UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de Acopio de Información en Línea de las Actividades Académicas de la Facultad de Ingeniería

TESIS PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A N :

JIMÉNEZ ROMÁN DAVID FRANCISCO
MATIAS MORALES EMA

DIRECTOR DE TESIS: ING. AURELIO SÁNCHEZ VACA

CIUDAD UNIVERSITARIA **2005**

Índice

Introducción

La estructura general que se siguió para realizar este trabajo de tesis, que lleva por nombre Sistema de Acopio de Información en Línea de las Actividades Académicas de la Facultad de Ingeniería (SAILFI), es la que se cita a continuación, describiendo brevemente el contenido de cada uno de los capítulos que lo conforman.

Capítulo 1. Consiste en una fundamentación teórica de la realización de este trabajo. Se menciona que objetivo se planteó para el mismo, que resultados y alcances se espera que tenga.

Capítulo 2. Se revisaron los requerimientos para la realización del sistema, desde el tipo de información que maneja, así como las herramientas que se utilizaron (sistema operativo, lenguaje de programación y manejador de base de datos) y las características principales del equipo de cómputo requerido para su implantación.

Capítulo 3. Es el diseño o planteamiento del sistema. Contiene los diagramas de flujo de datos de los procedimientos que éste realiza. Se hizo la elección del modelo de base de datos más apropiado y a partir de éste se elaboró el propio a utilizar en el sistema de acopio de información.

Capítulo 4. Se realizaron los pasos centrales del desarrollo del sistema. Se hace mención de los procesos que intervienen en su funcionamiento, de la pruebas que se llevaron a cabo en su desarrollo, así como de los aspectos requeridos para una correcta puesta en marcha o implantación.

Capítulo 5. Se refiere a procedimientos de mantenimiento para el sistema, revisiones que pueden hacerse para evitar fallas a futuro y principales medidas tanto preventivas como correctivas que se pueden llevar a cabo para mantener la calidad en el sistema.

Capítulo 6. Se citan los puntos más importantes para una correcta capacitación de quienes utilizarán el sistema y del tipo de soporte técnico que se les puede proporcionar en caso de ser necesario. Además se hace referencia de la ayuda en línea que los usuarios pueden consultar ante el surgimiento de alguna duda durante su utilización.

Capítulo 1

Marco teórico

Marco Teórico

Desde su surgimiento, las universidades se erigieron como espacios idóneos para la enseñanza, la reflexión y el desarrollo de nuevos conocimientos, por ello su función ha sido determinante para el avance dentro de la sociedad.

El propósito de la Universidad Nacional Autónoma de México es estar al servicio del país y de la humanidad de acuerdo con un sentido ético y de servicio social, superando constantemente cualquier interés individual. Sus fines se inspiran en los principios de libertad de cátedra y de investigación; en ella se acogen todas las corrientes de pensamiento y las tendencias de carácter científico y social.

Así, la esencia y la función docente de nuestra Universidad consiste en instruir, educar y formar individuos de alto nivel, competentes e informados, dotados de sentido social y conciencia nacional, que actúen con convicción para desarrollar una actividad fructífera en el medio en que han de prestar sus servicios.

De esta manera, la Universidad desarrolla tres funciones fundamentales: la docencia, la investigación y la extensión de la cultura. Estas tareas no pueden concebirse, ni cumplirse en forma correcta si no buscan su objetivo en las necesidades nacionales y si no repercuten en beneficio del desarrollo en México. Para realizarlas se apoya en las facultades, escuelas, institutos y centros de extensión.

1.1 Aspectos Generales

La Ingeniería, como actividad encauzada hacia la solución de problemas que aquejan al individuo y a la colectividad con base en el conocimiento de las leyes de la naturaleza, es tan antigua como el hombre mismo. Esta actividad creó en el México indígena, obras que aún hoy en día nos causan admiración y reverencia.

El desarrollo de México, en todos los órdenes, ha sido factible, en gran medida, por la labor de generaciones de ingenieros mexicanos conscientes de su responsabilidad, que han dado lo mejor de ellos mismos para coadyuvar a la creación de un país más justo y mejor dotado.

La Facultad de Ingeniería no sólo es la más antigua escuela de ingeniería de América, sino la primera institución de carácter científico del continente. Por otra parte, es reconocida como un centro formador de grandes ingenieros.

A lo largo de sus más de doscientos años, la Facultad de Ingeniería ha sido gestora y promotora de los avances científicos y tecnológicos de nuestro país. Desde sus inicios se ha preocupado por mejorar la preparación de sus egresados, por esta razón la enseñanza de la ingeniería se ha modificado, tanto en sus contenidos como en sus métodos, con objeto de estar acorde a las nuevas condiciones y formas de la actividad profesional, la docencia y la investigación.

La misión de Facultad de Ingeniería es formar integralmente profesionales en los niveles de licenciatura y posgrado, con los conocimientos, habilidades, actitudes y aptitudes que les permitan ejercer liderazgo en el campo profesional, la investigación y en la sociedad; así como satisfacer las necesidades del país y ser competitivos internacionalmente.

Además, genera acciones académicas para la actualización o profundización de los conocimientos de los profesores de ingeniería y mantiene con la más alta calidad el espacio académico para crear, transmitir, recuperar y preservar el conocimiento ingenieril, con el fin de transformar los productos del quehacer académico en factores que contribuyan a la solución de los problemas de la nación.

La Facultad de Ingeniería tiene entre sus objetivos primordiales:

Impartir educación superior a nivel licenciatura, especialización, maestría y doctorado en las diferentes ramas de la ingeniería, para contribuir a la formación de profesionales de la ingeniería que coadyuven al desarrollo nacional.

Realizar y difundir investigaciones sobre problemas de nivel nacional, que promuevan el desarrollo tecnológico y contribuyan a la actualización y especialización de profesionales en las distintas ramas de la ingeniería.

Promover actividades orientadas a un mayor acercamiento con el entorno social y cultural para lograr la educación integral de la comunidad de la facultad de Ingeniería.

Dentro de las atribuciones que tiene asignadas la Facultad de Ingeniería, están las siguientes:

- Revisión y actualización de los planes y programas de estudio que específicamente se han determinado y elaborado para obtener los grados de licenciatura en sus once carreras.

- Impartir cursos para obtener diplomas de especialización, grados de maestría y de doctorado, en las distintas ramas de la ingeniería.

- Mantener y fomentar las relaciones de intercambio académico con las dependencias universitarias y con otras instituciones afines tanto nacionales como extranjeras.

