de datos según diseño del DER
RP	Preparación y realización de pruebas de funcionamiento y sintaxis del lenguaje SQL
RD	Redactar Entradas para la base de conocimiento

Tabla 11.- Responsables y actividades (capa de datos)

Diagrama de entidad-Relación (DER)

Este diagrama expresa las entidades relevantes para el proyecto, sus interrelaciones y propiedades. Es el Diseñador (DI) y el Analista (AN) quienes se encargan de esta otra parte de la capa del modelo, la capa de datos:

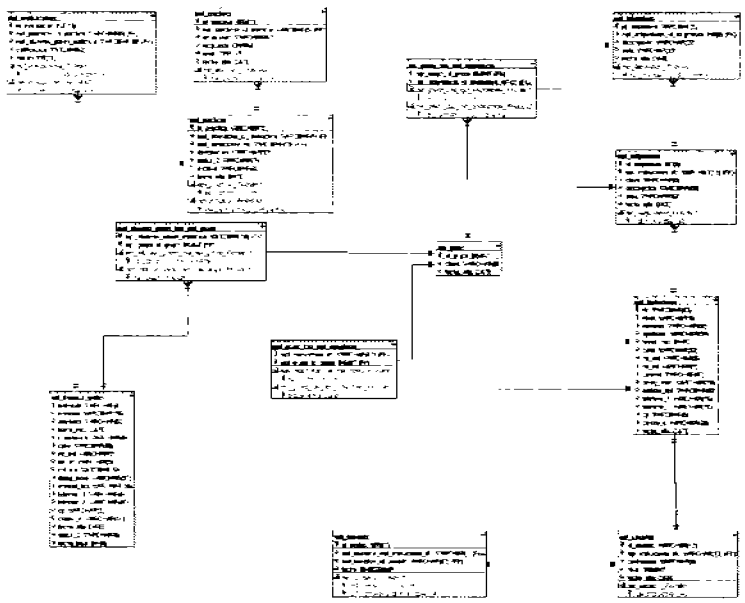


Ilustración 21.- Diagrama Entidad-Relación con DBDesigner

Integridad referencial.

El término de integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Si los contenidos de una base de datos son modificados con sentencias INSERT (insertar), DELETE (borrar) o UPDATE (actualizar), la integridad de los datos almacenados puede perderse, por ejemplo:

Podrían añadirse datos no validos a la base de datos, tal como una matricula de un alumno a un grupo que no existe. Por este motivo utilizamos la declaración explicita de llaves primarias y llaves foráneas, como restricciones de integridad.

Normalización

La importancia de la normalización se vera gratificada en la velocidad con la que el DBMS (Sistema Manejador de Bases de Datos) trabajará, sin embargo, al contener nuestra base de datos información explicativa y de nombres o lugares de archivos, en primera instancia se llevará hasta la segunda forma normal (2FN), y después se valorará el rendimiento de incrementar los catálogos o el uso de índices (3FN), que de inició basta en importancia según nuestro DER.

Primera forma normal

Una relación está en primera forma normal (1FN) si y sólo si todos los dominios son atómicos. Un dominio es atómico si los elementos del dominio son indivisibles.

Es decir, no tenemos grupos de repetición o un conjunto de valores asociados repetidos asociados a una misma tupla.

Segunda forma normal

Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y todos los atributos que no sean llaves dependen por completo de llave primaria.

Tercera forma normal

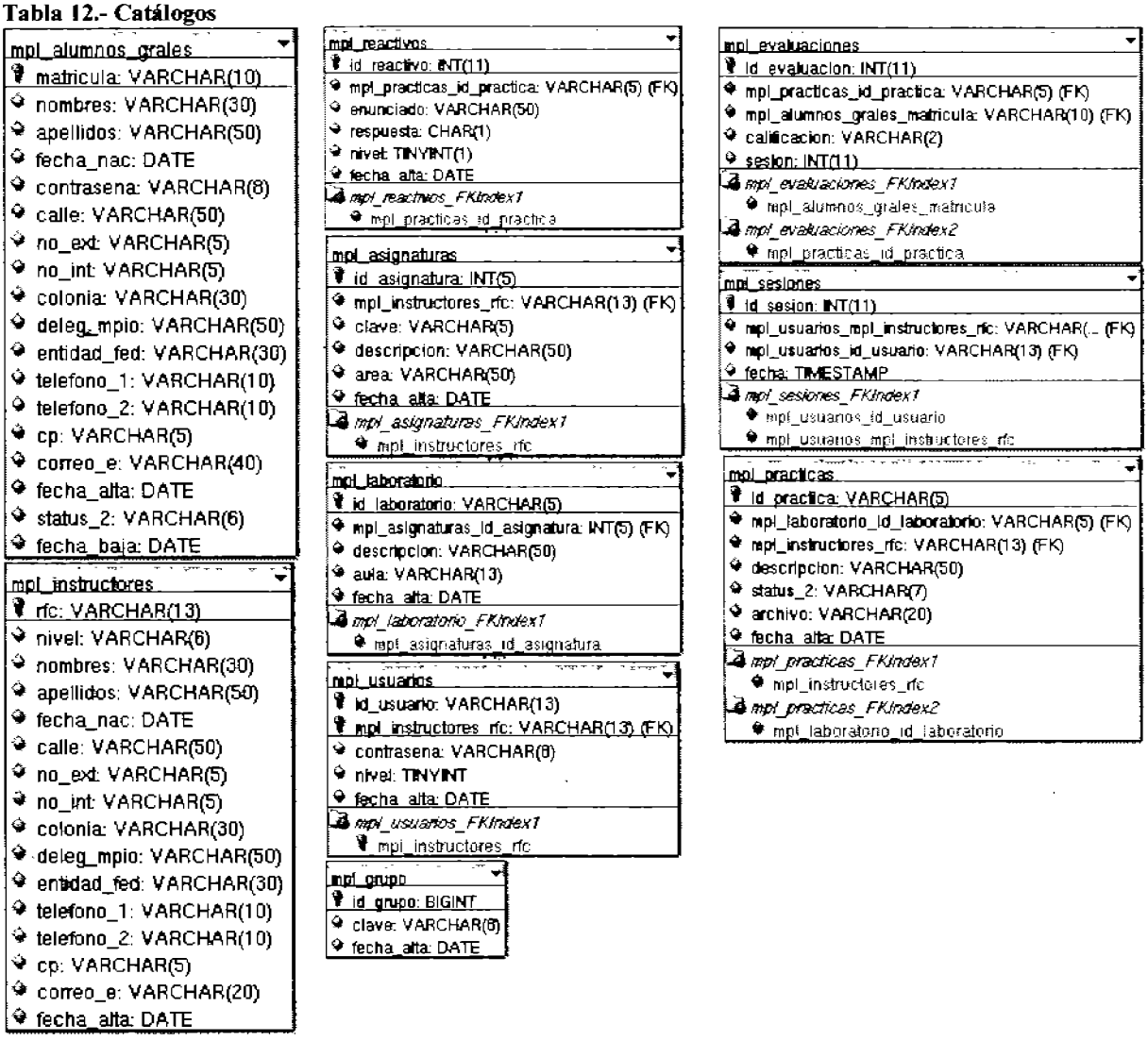
Una relación están en tercera forma normal (3FN) si y sólo si están en 2FN y todos los atributos no llave dependen de manera no transitiva de la llave primaria. Se dice que existe

una dependencia transitiva cuando tenemos el par de dependencias funcionales: R.1<- R.2 y R.2 <- R.3 porque de ellas trasciende que R.1 <- R.3.