- Preparar conferencias, seminarios, exposiciones y cursos especiales, así como organizar y colaborar en congresos científicos nacionales e internacionales, relativos a disciplinas que se imparten en la Facultad.

- Preparar personal especializado en docencia e investigación en ingeniería para la propia Facultad y para otras instituciones del país.

- Prestar asesorías sobre problemas de ingeniería a organismos de los sectores público y privado.

- Planear, programar y supervisar el servicio social de los alumnos.

- Organizar programas de actualización y perfeccionamiento de las distintas ramas de la ingeniería.

- Difundir los aspectos generales y los conocimientos especializados de la ingeniería a través de publicaciones y diversos medios de comunicación.

- Organizar actividades de extensión académica y cultural.

1.2 Descripción de los objetivos

La Facultad de Ingeniería genera una gran cantidad de información relacionada con sus actividades académicas; por lo que resulta indispensable contar con un sistema de acopio de información de dichas actividades.

El objetivo del presente trabajo es diseñar y desarrollar un sistema de información automatizado, que permita recopilar información de manera organizada, estandarizada, oportuna y confiable de las diferentes actividades académicas que realiza la Facultad de Ingeniería para poder rendir informes que son solicitados por la propia facultad y por otras dependencias de la UNAM.

1.3 Metas y ámbito del sistema

Con la implementación de este sistema, se espera tener los siguientes resultados:

- Contar con la información, confiable y oportuna de las actividades académicas que se desarrollan en la Facultad de Ingeniería.

- Generar información útil para el informe anual de actividades presentado por el Director de la facultad.

- Concentrar la información de dichas actividades, que puede ser requerida por diversas instancias de la UNAM.

- Tener disponible la información de los proyectos que se desarrollan en la facultad.

- Elaboración de un resumen estadístico del diferente material didáctico que desarrolla el personal académico.

- Facilitar la captura de toda la información requerida por la facultad, con una interfaz amigable al usuario

- Disminuir la carga de trabajo que todo lo anteriormente citado representa al tener que hacerse hasta el momento que es requerido. Se pretende que el acopio de la información pueda realizarse prácticamente durante todo el año por parte de los usuarios contemplados para el sistema.

Capítulo 2

Requerimientos del sistema

Requerimientos del sistema

2.1 Análisis del tipo de información que se va a procesar

Actualmente el sistema que maneja toda la información académica de la Facultad de Ingeniería esta diseñado en un ambiente de trabajo de Microsoft Access. Dicho sistema consiste en un menú que contiene los distintos tipos de actividades y/o información que maneja y requiere la propia Facultad.

En la figura 2.1.1 se pueden ver las distintas opciones que contiene el menú principal de este sistema para la recopilación de la información.

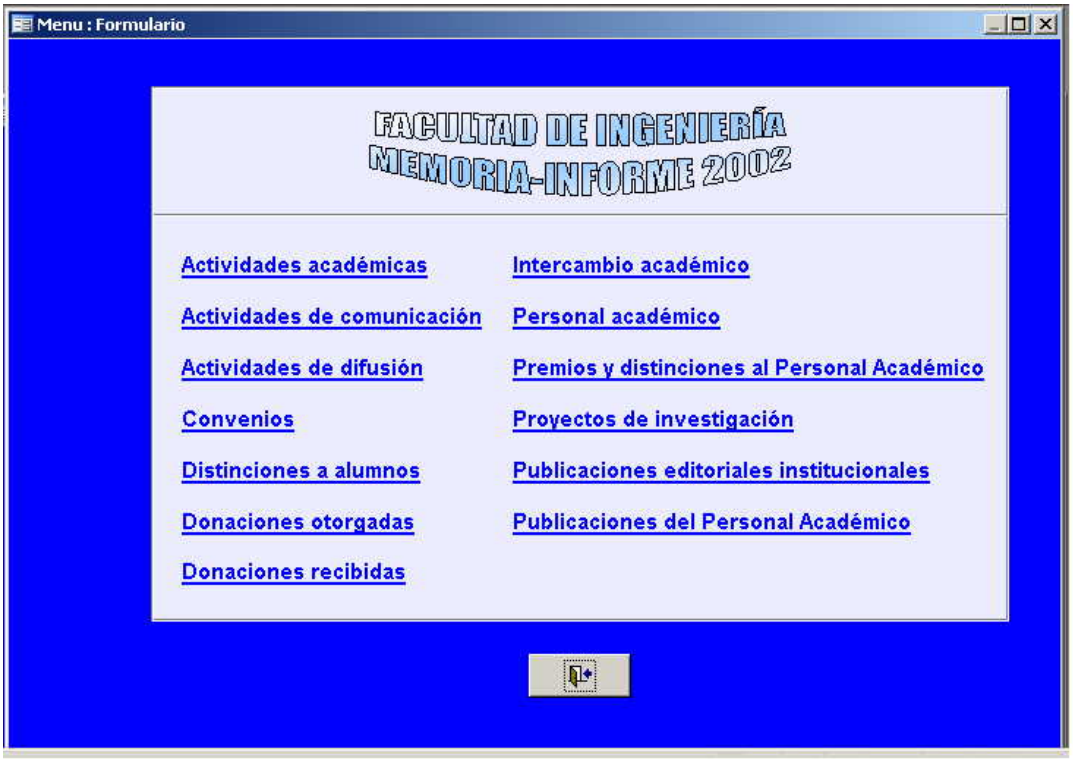


figura 2.1.1 Menú de la interfaz en Access

Aquí se puede apreciar que, las actividades dentro de la Facultad están clasificadas en tres tipos distintos que son: Actividades académicas, Actividades de comunicación y Actividades de difusión.

Así mismo, además de estas actividades también es posible acceder a información relacionada con Convenios, Donaciones tanto otorgadas como recibidas por la Facultad, Premios y distinciones otorgados, así como Premios y distinciones recibidos por el personal académico.

Se cuenta también con un apartado disponible para la información concerniente con productos editoriales. Esta opción presenta a su vez un submenú de opciones para los distintos productos editoriales que se pueden tener: Anales, Boletines, Carteles, Material cartográfico, Catálogos, Cuadernos, Folletos, Informes y reportes, Libros, Material audiovisual, Material de apoyo docente, Memorias, Revistas Arbitradas y no arbitradas, así como Software.

Las últimas opciones del menú principal corresponden a los Proyectos de Investigación, además de un vínculo hacia Publicaciones del personal, las cuales incluyen Artículos en memorias, Artículos en periódico, Artículos en revistas, Capítulos en libros, Catálogos, Informes y reportes, Introducción, Libros, Material de apoyo docente y Prólogos.