De acuerdo a nuestro DER (I. 20), el modelo de entidad relación se propone de la siguiente manera:

= llave primaria
* = valor no nulo
0 = nulo.

Tabla 12.- Catálogos



Los catálogos junto con el DER producen el siguiente código:

Sesión MySQL

Primero se crea la base de datos:

```
# mysqladmin -u root create mpl
# mysql -u root mysql
mysql> INSERT INTO user \
(host,user,password,select_priv,insert_priv,update_priv,
delete_priv,create_priv,drop_priv,index_priv) \
VALUES ('localhost','mpl',PASSWORD('PaSsWoRd'),'Y','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
mysql> \q

# mysql -u docel mpl -p
```

Ingresamos el password y creamos las tablas (prototipo):

```
CREATE TABLE mpl_grupo (
  id_grupo BIGINT NOT NULL AUTO_INCREMENT,
  clave VARCHAR(8) NOT NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_grupo)
);

CREATE TABLE mpl_instructores (
  rfc VARCHAR(13) NOT NULL,
  nivel VARCHAR(6) NOT NULL,
  nombres VARCHAR(30) NOT NULL,
  apellidos VARCHAR(50) NOT NULL,
  fecha_nac DATE NOT NULL,
  calle VARCHAR(50) NOT NULL,
  no_ext VARCHAR(5) NOT NULL,
  no_int VARCHAR(5) NOT NULL,
  colonia VARCHAR(30) NOT NULL,
  deleg_mpio VARCHAR(50) NOT NULL,
  entidad_fed VARCHAR(30) NOT NULL,
  telefono_1 VARCHAR(10) NOT NULL,
  telefono_2 VARCHAR(10),
  cp VARCHAR(5) NULL DEFAULT 'NULL',
  correo_e VARCHAR(20) NOT NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(rfc)
);

CREATE TABLE mpl_alumnos_grales (
  matricula VARCHAR(10) NOT NULL,
  nombres VARCHAR(30) NOT NULL,
  apellidos VARCHAR(50) NOT NULL,
  fecha_nac DATE,
  contrasena VARCHAR(8) NOT NULL,
  calle VARCHAR(50) NOT NULL,
  no_ext VARCHAR(5) NOT NULL,
  no_int VARCHAR(5),
  colonia VARCHAR(30) NOT NULL,
  deleg_mpio VARCHAR(50) NOT NULL,
  entidad_fed VARCHAR(30) NOT NULL,
  telefono_1 VARCHAR(10) NOT NULL,
```

```
  telefono_2 VARCHAR(10),
  cp VARCHAR(5),
  correo_e VARCHAR(40),
  fecha_alta DATE NOT NULL,
  status_2 VARCHAR(6) NOT NULL DEFAULT 'ACTIVO',
  fecha_baja DATE,
  PRIMARY KEY(matricula)
);

CREATE TABLE mpl_asignaturas (
  id_asignatura INT(5) NOT NULL AUTO_INCREMENT,
  mpl_instructores_rfc VARCHAR(13) NOT NULL,
  clave VARCHAR(5) NOT NULL,
  descripcion VARCHAR(50) NOT NULL,
  area VARCHAR(50) NOT NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_asignatura),
  INDEX mpl_asignaturas_FKIndex1(mpl_instructores_rfc)
);

CREATE TABLE mpl_usuarios (
  id_usuario VARCHAR(13) NOT NULL,
  mpl_instructores_rfc VARCHAR(13) NOT NULL,
  contrasena VARCHAR(8) NOT NULL,
  nivel TINYINT UNSIGNED NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_usuario, mpl_instructores_rfc),
  INDEX mpl_usuarios_FKIndex1(mpl_instructores_rfc)
);

CREATE TABLE mpl_alumnos_grales_has_mpl_grupo (
  mpl_alumnos_grales_matricula VARCHAR(10) NOT NULL,
  mpl_grupo_id_grupo BIGINT NOT NULL,
  PRIMARY KEY(mpl_alumnos_grales_matricula, mpl_grupo_id_grupo),
  INDEX mpl_alumnos_grales_has_mpl_grupo_FKIndex1(mpl_alumnos_grales_matricula),
  INDEX mpl_alumnos_grales_has_mpl_grupo_FKIndex2(mpl_grupo_id_grupo)
);

CREATE TABLE mpl_grupo_has_mpl_intructores (
  mpl_instructores_rfc VARCHAR(13) NOT NULL,
  mpl_grupo_id_grupo BIGINT NOT NULL,
  PRIMARY KEY(mpl_instructores_rfc, mpl_grupo_id_grupo),
  INDEX mpl_grupo_has_mpl_intructores_FKIndex1(mpl_instructores_rfc),
  INDEX mpl_grupo_has_mpl_intructores_FKIndex2(mpl_grupo_id_grupo)
);

CREATE TABLE mpl_grupo_has_mpl_asignaturas (
  mpl_grupo_id_grupo BIGINT NOT NULL,
  mpl_asignaturas_id_asignatura INT(5) NOT NULL,
  PRIMARY KEY(mpl_grupo_id_grupo, mpl_asignaturas_id_asignatura),
  INDEX mpl_grupo_has_mpl_asignaturas_FKIndex1(mpl_grupo_id_grupo),
  INDEX mpl_grupo_has_mpl_asignaturas_FKIndex2(mpl_asignaturas_id_asignatura)
);

CREATE TABLE mpl_laboratorio (
  id_laboratorio VARCHAR(5) NOT NULL,
  mpl_asignaturas_id_asignatura INT(5) NOT NULL,
```

```
  descripcion VARCHAR(50) NOT NULL,
  aula VARCHAR(13) NOT NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_laboratorio),
  INDEX mpl_laboratorio_FKIndex1(mpl_asignaturas_id_asignatura)
);

CREATE TABLE mpl_sesiones (
  id_sesion INT(11) NOT NULL AUTO_INCREMENT,
  mpl_usuarios_mpl_instructores_rfc VARCHAR(13) NOT NULL,
  mpl_usuarios_id_usuario VARCHAR(13) NOT NULL,
  fecha TIMESTAMP NULL,
  PRIMARY KEY(id_sesion),
  INDEX mpl_sesiones_FKIndex1(mpl_usuarios_id_usuario, mpl_usuarios_mpl_instructores_rfc)
);

CREATE TABLE mpl_practicas (
  id_practica VARCHAR(5) NOT NULL,
  mpl_laboratorio_id_laboratorio VARCHAR(5) NOT NULL,
  mpl_instructores_rfc VARCHAR(13) NOT NULL,
  descripcion VARCHAR(50) NOT NULL,
  status_2 VARCHAR(7) NOT NULL DEFAULT 'ACTIVO',
  archivo VARCHAR(20) NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_practica),
  INDEX mpl_practicas_FKIndex1(mpl_instructores_rfc),
  INDEX mpl_practicas_FKIndex2(mpl_laboratorio_id_laboratorio)
);

CREATE TABLE mpl_evaluaciones (
  id_evaluacion INT(11) NOT NULL AUTO_INCREMENT,
  mpl_practicas_id_practica VARCHAR(5) NOT NULL,
  mpl_alumnos_grales_matricula VARCHAR(10) NOT NULL,
  calificacion VARCHAR(2) NOT NULL,
  sesion INT(11) NOT NULL,
  PRIMARY KEY(id_evaluacion),
  INDEX mpl_evaluaciones_FKIndex1(mpl_alumnos_grales_matricula),
  INDEX mpl_evaluaciones_FKIndex2(mpl_practicas_id_practica)
);

CREATE TABLE mpl_reactivos (
  id_reactivo INT(11) NOT NULL AUTO_INCREMENT,
  mpl_practicas_id_practica VARCHAR(5) NOT NULL,
  enunciado VARCHAR(50) NOT NULL,
  respuesta CHAR(1) NOT NULL,
  nivel TINYINT(1) UNSIGNED NULL,
  fecha_alta DATE NULL,
  PRIMARY KEY(id_reactivo),
  INDEX mpl_reactivos_FKIndex1(mpl_practicas_id_practica)
);
```