Para todas las actividades o categorías citadas con anterioridad se manejan diferentes campos para el registro de la información, siendo comunes a todas ellas dos campos: uno para la División de la Facultad de Ingeniería a la que corresponde la actividad, las cuales son: DCB (División de Ciencias Básicas), DCSH (División de Ciencias Sociales y Humanidades), DEC (División de Educación Continua), DEP (División de Estudios de Posgrado), DICT (División de Ingeniería en Ciencias de la Tierra), DICTyG (División de Ingeniería Civil, Topográfica y Geodésica), DIE (División de Ingeniería Eléctrica), DIMEI (División de Ingeniería Mecánica e Industrial), SG (Secretaría General) y otro para el Departamento al que pertenece dentro de dicha división. Los Departamentos pueden variar y en cada caso dependerán de la División de la que se trate.

El tipo de información que se procesa es diferente en cada actividad teniendo sólo en común los mencionados anteriormente. Así, para las *actividades académicas* la información que se requiere es el tipo de actividad (Coloquio, Concursos, Conferencias, Congresos, Cursos, Encuentros, Foros, Seminarios, Simposio, Talleres), el nombre de la actividad, la sede donde se desarrolla, las fechas de inicio y de término, el nombre del ponente, el número de sesiones y la asistencia.

Para las *actividades de comunicación* los datos que se requieren son: el tipo (Cápsula, Programa, Promocional) y el nombre de la actividad, sus fechas de inicio y término, la duración, las emisiones y el medio (radio/TV).

Las *actividades de difusión* incluyen el tipo de actividad (Concierto, Exposición, Feria, Festival, Jornada, Lectura, Mesa redonda, Muestra, Obra de danza, Obra de teatro, Obra fílmica y/o video, Performance, Presentación de publicaciones o Visita guiada), su nombre y sede, sus fechas, asistencia y recinto.

La información de interés en *convenios* es el nombre del convenio, su objetivo, su vigencia, las fechas de inicio y de término, además de especificarse el tipo de renovación (tácita / no existe) y el tipo de convenio (específico o general).

Tanto en las *donaciones otorgadas* como en las *donaciones recibidas* se debe incluir la institución que otorga o recibe (según sea el caso), el monto de la donación, la fecha de ésta y si fuera en especie se debe especificar.

En el caso de los *premios y distinciones* (otorgados o recibidos) se requiere el nombre del premio o distinción, el nombre y RFC del académico que recibe el premio, la fecha en que se otorgó, así como la institución beneficiada o que otorga el premio.

En cuanto a los *productos editoriales*, se tiene para los anales y para las revistas: título, nombre del autor, de los colaboradores y de la editorial, ciudad y año de la edición, volumen, número, número de páginas, tiraje, ISSN, periodicidad (anual, semestral, cuatrimestral, trimestral, bimestral, mensual, quincenal, semanal, indefinida, irregular u otra), y si es o no arbitrado.

Los boletines incluyen título, nombre del autor y de los colaboradores, el año de edición, la fecha de publicación, el tipo de periodicidad (mismos tipos que en anales), número de páginas, editorial, tiraje, número, volumen, así como la ciudad de edición.

Para los carteles la información requerida es nombre del autor y de los colaboradores, el título, la actividad que promueve, su tiraje y año de publicación.

En el material cartográfico se necesita la siguiente información: nombre del primer autor y de los colaboradores, el título, las escalas 1 y 2 y la fecha de elaboración.

Se cuenta además con catálogos, los cuales presentan título del catálogo, nombre de su autor y de los colaboradores, lugar de edición, la editorial, número de páginas, volumen, número, tiraje, año de publicación y el ISSN.

Otro de los productos editoriales que se manejan son los cuadernos, de los cuales se requiere nombre del primer autor y, si existen, de los otros autores, título, edición, número de páginas, tiraje, año de publicación y el nombre de la colección si es el caso.

Se tienen además folletos con título, nombre del autor y de sus colaboradores, editorial, año de edición, número de páginas, tiraje, número y ciudad de edición.

En la opción de informes y reportes los datos son nombre del autor y de los colaboradores, título, número de páginas y año de edición.

Los libros requieren que se especifiquen nombre del autor y cuando existan, de los otros autores, título, la edición y editorial, lugar de edición, número de páginas,

tiraje, año de publicación, ISBN, colección (si aplica) y estatus del libro (porcentaje de avance).

Para el material audiovisual son necesarios su título, nombre del autor y de los colaboradores, año de edición, tipo y duración (en minutos).

Son requeridos el nombre del primer autor y de los colaboradores, el tipo de material, ciudad de edición, editorial, el número de páginas, tiraje, año de publicación, ISBN, edición, colección (si es el caso), para el material de apoyo docente, además de especificar a quien está dirigido.

En las memorias se debe incluir el título, nombre del autor y de los colaboradores, ciudad de edición, editorial, año de publicación, volumen, número, tiraje y número de páginas.

La última opción de los productos editoriales es el software. Este requiere nombre del autor y de los colaboradores, título, nombre del lenguaje en el que fue programado, la versión y el tamaño (especificado en KB).

En el caso de los *proyectos de investigación* los datos estipulados son: nombre del proyecto, línea de investigación, nombre, número de trabajador y RFC del responsable del proyecto, nombre de los colaboradores, campo de la ciencia, fuente de financiamiento, tipo de actividad, estatus, año de inicio, número de alumnos participantes (bachillerato, licenciatura y posgrado) y comentarios.

En las *publicaciones del personal* se encuentran artículos en memorias, artículos en periódicos, artículos en revistas, capítulos en libros, catálogos, informes y reportes, introducción, libros, material de apoyo docente y prólogos.

En los artículos en memorias, artículos en periódico y artículos en revistas se tienen campos comunes que son: nombre del autor y de los colaboradores, título del artículo, año de edición y los números de páginas inicial y final. Además de lo anterior requieren, los artículos en memorias: nombre formal de la memoria, volumen y número; los artículos en periódico: fecha de publicación, nombre del periódico y la época; los artículos en revistas: volumen, número, nombre formal de la revista y el tipo de revista (arbitrada / no arbitrada).

Los capítulos en libros requieren nombre del autor del capítulo y de sus colaboradores, título del capítulo, título del libro, edición, año, lugar de edición, tiraje, editorial, ISBN y colección (si es el caso).

La información a incluir en catálogos, informes y reportes (especificando ahora si es informe o reporte) y libros es igual que la de estos mismos apartados descritos anteriormente en los *productos editoriales*.