Ahora insertamos unos datos para hacer pruebas:

```
INSERT INTO `mpl_alumnos_grales` ( `matricula` , `nombres` , `apellidos` , `fecha_nac` ,
`contrasena` , `calle` , `no_ext` , `no_int` , `colonia` , `deleg_mpio` , `entidad_fed` ,
`telefono_1` , `telefono_2` , `cp` , `correo_e` , `fecha_alta` , `status_2` , `fecha_baja` )
VALUES (
'0303080122', 'ALBERTO ENRIQUE', 'RASO CANO', '1969-09-08', 'a1b3Rto.', 'ANTONIO CASO', '116', '4',
'SAN RAFAEL', 'CUAUHTEMOC', 'DISTRITO FEDERAL', '5525-5930', NULL , '06470',
'azul2412@prodigy.net.mx', '2006-11-18', 'ACTIVO', NULL
), (
'0303080123', 'CINTHIA GABRIELA', 'GUTIERREZ GARCIA', '1978-04-26', 'd0rM11OnA', 'MAR KARA', '24',
NULL , 'POPOTLA', 'MIGUEL HIDALGO', 'DISTRITO FEDERAL', '5527-9550', NULL , '11400',
'justgaby@hotmail.com', '2006-11-12', 'ACTIVO', NULL
);
```

Verificación 1

Para la primera verificación revisamos las tablas creadas:

```
mysql> show tables;
+----------------------------------+
| Tables_in_mpl                    |
+----------------------------------+
| mpl_alumnos_grales               |
| mpl_alumnos_grales_has_mpl_grupo |
| mpl_asignaturas                  |
| mpl_evaluaciones                 |
| mpl_grupo                        |
| mpl_grupo_has_mpl_asignaturas    |
| mpl_grupo_has_mpl_intructores    |
| mpl_instructores                 |
| mpl_laboratorio                  |
| mpl_practicas                    |
| mpl_reactivos                    |
| mpl_sesiones                     |
| mpl_usuarios                     |
+----------------------------------+
13 rows in set (0.00 sec)
```

Ahora revisamos las tablas:

```
[root@carmina ~]# myisamchk /var/lib/mysql/mpl/*.MYI
Checking MyISAM file: /var/lib/mysql/mpl/mpl_alumnos_grales_has_mpl_grupo.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_alumnos_grales.MYI
Data records:       2   Deleted blocks:       0
myisamchk: warning: 1 client is using or hasn't closed the table properly
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check record links
MyISAM-table '/var/lib/mysql/mpl/mpl_alumnos_grales.MYI' is usable but should be fixed

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_asignaturas.MYI
Data records:       0   Deleted blocks:       0
- check file-size
```

```
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_evaluaciones.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_grupo_has_mpl_asignaturas.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_grupo_has_mpl_intructores.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_grupo.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_instructores.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_laboratorio.MYI
Data records:       0   Deleted blocks:       0
```

```
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_practicas.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_reactivos.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_sesiones.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check record links

---------

Checking MyISAM file: /var/lib/mysql/mpl/mpl_usuarios.MYI
Data records:       0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check record links
```

Verificación 2

Revisamos los contenidos de las tablas con las que comenzaremos a probar nuestros scripts:

```
mysql> select nombres, apellidos, fecha_alta, status from alumnos;
+---------------+----------+--------------------+-------------+----------+
| matricula     | nombres  | apellidos          | fecha_alta | status_2 |
+---------------+----------+--------------------+-------------+----------+
| 0000303080123 | ALBERTO  | RASO CANO          | 2005-05-23  |  activo  |
| 0000303080132 | OSVALDO  | RODRIGUEZ ARGUELLO | 2005-05-13  |  activo  |
| 0000303080245 | GABRIELA | GUTIERREZ GARCIA   | 2006-02-14  |  activo  |
+---------------+----------+--------------------+-------------+----------+
3 rows in set (0.00 sec)

mysql> select rfc,nombres, apellidos, fecha_alta, nivel from profesores;
+---------------+----------+----------------+------------+-----------+
| rfc           | nombre   | apellidos      | fecha_alta |  nivel    |
+---------------+----------+----------------+------------+-----------+
| VEMA760306J1R | ALEJANDRO | VELAZQUEZ MENA | 1999-11-23 | PROFESOR  |
| PELM670406A5D | MARISELA  | PEDROZA LUNA   | 1997-11-23 | PROFESOR  |
+---------------+----------+----------------+------------+-----------+
2 rows in set (0.01 sec)


mysql>
```

Nos damos cuenta que el funcionamiento esta listo, faltará hacer una entrada en nuestra base del conocimiento:

BASE DE CONOCIMENTO

La instalación de la base de datos del DBMS se hace sin ninguna protección, por lo que inmediatamente después de instalarla, es preciso asegurar la cuenta de administración (también de nombre root) con un password.

```
[root@carmina practicas]# mysqladmin -u root password "CoNtRaSeÑa"
Es necesario usar las herramientas de chequeo y revisión para comprobar la
seguridad e integridad antes de comenzar a poblar nuestra base de datos.
myisamchk para hacer el diagnóstico.
myisamchk --recover para arreglar la consistencia de los indices en la base de
datos.
```

De aquí en adelante se usará al usuario docel para hacer inicio en la base de datos:

```
[root@carmina ~]# mysql -u docel -p practicas
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28 to server version: 4.1.12

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Es importante hacer mención sobre el trabajo de los tipos de datos en cuanto a las versiones del servidor, pues las entradas del tipo "BOOLEAN" no son aceptadas en aquellos anteriores a la versión 4, para esto se debe de utilizar