Para la introducción se necesita el nombre del autor y de los colaboradores, título de la introducción y del libro donde se publica, edición, año, lugar de edición, páginas inicial y final, tiraje, ISBN, editorial y colección (si aplica).

Cuando se trate de material de apoyo docente los datos serán: título, autor, colaboradores, editorial, edición, colección, número de páginas, tiraje, ISBN y año.

La última opción en las publicaciones del personal es el prólogo. El tipo de información en éste es: autor, colaboradores, título del prólogo y del libro, edición, ciudad de edición, editorial, páginas inicial y final, tiraje, año de publicación, ISBN y colección (si es el caso).

2.2 Descripción de las herramientas de software a utilizar.

El software es el conjunto de instrucciones que las computadoras emplean para manipular datos. Sin el software, una computadora sólo puede ser vista como un conjunto de medios sin utilidad. Con el software, una computadora puede almacenar, procesar y recuperar información.

En un sentido más amplio, el software es un conjunto de programas, documentos, procedimientos y rutinas asociados con la operación de un sistema de cómputo, distinguiéndose así de los componentes físicos llamados hardware.

Comúnmente a los programas para computadora se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opere con eficiencia, este adecuadamente documentado, y suficientemente sencillo de operar. Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados.

El hardware por si solo no puede hacer nada, pues es necesario que exista el software. Desde ese punto de vista, el software es el conjunto de instrucciones que hacen funcionar al hardware.

Clasificación del software

El software para computadora puede clasificarse en general en tres clases: Sistemas operativos, Lenguajes de programación y Software de uso general o Software de aplicación.

Sistemas operativos

El sistema operativo es el gestor y organizador de todas las actividades que realiza la computadora. Marca las pautas según las cuales se intercambia información entre la memoria central y la externa, y determina las operaciones elementales que puede realizar el procesador. El sistema operativo debe ser cargado en la memoria central antes que ninguna otra información.

Un Sistema Operativo (SO) es en sí mismo un programa de computadora. Sin embargo, es un programa muy especial, quizá el más complejo e importante en una computadora. El SO “despierta” a la computadora y hace que reconozca a la CPU, la memoria, el teclado, el sistema de vídeo y las unidades de disco.

Además, proporciona la facilidad para que los usuarios se comuniquen con la computadora y sirve de plataforma a partir de la cual se corren programas de aplicación.

Cuando se enciende una computadora, lo primero que ésta hace es llevar a cabo un autodiagnóstico llamado autoprueba de encendido (Power On Self Test, POST). Durante la POST, la computadora identifica su memoria, sus discos, su teclado, su sistema de vídeo y cualquier otro dispositivo conectado a ella. Lo siguiente que la computadora hace es buscar un SO para arrancar (boot).

Una vez que la computadora ha puesto en marcha su SO, mantiene parte de éste en su memoria en todo momento. De esta forma, mientras la computadora esté encendida, el SO tiene 4 tareas principales:

1. Proporcionar ya sea una interfaz de línea de comando o una interfaz gráfica al usuario, para que este último se pueda comunicar con la computadora. Interfaz de línea de comando: se introducen palabras y símbolos desde el teclado de la computadora, ejemplo, el MS-DOS. Interfaz gráfica del Usuario (GUI), se seleccionan las acciones mediante el uso de un Mouse para pulsar sobre figuras llamadas iconos o seleccionar opciones de los menús.

2. Administrar los dispositivos de hardware en la computadora. Cuando corren los programas, necesitan utilizar la memoria, el monitor, las unidades de disco, los puertos de Entrada/Salida (impresoras, módems, etc). El SO sirve de intermediario entre los programas y el hardware.

3. Administrar y mantener los sistemas de archivo de disco. Los SO agrupan la información dentro de compartimentos lógicos para almacenarlos en el disco. Estos grupos de información son llamados archivos. Los archivos pueden contener instrucciones de programas o información creada por el usuario. El SO mantiene una lista de los archivos en un disco y proporciona las herramientas necesarias para organizar y manipular estos archivos.

4. Apoyar a otros programas. Otra de las funciones importantes del SO es proporcionar servicios a otros programas. Estos servicios son similares a aquellos que el SO proporciona directamente a los usuarios. Por ejemplo, listar los archivos, grabarlos a disco, eliminar archivos, revisar espacio disponible, etc. Cuando los programadores escriben programas de computadora, incluyen en sus programas instrucciones que solicitan los servicios del SO. Estas instrucciones son conocidas como "llamadas del sistema".

Software de uso general (software de aplicaciones)

El software de uso general ofrece la estructura para un gran número de aplicaciones. Está diseñado y escrito para realizar tareas específicas personales, empresariales o científicas como el procesamiento de nóminas, la administración de los recursos humanos o el control de inventarios. Todas estas aplicaciones procesan datos y generan información para el usuario.

El software de hoja de cálculo, de diseño asistido por computadora (CAD), de procesamiento de texto, de manejo de Bases de Datos, pertenecen a esta categoría. La mayoría de software para uso general se vende como paquete; es decir, software con documentación para el usuario (manuales de referencia, entre otros).

Algunos ejemplos de software de aplicación que se pueden mencionar son los siguientes:

a) Procesadores de Texto

Son utilizados para escribir cartas, memorándums y otros documentos. El usuario teclea una serie de letras o párrafos que son mostradas en la pantalla. Esto facilita las tareas de adherir, borrar y cambiar el texto hasta que el documento quede exactamente como se desea.

Algunas características avanzadas que se pueden encontrar en la actualidad en los procesadores de texto son: corrector de ortografía, diccionario de sinónimos, presentación preliminar del texto antes de imprimir, entre otras.

> Ejemplos de procesadores de texto: Word, AmiPro, Wordperfect, StarOffice Writer.

b) Hojas de Cálculo

Una Hoja de Cálculo es una herramienta para calcular y evaluar números. También ofrece capacidad para crear informes y presentaciones para comunicar lo que revela el análisis; el usuario teclea los datos y las fórmulas que serán usadas para obtener los resultados; después el programa aplica las fórmulas a los datos y así obtiene los resultados finales.

La mayoría de las Hojas de Cálculo cuentan también con la posibilidad de graficar estos resultados en diferentes estilos de gráficas (Barras, Líneas, Pastel, etc).