```
tinyint(1) NOT NULL default '0' .
```

3.5.- Prototipo del software (3)

3.5.1 Lógica del negocio (programación).

Análisis del flujo de datos

Responsable	Actividad
RDM AN, DI	Distribución de las tareas al equipo de trabajo según su rol, según el plan de desarrollo
AN DI, PR	Documentar y/o modificar las especificaciones del diseño. Creación del diagrama de flujo de datos..
RE	Análisis y validación del flujo de datos de la operación.
DI, PR	Seccionamiento en pequeños módulos y la programación de estos de acuerdo a la operación de la aplicación
RD	Redacción del funcionamiento de cada módulo identificado por proceso para formación del manual de usuario..
RE	Verificación del registro de rastreo .

Tabla 13.-Responsables y actividades (capa de negocio)

Después de la obtención de los datos a trabajar, y de contar con los diagrama de hipervínculos, es necesario establecer el flujo de datos que se van a revisar para entrar en la fase de programación de la lógica del negocio.

Los diagramas prototipos de esta fase están divididos de acuerdo a los "módulos" en los que se seccionó la realización de los diagramas de hipervínculos y de ahí se desprende el flujo de datos.

El revisor deberá encontrar las fallas dentro del flujo de datos o que por su lógica deban de tratarse de alguna forma distinta al diseño. Sin embargo los procesos automáticos (como el de evaluación) no se presentan al usuario, manifestando la independencia del desarrollo entre una capa y otra.

El flujo de datos nos permite a través de la observación y de su empate con el diagrama de procesos las secuencias a codificar para hacer la aplicación funcional de acuerdo a las especificaciones del cliente y entendible para el usuario. El funcionamiento en cada modulo deberá quedar comentado por el revisor documental (RD).

Diagramas de flujo

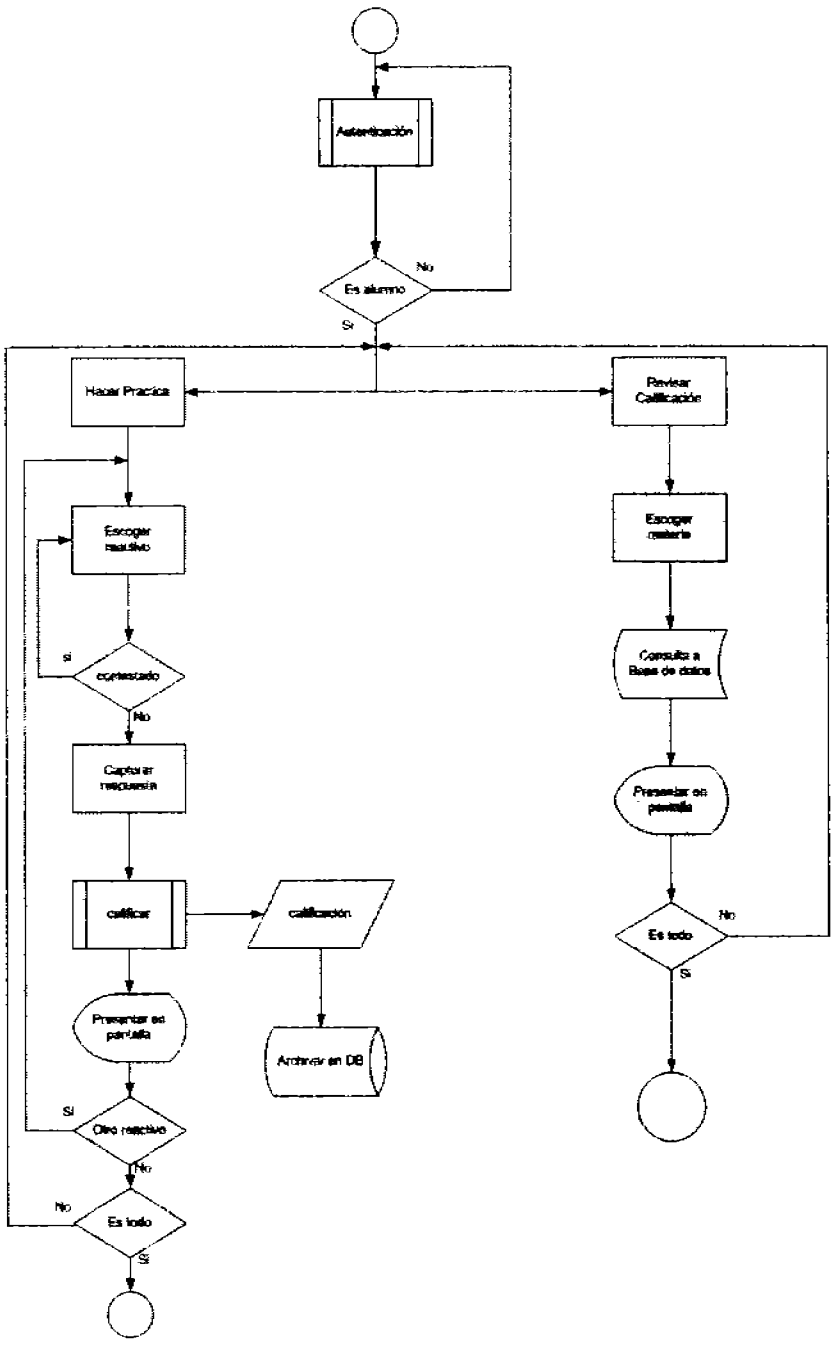


Ilustración 22.- Flujo de datos (alumnos).

Esta es en grandes trazos la operación que llevara a cabo el alumno al utilizar el módulo de evaluación de prácticas.

La operación de los procesos principales queda identificada y se pueden especificar con más detalle en una de ellas por ejemplo en el proceso de autentificación:

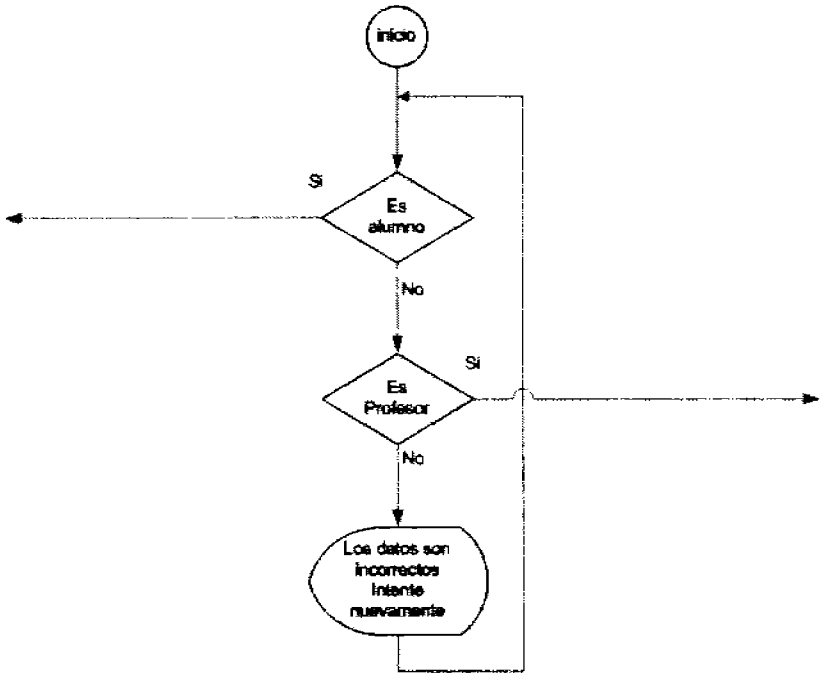


Ilustración 23.- Proceso de Autenticación

Se puede resumir que en cada triangulo de decisión se hace una consulta a la base de datos antes de determinar la salida en la pantalla del usuario. En el diseño de la interfaz aparecería como sigue:

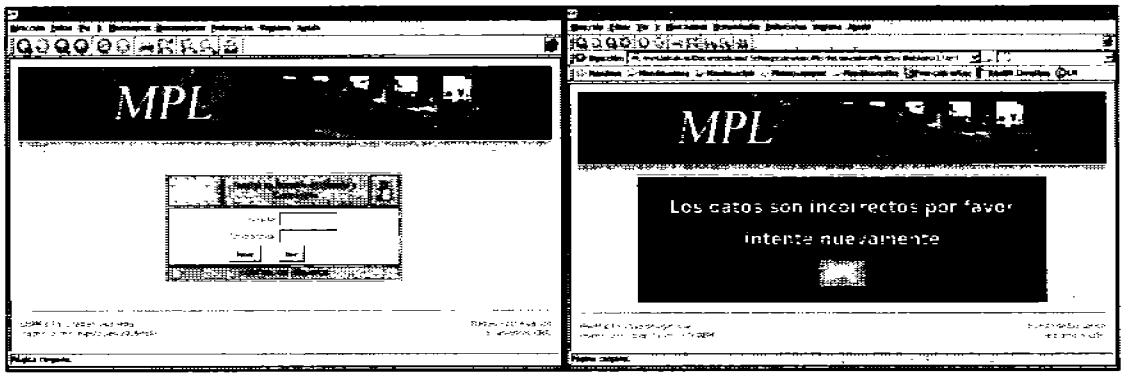


Ilustración 24.- Inicio

Ilustración 25.- Error

El código que usando php determinara la programación en cada paso del proceso viene también incluido en el cdrom, junto con los manuales y la aplicación, sin embargo y de manera ilustrativa, se imprime precisamente el de este pequeño proceso de ejemplo:

```
* Alberto Rasso Cano
* Formulario de Inicio
* Index.php
*****************************************************************************************************
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>MPL - Ingresa</title>
<meta name="keywords" content="Practicas, Evaluacion, CMR, Laboratorios">
<meta name="description" content="Sistema de evaluacion para practicas de laboratorio">
<link rel="stylesheet" type="text/css" href="aerc.css">
</head>
<body>
<table border="0" width="100%" id="table1">
  <tr>
        <td colspan="3" bgcolor="#000000">
        <p class="th" align="center">
        <img border="0" src="images/heading.png" width="631" height="116"></td>
  </tr>
  <tr>
        <td colspan="3" background="images/m_fdo.jpg" height="10">
        <p class="menu_jsd"> </td>
  </tr>
  <tr>
        <td width="15%">
         </td>
        <td width="69%">
        <div align="center">
<center>
<table border="0" cellpadding="0" cellspacing="0" background="imgs/logoiems1.jpg" width="100%" " height="75%">
 <tr>
  <td width="100%" height="100%"><CENTER>
<p class="cuerpo"> </p>
<TABLE borderColor=#0b4550 width=345 border=0>
 <TBODY>
 <TR>
  <TD bgColor=#FFCC66>
  <IMG
   src="images/logo1.png" width="74" height="42"></TD>
  <TD class=td2 bgColor=#FFCC66>Inserte su Nombre de Usuario y Contraseña</TD>
  <TD bgColor=#FFCC66>
  <IMG
   src="images/clave.gif" width="40" height="40"> </TD>
 <TR>
  <TD colSpan=3>
   <TABLE bgColor=#ffffff align="center">
    <FORM action="comprobacion.php" method=post>
    <TBODY>
    <TR>
```

```
    <TD class=td>Matricula:</TD>
    <TD><INPUT maxLength=10 size=5 name=matricula></TD></TR>
   <TR>
    <TD class=td>Contraseña:</TD>
    <TD><INPUT type=password maxLength=20 size=5 name=password></TD></TR>
   <TR>
    <TD><INPUT type=submit value=Iniciar name=iniciar></TD></FORM>
    <FORM>
    <TD><INPUT onclick=window.close() type=button value=Salir name=finalizar></TD></FORM></TR></TBODY></TABLE></TD></TR>
<TR>
 <TD class=td2 bgColor=#FFCC66 colSpan=3>CONTROL DE USUARIOS</TD></TR></TBODY></TABLE>
<p> </p>
  </CENTER></td>
 </tr>
</table>
</center>
</div>                          <td width="67%">
                <p class="textoA"> </td>
  </tr>
  <tr>
                <td colspan="3">
                <p class="foot"><hr>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
 <tr>
  <td width="50%"><font class="foot">UNAM
  &#8211; Facultad de Ingenier&iacute;a<br>
  Ingenier&iacute;a en computaci&oacute;n en 2006/AERC.</font></td>
  <td width="50%">
  <p align="right"><font class="foot">M&oacute;dulo de Evaluaci&oacute;n<br>
  de pr&aacute;cticas. CMR.</td>
 </tr>
</table></td>
  </tr>
</table>
</body>
</html>
<?
}
?>
```

* Alberto Rasso Cano

```
* Formulario de comprobacion
* comprobacion.php
*****************************************************************************************

<?
session_start();


$server="localhost"; /* server mysql */
$database="practicas"; /* base de datos */
$dopass="ContRaSeña"; /*password mysql */
$douser="docel"; /* user mysql */


$query_a="SELECT * FROM alumno WHERE matricula='$matricula'";

$link=mysql_connect($server,$douser,$dopass);
$result=mysql_db_query($database,$query_a,$link);

if(mysql_num_rows($result)==0){
echo "No existe la matricula introducida";
} else {

$array=mysql_fetch_array($result);
if($array["password"]==crypt($pass,"docel") ){

$SESSION["matricula"]=$matricula;
$SESSION["nombre"]=$array["nombre"];
$SESSION["apellidos"]=$array["apellidos"];
session_register("SESSION");
header("location: menua.php");