Ejemplos de Hojas de Cálculo: Excel, Lotus 123, Quatro, StarOffice Calc.

c) Administradores o manejadores de Bases de Datos:

Los DBMS (Data Base Management System) son la herramienta que las computadoras utilizan para realizar el procesamiento y almacenamiento ordenado de los datos. Una base de datos es un recipiente para colecciones relacionadas de datos. Por ejemplo, una agenda puede ser una base de datos donde se almacenan los nombres, direcciones y números telefónicos de amigos y contactos de negocios. La Base de Datos de una Compañía puede contener información acerca de los consumidores, vendedores, empleados, ventas en inventario.

Ejemplos de DBMS: PostgreSQL, Oracle, Access.

d) Paquetes de Presentación.

Son un tipo de software que permite al usuario diseñar presentaciones para desplegarlas a través de la misma computadora o imprimir diapositivas y acetatos. Contienen opciones avanzadas para integrar efectos en cada cambio de diapositiva.

Ejemplos: Presentation, Power Point.

Lenguajes de programación

Los lenguajes de programación cierran el abismo entre las computadoras, que sólo trabajan con números binarios, y los humanos, que prefieren utilizar palabras y otros sistemas de numeración.

Mediante los programas se indica a la computadora que tarea debe realizar y cómo efectuarla, pero para ello es preciso introducir estas órdenes en un lenguaje que el sistema pueda entender. En principio, la computadora sólo entiende las instrucciones en código máquina, es decir, el específico de la computadora. Sin embargo, a partir de éstos se elaboran los llamados lenguajes de alto y bajo nivel.

a) Lenguajes de bajo nivel

Utilizan códigos muy cercanos a los de la máquina, lo que hace posible la elaboración de programas muy potentes y rápidos, pero son de difícil aprendizaje.

b) Lenguajes de alto nivel

Por el contrario de los lenguajes de bajo nivel, son de uso mucho más fácil, ya que en ellos un solo comando o instrucción puede equivaler a millares en código máquina. El programador escribe su programa en alguno de estos lenguajes mediante secuencias de instrucciones. Antes de ejecutar el programa la computadora lo traduce a código máquina de una sola vez (lenguajes compiladores) o interpretándolo instrucción por instrucción (lenguajes intérpretes).

Dado que es muy difícil que los programadores creen programas directamente en código máquina, utilizan lenguajes que consideran más fáciles de leer, escribir y entender.

Ejemplos de lenguajes de alto nivel son: Pascal, Cobol, Basic, Fortran, C++.

Tipos de software

Una vez que se han citado algunas definiciones de software y se ha visto de que manera se puede clasificar, es conveniente establecer otra forma en que se puede separar el software.

Lo anterior significa que además se pueden considerar dos tipos de software, que en los últimos años ha tomado mucha importancia está distinción que se hace entre ellos, principalmente por la diferencia en costo (aunque no es la única), que lleva a concebirlos como software libre y software propietario.

Software Propietario

El Software Propietario es el software que se conoce habitualmente y hasta el día de hoy es el más utilizado.

Se distribuye en "formato binario" y su licencia restringe su copia, redistribución, modificación y uso a lo especificado en la licencia (realmente sólo permite el uso en ciertas condiciones, el resto está prohibido).

a) Ventajas del software propietario

- Las compañías productoras de software propietario por lo general tienen departamentos de control de calidad que llevan a cabo muchas pruebas sobre el software que producen.
- Se destina una parte importante de los recursos a la investigación sobre la utilidad del producto.
- Se tienen contratados algunos programadores muy capaces y con mucha experiencia.
- El software propietario de marca conocida ha sido usado por muchas personas y es relativamente fácil encontrar a alguien que lo sepa usar.

- Existe software propietario diseñado para aplicaciones muy específicas que no existe en ningún otro lado mas que con la compañía que lo produce
- Los planes de estudios de la mayoría de las universidades del país tienen tradicionalmente un marcado enfoque al uso de herramientas propietarias y las compañías fabricantes ofrecen a las universidades planes educativos de descuento muy atractivos. De ahí que los recién egresados pueden comenzar su vida productiva utilizando estos productos de inmediato. No obstante, en los centros de estudio más prestigiados se observa un cambio en esta tendencia.
- Existe gran cantidad de publicaciones, ampliamente difundidas, que documentan y facilitan el uso de las tecnologías proveídas por compañías de software propietario.

b) Desventajas del software propietario

- Es difícil aprender a utilizar completamente el software propietario sin haber asistido a costosos cursos de capacitación.
- El funcionamiento del software propietario es un secreto que guarda celosamente la compañía que lo produce. En muchos casos resulta riesgosa la utilización de un componente que es como una caja negra, cuyo funcionamiento se desconoce y cuyos resultados son impredecibles. En otros casos es imposible encontrar la causa de un resultado erróneo, producido por un componente cuyo funcionamiento se desconoce.
- En la mayoría de los casos el soporte técnico es insuficiente o tarda demasiado tiempo en ofrecer una respuesta satisfactoria.
- Es ilegal extender una pieza de software propietario para adaptarla a las necesidades particulares de un problema específico. En caso de que sea vitalmente necesaria tal modificación, es necesario pagar una elevada suma de dinero a la compañía fabricante, para que sea ésta quien lleve a cabo la modificación a su propio ritmo de trabajo y sujeto a su calendario de proyectos.
- La innovación es derecho exclusivo de la compañía fabricante. Si alguien tiene una idea innovadora con respecto a una aplicación propietaria, tiene que elegir entre venderle la idea a la compañía dueña de la aplicación o escribir desde cero su propia versión de una aplicación equivalente, para una vez logrado esto poder aplicar su idea innovadora.
- Es ilegal hacer copias del software propietario sin antes haber contratado las licencias necesarias.
- Si una dependencia de gobierno tiene funcionando exitosamente un sistema dependiente de tecnología propietaria no lo puede compartir con otras dependencias a menos que cada una de éstas contrate todas las licencias necesarias.
- Si la compañía fabricante del software propietario se va a la banca rota el soporte técnico desaparece y las posibilidades de en un futuro tener versiones mejoradas o de corregir las erratas de dicho software también. Los clientes que contrataron licencias para el uso de ese software quedan completamente abandonados a su propia suerte.

- Si una compañía fabricante de software es comprada por otra más poderosa, es probable que esa línea de software quede descontinuada y nunca más en la vida vuelva a tener una modificación.
- En la mayoría de los casos el usuario se hace dependiente de un solo proveedor.

Software Libre

El Software Libre se refiere a un modelo de desarrollo y de distribución del software desarrollado cooperativamente. En los últimos años se han escuchado cada vez más los términos software libre y más recientemente software de fuentes abiertas. Aunque esta concepción es bastante antigua (hablando de tiempo informático, 1984) ha tomado un auge espectacular gracias a la interconexión de proyectos por Internet.