} else {
include(error1.php);
   }
}

?>
```

Terminado la incorporación del software, es en esta parte donde se redacta los procedimientos que generaran los manuales de usuario, en base a los documentos de operación de la aplicación y las anotaciones generadas después del plan de pruebas.

Capitulo 4

Documentación, Pruebas y Mantenimiento.

4.1.- Documentación.

Responsable	Actividad
RDM	Distribución de las tareas al equipo de trabajo según su rol, según el plan de desarrollo, elaboración de reportes de actividades.
RD	Redacción de los documentos de Instalación y operación (manual del usuario).
RDM	Redacción de las especificaciones de requerimientos, plan de pruebas y la parte correspondiente a la configuración del software del manual del usuario.
RE	Verificación del manual de usuario y Especificación de requerimientos
RM	Compilación de los documentos de salida y documentación del manual de mantenimiento, según reportes de verificación
RE	Reportes de verificación.

En realidad la documentación se va compilando desde el inicio, mientras se están definiendo las actividades (RAPE) y durante el análisis y diseño (AN y DI) de la aplicación se van generado registros de control y reportes de actividades. Son las bases para compilación documental que RD y RM formalizan y se pasan a revisión (RE).

De acuerdo con MoProSof, la generación compilada de documentación que se incorpora a la base del conocimiento es como sigue:

- *Especificación de requerimientos*
- *Plan de Pruebas del Sistema.*
- *Manual de Usuario. (propuesta)*
- *Análisis y diseño.*
- *Registro de rastreo.*
- *Plan de pruebas (integración).*
- *Componente(s)*
- *Registro de rastreo.(Evaluación unitaria)*
- *Manual de operación.*
- *Manual de Usuario*
- *Manual de Mantenimiento*
- *Reportes pruebas (de integración y del sistema).*
- *Reporte de actividades*
- *Lecciones aprendidas*
- *Reporte de verificación*
- *Reporte de validación*

4.1.1 Tipos de Documentación

La documentación que se entrega al cliente (CL) se divide claramente en dos categorías, interna y externa:

- Interna: (RDM, AN, PR, DI, RP) Es aquella que se crea en el mismo código, ya puede ser en forma de comentarios o de archivos de información dentro de la aplicación.

- Externa: (RD, RM) Es aquella que se escribe en cuadernos o libros, totalmente ajena a la aplicación en si. Dentro de esta categoría también se encuentra la ayuda electrónica.

Manual de operación

El manual de operación es la guía técnica o manual técnico que refleja el diseño del proyecto, la codificación de la aplicación y las pruebas realizadas para su correcto funcionamiento. Por lo general este documento esta diseñado para personas con conocimientos de informática, generalmente programadores.

El principal objetivo es el de facilitar el desarrollo, corrección y futuro mantenimiento de la aplicación de una forma rápida y fácil.

Este manual se compone por tres apartados claramente diferenciados:

- ***Cuaderno de carga:*** Es donde queda reflejada la solución o diseño de la aplicación. Esta parte del manual es únicamente destinada a los programadores. Se realiza de tal forma que permita la división del trabajo

  - *Lecciones aprendidas*
  - *Análisis y diseño.*
  - *Especificación de requerimientos*
  - *Plan de pruebas (integración).*
  - *Registro de rastreo.(Evaluación)*
  - *Lecciones aprendidas*
  - *Reporte de actividades*

- ***Programa fuente:*** Es donde se incluye la codificación realizada por los programadores. Este documento puede tener, a su vez, otra documentación para su mejor comprensión y puede ser de gran ayuda para el mantenimiento o desarrollo mejorado de la aplicación. Este documento debe tener una gran claridad en su escritura para su fácil comprensión.

  - *Código Fuente*
  - *Componente(s)*
  - *Registro de rastreo*

- ***Pruebas:*** es el documento donde se especifican el tipo de pruebas realizadas a lo largo de todo el proyecto y los resultados obtenidos.

  - *Plan de pruebas(sistema)*
  - *Plan de pruebas (integración).*
  - *Reportes pruebas del sistema*
  - *Reporte de actividades*
  - *Reportes pruebas de integración y del sistema).*
  - *Lecciones aprendidas*
  - *Reporte de verificación*
  - *Reporte de validación*

Manual de usuario

Contiene la información necesaria para que los usuarios utilicen correctamente la aplicación.

Este documento se compila desde el manual de operación pero se suprimen los tecnicismos y se presenta de forma que sea entendible para el usuario que no sea experto en informática.

Un punto a tener en cuenta en su creación es que no debe hacer referencia a ningún apartado del manual de operación y en el caso de que se haga uso de algún tecnicismo debe ir acompañado de un glosario al final de la misma para su fácil comprensión.

La guía de instalación

Es la guía que contiene la información necesaria para implementar la aplicación.

Dentro de este documento se encuentran las instrucciones para la puesta en marcha del sistema y las normas de utilización del mismo (notas de lanzamiento).

Dentro de las normas de utilización se incluyen también las normas de seguridad, tanto las físicas como las referentes al acceso a la información.

Como punto de observación, por norma (ISO/IEC 12207) debemos de encontrar dentro de los recursos que aporta nuestra base de conocimiento, el espacio necesario para la actualización de los manuales de mantenimiento, del usuario y de operación que son generados desde el lanzamiento del software.

Algunos de los puntos que no quedan explícitos de forma concreta en MoProSoft pueden ser complementados desde otros modelos (por ejemplo el de evaluación) o normas extendidas. Y aunque MoProSoft tiene muchas equivalencias con CMM v1.1 y ISO/IEC (15504, 12207, 9001:2000) al ser adaptativo, el desarrollo para aplicaciones Web, donde se requiere flexibilidad y rapidez, es una buena opción, si robusta, pero ágil

4.2.-Diseño de Mantenimiento

Dentro de la Ingeniería de Software el mantenimiento es un proceso en donde se pretende mejorar y optimizar el despliegue de programa, además de corregir defectos. Forma parte del ciclo de vida en el desarrollo del software, este se lleva a cabo después de la implantación (despliegue) del software ya en su etapa de producción.

Dentro del estilo de trabajo de la comunidad que desarrolla código abierto, suele trabajar con la documentación en línea (generalmente wikis), y cuando se trata de desarrollo formales para aplicaciones Web, las herramientas CASE son suplidas en su mayoría de las veces por editores sencillos de texto (a no ser que tratemos con aplicaciones JAVA) para la creación o modificación de scripts.

Los cambios implicados en la fase del mantenimiento son aplicados para la corrección de errores así como la expansión de funcionalidades tanto en su aplicación, como en su uso.

Para la aplicación del Módulo de Evaluación de Practicas, se inscribe dentro del circulo virtuoso (desarrollo en espiral), y como el resto del software de código abierto, el software es lanzado a veces con defectos conocidos porque Responsable de Desarrollo y Mantenimiento de Software (RDM) determina que tanto la utilidad como el valor del software en un cierto nivel de calidad compensa las carencias y defectos conocidos.

Las personas involucradas en la fase de mantenimiento de software esperan trabajar en estos defectos conocidos, ubicarlos y preparar un nuevo lanzamiento del software, conocido como un lanzamiento de mantenimiento, el cual resolverá los temas pendientes.

De acuerdo con varios autores, señalan que la etapa del mantenimiento del software es de entre todas la parte mas costosa dentro de su desarrollo y es por esto que el uso de herramientas de código abierto mas la filosofía del software libre (de ponerlo a disposición del público en general) disminuye sustancialmente el costo.

4.2.1 Tipos de mantenimiento

El mantenimiento para el Módulo de Evaluación de prácticas queda determinado según la norma ISO/IEC 12207 ya que MoProSoft no tiene un modelo de valoración explicito

- **Perfectivo**: Mejora del software (*rendimiento, flexibilidad, reusabilidad.*) o implementación de nuevos requisitos. También se conoce como mantenimiento evolutivo.
- **Adaptativo**: Adaptación del software a cambios en su entorno tecnológico (nuevo hardware, otro sistema de gestión de bases de datos, otro sistema operativo...)
- **Correctivo**: Corrección de fallos detectados durante la explotación.
- **Preventivo**: Facilitar el mantenimiento futuro del sistema (verificar precondiciones, mejorar legibilidad.).

Como se propone una visión de trabajo congruente a la filosofía del Software de código abierto, RDM establece como ciclos de lanzamiento, de acuerdo al tamaño de la aplicación un término de tres meses en promedio. Además se debe tener en cuenta durante el desarrollo del mantenimiento del software, las posibles migraciones ya sea de lenguaje y/o de entorno de operación, así como el retiro de éste cuando el ciclo llegue a su fin.

Los defectos conocidos durante el desarrollo son documentados por Revisor de Documentación (RD) quien se ha encargado de organizar y redactar en la base del conocimiento todas las consideraciones necesarias para la implantación de software, dentro de esas, se encuentran las notas de lanzamiento *(release notes)*, así los usuarios quedan advertidos sobre los defectos, permitiéndoles trabajar evitando las deficiencias documentadas. Después del lanzamiento del software se descubren nuevas deficiencias y tan pronto sean reportadas se ingresan en algún sistema de rastreo de errores.

4.2.2 Actividades

Las actividades comprendidas durante el proceso de mantenimiento son:

Responsable	Actividad
RDM	Implementación del proceso
AN y RE	Análisis y modificaciones de problemas
DI, AN, PR y RD	Implementación de Modificaciones
RP, RE y RM	Revisión y aceptación del mantenimiento
RDM, ET	Migración
RDM, RD y RM	Retiro

Tabla 14.- Proceso del mantenimiento

Las entradas se transforman (o consumen) por las actividades de mantenimiento para producir salidas. Los controles proporcionan una guía para asegurar que la actividad de mantenimiento produce salidas correctas.

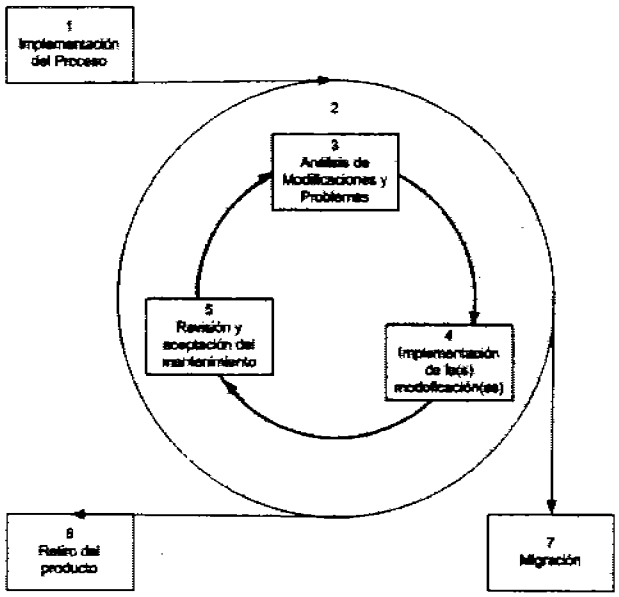


Ilustración 26.- Secuencia de actividades del mantenimiento.

El soporte (ISO/IEC 12207) identifica los procesos usados por las actividades de mantenimiento.

De entre las herramientas que se dispone para el seguimiento de errores en el desarrollo de software de código abierto destaca Bugzilla, la herramienta creada por la organización Mozilla.

También los foros de discusión son una herramienta en la que tanto usuarios como desarrolladores pueden verter dudas y encontrar respuesta, el RD consultará estas herramientas para mejorar los manuales y documentos que van sufriendo transformaciones de igual forma que la aplicación.

Las guías de ajuste y las verificaciones permiten mantener el control sobre los costos, para que estos se mantengan dentro del estimado. Se toman en cuenta los fijos generados por mantener un servidor con las herramientas en línea (como las ya mencionadas), las cuales ofrecen un gasto menor que las que derivarían de usar un equipo mantenedor privado con herramientas CASE de software comercial.

Como parte del inicio del mantenimiento, así como en la etapa de Análisis y desarrollo, al consulta de las lecciones aprendidas registradas en la base del conocimiento

4.3.- Pruebas

Responsable	Actividad
DI	Apoyo en el diseño del banco de pruebas de acuerdo a las especificaciones generadas via diseño de la aplicación.
AN	Supervisión y apoyo en la elaboración del banco de pruebas de acuerdo a la operación de la aplicación.
RP	Diseño del banco de pruebas sobre la operación del la aplicación.
RE	Aplicación de las pruebas según el plan propuesto por el RP.
CL	Validación del cumplimiento del banco de pruebas realizado por el RE.
RM	Documentación de los resultados para la generación de las notas de lanzamiento y el manual de operación
US	Aplicación de las pruebas de usabilidad y comprensión de la documentación generada, según el plan propuesto por el RP bajo la supervisión del RE.

Tabla 15.- Responsabilidades y actividad en la fase de pruebas.

En esta parte es importante resaltar la importancia de cómo tomar las decisiones para hacer las mejoras dentro del funcionamiento del software, si hay que afectar a una o mas capas de la arquitectura o de la configuración del servidor.

Ya en el capitulo 3 se analizó en un banco de pruebas el hardware propuesto y la compatibilidad del manejador de base de datos, en esta parte se proponen pruebas ya sobre la aplicación generada con el banco de pruebas propuesto por el Responsable de Pruebas (RP).

BANCO DE PRUEBAS

4.3.1 Instalación de la aplicación

a) Se descomprime el archivo tar.gz dentro del directorio designado en el servidor web, que en este caso se nombra como **mpl.**

```
[root@carmina ~]# tar -xzvf practicas.tar.gz /var/www/html/mpl
```

b) Verificamos la configuración del servidor para este directorio.

```
<Directory /var/www/html/mpl>
    Options None
    Options +Indexes
    Options +Includes
    AllowOverride None
    order deny,allow
    deny from all
```

```
    allow from 127.0.0.1 192.168.1.0/255.255.255.0
    AuthName "CMR - MPL"
    AuthType Basic
    AuthExternal pwauth
    require user docel
    Satisfy any
</Directory>
```

c) Comprobamos el estado del motor de base de datos.

```
[root@carmina ~]# mysqladmin status
Uptime: 7085  Threads: 1  Questions: 1  Slow queries: 0  Opens: 11  Flush tables:
1  Open tables: 0  Queries per second avg: 0.000
[root@carmina ~]#
```

d) Creamos la Base de datos y el usuario de ésta.

```
# mysqladmin -u root create practicas
# mysql -u root mysql
mysql> INSERT INTO user \
(host,user,password,select_priv,insert_priv,update_priv,delete_priv,create_priv,dr
op_priv,index_priv) \
VALUES ('localhost','docel',PASSWORD('PaSsWoRd'),'Y','Y','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
mysql> \q
# mysql -u docel practicas -p
mysql> \. /var/www/html/mpl/basedatos/practices.sql
```

4.3.2 Seguridad

La seguridad inherente a la configuración del servidor, la base de datos y el lenguaje son una parte importante, sin embargo la aplicación también debe estar sujeta a políticas que el cliente define de acuerdo al rol del usuario. Estas restricciones están sujetas a la observación del analista (AN) y el diseñador (DI) para que el responsable de pruebas (RP) determine el cumplimiento de dichas restricciones.

a) Login de usuario.- Comprobación de identidad. Funcionamiento adecuado de ingreso a la aplicación de acuerdo con el rol de usuario. Revisión de jerarquías en el ingreso a las pantallas de la aplicación.

En la pagina 100, se muestra el script *comprobación.php* que ilustra como el programa hace una consulta a la base de datos y determina:

1. Que el usuario existe en la base de datos.
2. Que la contraseña ingresada y la contraseña devuelta (y desencriptada) son la misma.

```
En base al diseño original, se contaba con solo dos roles de usuario. El
rol del profesor ó instructor haciendo tareas de administración, por lo
que en la fase de mantenimiento, para la tercera versión se hizo un
cambio estructural en la base de datos y la capa de aplicación,
agregando un rol mas, el de un usuario administrador, que se encargaría
de hacer las altas y bajas y demás tareas administrativas.
```

b) Administración de sesiones. Manejo de la temporalidad según la sesión. Cierre adecuado de la sesión.

Desde el script de comprobación (comprobacion.php) se maneja la sentencia php:

```
<?