Se distribuye en "formato de código fuente" o "formato binario" siempre que se indique donde encontrar el código fuente y su licencia permite su copia, redistribución, modificación y uso sin las restricciones citadas.

Un "formato binario" es una secuencia de "unos" y "ceros" ilegible para el hombre, sólo lo entienden las computadoras, y es el que se ejecuta en el procesador.

Un "formato de código fuente" es un archivo o archivos de texto donde se especifican las acciones a realizar por el procesador, es legible y entendible por el hombre y a través de un programa llamado "compilador" se genera el (traduce a) "formato binario" que se ejecutará en el procesador.

"Software Libre" se refiere a la libertad de los usuarios de correr, copiar, distribuir, estudiar, cambiar y mejorar el software, es decir, se refiere a las cuatro libertades de los usuarios de software:

1. La libertad de correr el programa, con cualquier propósito

2. El ser libre de distribuir copias sin cobrar o cobrando sólo una pequeña cantidad, es decir, no se tienen que pedir ni pagar licencias.

3. También se debe tener la libertad de hacer modificaciones y utilizarlas de manera privada para un trabajo o juego, sin ni siquiera mencionar que dichas modificaciones existen. Si se publican cambios, no se necesita avisar a nadie en particular, o de una manera específica.

4. La libertad de estudiar como funciona el programa, y adaptarlo a sus necesidades. El acceso al código fuente es una precondición para esto. En particular, esto significa que el código fuente debe estar disponible.

Sin embargo, algunos tipos de reglas acerca de la manera de distribuir software libre son aceptables, cuando no entran en conflicto con las libertades centrales.

Por ejemplo, "*copyleft*" es la regla que implica que cuando se redistribuya el programa, no se pueden agregar restricciones para denegar a otras personas las libertades centrales. Esta regla no entra en conflicto con las libertades centrales; sino que las protege.

El software libre es confiable porque los desarrolladores se preocupan realmente de la fiabilidad. Los paquetes de software libre no siempre compiten comercialmente, pero sí compiten por una buena reputación y un programa que sea insatisfactorio no alcanzará la popularidad que los desarrolladores esperan. Lo que es más, un autor que pone el código fuente al alcance de la vista de todos arriesga su reputación, y le conviene hacer el software limpio y claro, bajo pena de la desaprobación de la comunidad.

a) Ventajas del software libre

- Cualquier persona tiene derecho de usarlo sin costo alguno.
- Todo el que quiera tiene derecho a acceder a su diseño y aprender de él.
- Todos los usuarios tiene derecho de modificarlo: si el software tiene limitaciones o no es adecuado para una tarea, es posible adaptarlo a necesidades específicas y redistribuirlo libremente.
- No tiene un costo asociado (gratuito).
- Es de libre distribución (cualquier persona puede regalarlo, venderlo o prestarlo).
- Ahorros multimillonarios en la adquisición de licencias
- Combate efectivo a la copia ilícita de software
- Eliminación de barreras referentes a presupuesto
- Beneficio social para el país
- Beneficio tecnológico para el país
- Existen muchos colaboradores dispuestos a ayudar
- Tiempos de desarrollo sobre algo que no exista son menores por la amplia disponibilidad de herramientas y librerías
- Las aplicaciones son fácilmente auditadas antes de ser usadas en procesos de misión crítica, además del hecho de que las más populares se encuentran muy depuradas.

b) Desventajas del software libre

- El tiempo de aprendizaje es mayor
- El software libre no tiene garantía proveniente del autor
- Se necesita dedicar recursos a la reparación de erratas
- No existe una compañía única que respalde toda la tecnología
- Las interfaces amigables con el usuario (GUI) y la multimedia apenas se están estabilizando.
- La mayoría de la configuración de hardware no es intuitiva, se requieren conocimientos previos acerca del funcionamiento del sistema operativo y fundamentos del equipo a conectar para lograr un funcionamiento adecuado.

- El usuario debe tener nociones de programación, ya que la administración del sistema recae mucho en la automatización de tareas y esto se logra utilizando, en muchas ocasiones, lenguajes como perl, python, shell, etc.
- La diversidad de distribuciones, métodos de empaquetamiento, licencias de uso, herramientas con un mismo fin, etc., pueden crear confusión en cierto número de personas.

Con base en un análisis comparativo de ventajas y desventajas para los tipos de software mencionados, se decidió optar por la utilización de software libre para el diseño, desarrollo e implantación del Sistema de Acopio de Información en Línea de las Actividades Académicas de la Facultad de Ingeniería, aunado a que este tipo de software ha tenido una difusión considerable en los últimos años y es utilizado ya en muchos sectores, tanto públicos como privados, por ejemplo en sistemas empleados en el gobierno, demostrándose con ello la creciente difusión del mismo.

Además de lo anterior es importante mencionar que también se optó por el uso de software libre debido a que los servidores que se utilizaron para desarrollo y pruebas del SAILFI (servidores de la propia Facultad de Ingeniería, a cargo de su Unidad de Cómputo Académico) contaban ya con este tipo de software, entre ellos: sistema operativo, lenguaje de programación, manejador de bases de datos y servidor web.

De tal forma, las herramientas utilizadas en este proyecto son el sistema operativo Linux, el manejador de bases de datos PostgreSQL, el lenguaje de programación PHP y el servidor de HTTP Apache. Es conveniente citar además que como diseñadores y desarrolladores del sistema, no se fungió como (ni se tenían los privilegios de) administradores de los servidores que proporcionan estos servicios.

2.2.1 Sistema Operativo Linux

Linux es una versión de UNIX libremente distribuible e independiente, para máquinas con procesadores x86, Motorola 68k, Digital Alpha, Sparc, Mips y Motorola Power PC.

A menudo se dice que Linux es un clon de UNIX, pero más allá de ello, es un sistema operativo multiusuario y multitareas que cumple con *POSIX*, para los procesadores 386 de Intel y posteriores. POSIX es un estándar internacional para sistemas operativos y software, en el que se detallan estándares de interoperabilidad. Así, Linux no requiere de MS-DOS o de Windows para funcionar, de hecho, puede sustituir a estos programas en la computadora.

Linux ha sido un acontecimiento único en la evolución de la computación. Aunque no es un producto comercial respaldado por una gran corporación, es un sistema operativo fruto de la colaboración de un equipo heterogéneo de entusiastas de la

computación de todo el mundo. Este equipo se valió de los recursos de Internet para comunicarse y crear este sistema operativo.