session_start();
```

que asignará un número de sesión aleatorio durante todo el tiempo que el usuario permanezca dentro de la aplicación. Además se guardará en una tabla de la base de datos junto con el dato del tiempo (fecha y hora) del servidor y el usuario.

4.3.4 Manejo la interfaz

a) Visualización correcta de la interfaz.- Despliegue en el navegador de acuerdo a las hojas de estilo. Aplicación de cambio de elementos de la interfaz a través de éstas.

Estilo	Internet Explorer	Mozilla 1.5	Opera 9.0
azul.css	✔	✔	✔
verde.css	✔	✔	✔
sme.css	✔	✔	✔

Tabla 16.- Resultado de prueba de la visualización.

b) Facilidad de uso. Valoración de la ayuda proporcionada a través de la aplicación. Manejo intuitivo de los formularios. Revisión de las rutas de los hiperenlaces. Despliegue de comentarios.

Aplicación	Alumno	Profesor	Administrador
Ingreso	✔	✔	✔
Consulta	✔	✔	✔
Altas y bajas			✔
Registro de practicas		✔	
Despliegue de practicas	✔	✔	
Registro de reactivos		✔	
Despliegue de reactivos	✔	✔	✔
Edición de reactivos			
Resolución de reactivos	✔		
Despliegue de resultados	✔	✔	✔
Impresión de reportes	✔	✔	✔
inserción de avisos			✔
Cambio de temas (css)			✔

Tabla 17.- Resultado de facilidad de uso

c) Funciones de archivo.- Subida de prácticas a la aplicación. Registro correcto de archivos en la base de datos. Enlace veraz con los reactivos correspondientes. Despliegue adecuado en la pantalla del usuario, verificación de status.

Aplicación	Registro BD	Subida	Borrado	Edición Status
Practicas	✓	✓		✓
Reactivos	✓		✓	✓
Evaluaciones	✓			✓

Tabla 18 - Resultado de funciones de archivo

d) Funciones de operación.- Evaluación correcta de reactivos. Aparición aleatoria de reactivos en cada sesión de una misma practica. Resguardo correcto de resultados. Visualización de resultados.

Funciones Automáticas esenciales de la aplicación	
Formulación de reactivos (prueba)	✓
Evaluación de la prueba	✓

Tabla 19.- Resultado de las operaciones automáticas de la aplicación.

Conclusiones

El desarrollo de Software mediante la aplicación metodologías formales usadas en el desarrollo de aplicaciones tradicionales, pero con apoyo en las herramientas disponibles en el mundo del software de código abierto es una realidad indiscutible, que pueden lograrse con la misma calidad.

El despliegue adecuado y la funcionalidad permiten el trabajo proyectado, sin embargo, la ventaja de trabajar en el desarrollo de un proyecto de software de código abierto, permite que al poner a disposición del la comunidad desarrolladora de este tipo de software, el trabajo realizado entre mas velozmente en esa espiral virtuosa de mejoramiento y adaptación, corrigiendo errores y ausencia de funciones e integración que muchas veces el software propietario alcanza solo cuando la entidad emisora tiene los recursos suficientes para sostener dicha evolución.

El ahorro de recursos, sobre todo financieros, debido a uso de herramientas de Código Abierto, que en su mayoría están dentro de la categoría del software libre, ayuda a que las inversiones se realicen sobre todo en el hardware y la calidad del personal (capital humano).

Además de valorar la decisión por parte del cliente, de ofrecer dicha aplicación dentro del esquema de licenciamiento para código abierto, el mantenimiento puede verse disminuido en tiempo y costo mientras se incrementa la calidad según la capacidad de aprovechamiento de la aplicación para los usuarios.

En cuanto a la aplicación Módulo de Evaluación de Practicas, se destaca como ejemplo en el desarrollo como aplicación de código abierto bajo la adaptación de una metodología formal (Moprosoft) que pretende ser utilizada en los ambientes académicos y de capacitación de personal.

MoProSoft cabe dentro de las metodologías robustas, pero es flexible y ágil, de reciente aparición (2003), es una norma mexicana y además se desarrolló en la UNAM. El desarrollo de aplicaciones Web suele necesitar modelos de desarrollo no tan robustos, pero la

adaptabilidad, en el caso del Módulo de Evaluación de Practicas, fue esencial para su realización.

Considerado como módulo en su concepción IBT para trabajo en su forma autónoma no le excluye de poder modificarse y ser integrado a un LMS. En su evolución hacia esa posición, ha cambiado varias veces en una o más en las capas de su arquitectura, y seguramente lo seguirá haciendo, hasta que encuentre la forma de integrarse a alguna solución de este tipo.

La valoración de l software debe darse sobre todo por su utilidad, la razón de trabajar dentro de la filosofía del Open Source apuntala esta aseveración, sobre todo porque en este paradigma de desarrollo se apoya el uso de herramientas de la misma naturaleza, lo que disminuye los costos en este rubro a casi cero.

Fases principales	Software propietario US dlls.		Software Open Source ($0.00)
Análisis y Definición de Requisitos.	MS Office.	399.00	OpenOffice.
Especificación.	S.O. Windows 2003 web edition MSQL	400.00 3,700.00	S.O. CentOS 4.2 con apache MySQL
Diseño.	Rational rose ERwin	4,000.00 3,995.00	Graphor DBDesigner 4
Programación (escritura del código).	ASP .NET DreamWeaver	N/A 600.40	PHP Quanta
Prueba e instalación.	Visual Studio 2005 Team Test Load Agent	5,469.00	JMeter
Operación y mantenimiento.	AJAX + .NET Framework	N/A	Bugzilla + Blogs + BB phpMyAdmin

Tabla 20.- Comparativa económica de recursos de software.

La implantación de la aplicación ha sido bien acogida dentro del ámbito académico, ha tenido transformaciones al paso del tiempo mientras se recogen experiencias de los usuarios y/o el cliente, esta dentro de su etapa evolutiva, sufriendo cambios de adaptabilidad y perfeccionamiento.

El mantenimiento de la aplicación se lleva a cabo dentro de un ambiente controlado, de la misma forma que el resto de los proyectos de código abierto. En sus nuevas versiones,

seguirá así mientras el interés de cliente ó del público usuario perdure, cubriendo las necesidades específicas de entidades que apuesten su presupuesto por el capital humano.

Quizás MPL esté algún día a disposición en http://sourceforge.net como una aplicación abierta completamente y mantendrá un sitio donde resida la base de conocimiento, así como las demás herramientas que se ha utilizado en su mantenimiento, para que mantenga todas las características que los desarrollos de código abierto poseen.

Posiblemente todas las metodologías formales existentes sean aptas para la construcción de software libre mientras las herramientas de código abierto apoyen en cada fase de desarrollo y mantenimiento, sin embargo la parte mas importante, la voluntad humana será la que decida de que forma y que modelos aplicables tomarán rumbo al engrandecimiento del espíritu y conciencia del hombre.

Bibliografía

1. "RED HAT LINUX 9 UNLEASED", Bill Ball, Hoyt Duff, Sams Publishing, EUA 1993, 1002 p.
2. "Ingeniería de Software", Somerville Ian, Pearson Educacion, México 2002, 692 p.
3. "MySQL para Windows y Linux", Pérez López, César, AlfaOmega Grupo Editor, México 2004, 454 p.
4. "Creación de sitios web con php 4", Gil Rubio, Fco. Javier, McGraw-Hill, España 2001, 547 p.
5. "Navegar en Internet: Diseño de páginas web interactivas con Javascript y CSS, 3ª. Edición", Orós Cabello, Juan Carlos, AlfaOmega Grupo Editor, México 2002, 355 p.
6. "Web Project Management: Delivering susscessful comercial web sites", Friedlein, Ashley, Morgan Kaufmann Publishers, EUA 2001, 324 p.

Documentos:

7. "Modelo de Procesos Para la Industria del Software (MoProSoft Versión 1.1 mayo 2003), Oktaba, Hanna. Disponible en http://www.software.net.mx (NMX-I-059/02-NYCE-2005)
8. "ISO/IEC 12207:2002. AMENDMENT 1": Information Technology – Software Life Cycle Processes Amendment 1. Disponible en http://www.iso.org
9. Adaptación de las normas ISO/IEC 12207:2002 e ISO/IEC 15504:2003 para la evaluación de la madurez de procesos software en países en desarrollo Francisco J. Pino; Félix García; Francisco Ruiz; Mario Piattini. Disponible en http://www.ewh.ieee.org/reg/9/etrans/vol4issue2April2006/4TLA2_4Pino.pdf

Mesografía

http://es.wikipedia.org/wiki/Ingeniería_de_software

http://www.um.es/giisw/

http://www.fceia.unr.edu.ar/ingsoft/

http://yellospark.com/se_esp/

http://es.tldp.org/Presentaciones/200211hispalinux/robles/robles-ponencia-hispalinux-2002.pdf