En la actualidad, Linux es utilizado por miles de usuarios para desarrollo de software, redes y para plataformas de usuarios finales, además de que, entre la gran variedad de sistemas operativos alternos que existen, se ha convertido en una opción interesante para sustituir incluso a los más populares, entre los que se encuentran Windows en sus versiones 98, 2000, XP o NT.

Linux está disponible en Internet en cientos de servidores ftp y con algunos distribuidores en discos CD-ROM que lo ofrecen empacado con manuales o información, que es realmente la que cuesta, pues el programa es gratuito. Algunos versiones conocidas de Linux son: Caldera, Debian, Slackware, Red Hat, entre otros.

2.2.1.1 Breve historia de Linux

Linux es el invento de un estudiante de ciencias de la computación de la Universidad de Helsinki, en Finlandia, en aquel entonces de 23 años, cuyo nombre es Linus Torvalds.

Linux cobró vida en 1991 como un proyecto de entretenimiento de Linus, quién originalmente inició el hacking del núcleo como su proyecto favorito y que inspirado por su interés esperaba crear una versión más completa de UNIX para los usuarios de Minix.

Minix era un sistema operativo tutorial tipo UNIX disponible para las PC’s basadas en Intel, escrito por el reconocido y respetado científico de la computación Andrew Tannebaum, y que adquirió popularidad en varias plataformas de PC’s, incluidas las que estaban basadas en MS-DOS.

El 5 de octubre de 1991, Linus anunció su primera versión “oficial” de Linux, la 0.02. Desde entonces, muchos programadores han respondido a su llamado, y han ayudado a construir Linux como un sistema operativo completamente funcional.

De esta manera, versiones más recientes del sistema operativo soportan muchos más periféricos, desde procesadores hasta joysticks, sintonizadores de televisión, CD-ROMs no ATAPI y reconocen gran cantidad de tarjetas de sonido, incluyendo también soporte para tipos de archivos para Macintosh HFS, UNIX UFS y en modo de lectura, HPFS de OS/2 y NTFS, de NT.

Sin embargo, el proyecto aún no se completa, ya que prácticamente cientos o quizá miles de personas diseminadas en todo el mundo actualizan de manera constante a Linux, contribuyendo así a su crecimiento

2.2.1.2 Características del sistema operativo Linux

No es posible hablar de Linux dejando a un lado al sistema operativo UNIX, debido a que, como se mencionó anteriormente, Linux es un proyecto iniciado para crear una versión práctica de UNIX en máquinas basadas en Intel, mejor conocidas como computadoras compatibles con las PC's de IBM, tan familiares a casi todo el mundo.

Debido a que Linux es muy semejante a UNIX muchas de las operaciones y procedimientos necesarios para usar Linux también pueden aplicarse a numerosos sistemas UNIX. Por ello, al aprender a manejar Linux, también se hace con la mayoría de los sistemas UNIX.

UNIX, distribuido por diversos proveedores, viene en distintas presentaciones, incluidas las versiones para las plataformas Intel para PC; sin embargo, casi todas son muy costosas. Por otra parte, Linux ofrece una solución relativamente económica (gratuita si se tiene acceso a Internet) para aprender lo referente a procedimientos y comandos tipo UNIX, la interfaz gráfica para usuarios de X Windows y el modo de tener acceso a Internet mediante Linux.

Los beneficios derivados del uso del sistema operativo UNIX, y por tanto de Linux, provienen de su potencia y flexibilidad. Éstas son resultado de numerosas características integradas al sistema, las que están disponibles tan pronto como éste se inicia.

Multitareas

La palabra multitarea describe la habilidad de ejecutar, aparentemente al mismo tiempo, numerosos programas sin obstaculizar la ejecución de cada aplicación. Esto se conoce como multitareas preferente, porque cada programa tiene garantizada la posibilidad de correr, esto es, cada programa no se ejecuta sino hasta que el sistema operativo lo aparta para permitir que otros programas corran.

En Linux mientras se da prioridad a la tarea en la que se trabaja cada segundo, las que se ejecutan en segundo plano continúan su función hasta finalizar o hasta que necesiten datos de entrada.

Multiusuario

El concepto de que numerosos usuarios pudieran acceder aplicaciones o el potencial de procesamiento de una sola PC era sólo un sueño hasta hace algunos años. UNIX y Windows NT ayudaron a que este sueño se convirtiera en realidad. La capacidad de Linux para asignar tiempo del microprocesador a numerosas aplicaciones simultáneas se prestó como consecuencia a servir a numerosas personas al mismo tiempo, cada una ejecutando una o más aplicaciones.

Shells programables

El shell programable es otra característica que hace que Linux sea el sistema operativo más flexible. Dentro de la estructura del shell está al alcance un mundo totalmente nuevo para aquellos con la audacia suficiente para dominar los matices de la sintaxis de comandos de Linux.

Aunque muchas versiones de Linux incluyen más de un tipo de *shell*, en general, todos funcionan igual. La principal diferencia entre los tres shells disponibles radica en la sintaxis de la línea de comandos. A pesar de que no es una limitación grave, pueden surgir dificultades si se trata de usar comandos o sintaxis de shell C en el shell BOURNE o BASH.

La programación de shells de Linux sirve a tantas funciones diferentes como personas hay con el deseo de probarla. Muchos emplean esta característica con el objeto de personalizar su sistema y hacerlo más amigable para el usuario. Otros la encuentran útil para poner en línea muchas de las aplicaciones que ejecutan, permitiendo que las aplicaciones realicen algunos procesos en segundo plano a fin de tener libertad para trabajar en otras.

Comunicaciones y capacidades de red

La superioridad de Linux sobre otros sistemas operativos es igual de evidente en sus utilerías para comunicaciones y red. Ningún otro sistema operativo incluye capacidades para red estrechamente acopladas, y tampoco tiene la flexibilidad integral de estas mismas características. Si se necesita hablar con otro usuario mediante la utilería de correo electrónico o bajar (transferir) archivos grandes de un sistema que está al otro lado del país, Linux ofrece los medios para hacerlo.

Las capacidades de comunicación inherentes al sistema operativo fueron diseñadas para soportar numerosas tareas y numerosos usuarios en una red bastante extensa.

Portabilidad de sistemas abiertos

La portabilidad es tan solo la capacidad de transportar un sistema operativo de una plataforma a otra para que siga funcionando del mismo modo que lo hacía. Desde luego, Linux es un sistema operativo portátil. Hoy día Linux puede operar en cualquier ambiente y plataforma, desde laptos hasta mainframes.

La portabilidad proporciona los medios para que diferentes plataformas de cómputo que corren Linux se comuniquen adecuada y efectivamente con cualquiera de las otras sin necesidad de agregar interfaces de comunicación especiales, costosas y de última hora.

2.2.1.3 Ventajas de usar Linux

De los numerosos sistemas operativos disponibles en la actualidad, Linux es el sistema gratuito más popular y de amplio acceso. Linux proporciona un sistema completo con capacidades integradas para multiusuario y multitareas que aprovechan toda la potencia de procesamiento de los sistemas de cómputo. En pocas palabras las ventajas que proporciona el uso de Linux son:

1. Estabilidad, no se traba a cada rato.
2. Seguridad, es mucho más seguro que otros servidores.
3. Compatibilidad, reconoce la mayoría de los otros sistemas operativos en una red.
4. Velocidad, es mucho más veloz para realizar las tareas.
5. Posee el apoyo de miles de programadores a nivel mundial.
6. El paquete incluye el código fuente, lo que permite modificarlo de acuerdo a las necesidades del usuario.
7. Ideal para la programación, ya que se puede programar en Linux para distintas plataformas.
8. Un sistema de crecimiento rápido.
9. Se puede usar en casi cualquier computadora, desde una 386.
10. Multitareas real.
11. Puede manejar múltiples procesadores. Incluso hasta 16 procesadores
12. Libre de virus, aún no se conoce ningún virus para Linux.
13. Maneja discos duros de hasta 16 Terabytes.
14. Se consiguen parches con facilidad, además de ser gratuitos.
15. Se posee el apoyo de millones de usuarios a nivel mundial.
16. Los fabricantes de hardware le están dando su apoyo, como IBM y Compaq.
17. Vendedores y desarrolladores implementan un sistema de certificación para Linux
18. La corporación DATA internacional predice que el crecimiento de este programa será del orden de un 25% anual.
19. Utiliza TCP/IP para su propia conectividad en red.

2.2.1.4 Desventajas de usar Linux

Entre las desventajas que pueden citarse en el uso del sistema operativo Linux se encuentran:

1. Linux no cuenta con una empresa que los respalde, por lo que existe carencia de soporte técnico.
2. Incapacidad para usar software común. Es poco probable que las aplicaciones generales para sistemas operativos como el DOS y el OS/2 funcionen bajo Linux.
3. Posible falta de experiencia, ya que se debe aprender a administrar un sistema operativo Linux.

4. Linux corre el riesgo de llegar a fragmentarse como fue el caso de UNIX
5. Algunas empresas pueden llegar a ayudar a Linux con la intención de mejorar sus relaciones públicas, aunque en el fondo no tengan ninguna intención de utilizarlo fielmente.

2.2.2 Sistema Manejador de Base de Datos: PostgreSQL

PostgreSQL, desarrollada originalmente en el Departamento de Ciencias de la Computación de la Universidad de California en Berkeley, fue pionera en muchos de los conceptos de bases de datos relacionales orientadas a objetos que ahora empiezan a estar disponibles en algunas bases de datos comerciales. Ofrece suporte al lenguaje SQL92/SQL3, integridad de transacciones, y extensibilidad de tipos de datos. PostgreSQL es un descendiente de dominio público y código abierto del código original de Berkeley.

2.2.2.1 Breve historia de PostgreSQL

El Sistema Manejador de Bases de Datos (DBMS) Relacionales Orientadas a Objetos conocido como PostgreSQL (y brevemente llamado Postgres95) está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el manejador de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, PHP, Java, Perl, tcl y python).

La implementación del *DBMS* Postgres comenzó en 1986. Los conceptos iniciales para el sistema fueron presentados en The Design of Postgres y la definición del modelo de datos inicial apareció en The Postgres Data Model. El diseño del sistema de reglas fue descrito en ese momento en The Design of the Postgres Rules System. La lógica y arquitectura del gestor de almacenamiento fueron detalladas en The Postgres Storage System. El proyecto terminó oficialmente con la Versión 4.2.

En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de lenguage SQL a Postgres. Postgres95 fue publicado a continuación en la Web para que encontrara su propio hueco en el mundo como un descendiente de dominio público y código abierto del código original Postgres de Berkeley.

Muchos cambios internos mejoraron el rendimiento y la facilidad de mantenimiento.

Éstas fueron las principales mejoras:

- El lenguaje de consultas Postgres fue reemplazado con SQL (implementado en el servidor). Las subconsultas no fueron soportadas hasta PostgreSQL, pero podían ser emuladas en Postgres95 con funciones SQL definidas por el usuario. Las funciones agregadas fueron reimplementadas. También se añadió una implementación de la cláusula GROUP BY. La interfaz libpq permaneció disponible para programas escritos en C.

- Además del programa de monitorización, se incluyó un nuevo programa (psql) para realizar consultas SQL interactivas usando la librería GNU readline.

- Se revisó la interfaz con objetos grandes. Los objetos grandes de inversión fueron el único mecanismo para almacenar objetos grandes (el sistema de archivos de inversión fue eliminado).

- Se eliminó también el sistema de reglas al nivel de instancia, si bien las reglas siguieron disponibles como reglas de reescritura.

- Se distribuyó con el código fuente un breve tutorial introduciendo las características comunes de SQL y de Postgres95.

- Se utilizó GNU make (en vez de BSD make) para la compilación. Postgres95 también podía ser compilado con un gcc sin parches (al haberse corregido el problema de alineación de variables de longitud doble).

PostgreSQL

En 1996, se hizo evidente que el nombre “Postgres95” no resistiría el paso del tiempo, y por ello se eligió un nuevo nombre, PostgreSQL, para reflejar la relación entre el Postgres original y las versiones más recientes con capacidades SQL. Al mismo tiempo, se hizo que los números de versión partieran de la 6.0, volviendo a la secuencia seguida originalmente por el proyecto Postgres.

Con PostgreSQL, el énfasis en el código del motor de datos aumentó características y capacidades, aunque el trabajo continúa en todas las áreas. Las principales mejoras en PostgreSQL incluyen:

- Los bloqueos de tabla fueron sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad al momento desde pg_dump, mientras la base de datos permanece disponible para consultas.

- Se implementaron importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).

- Se añadieron funcionalidades en línea con el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadena literales, conversión de tipos y entrada de enteros binarios y hexadecimales.

- Los tipos internos fueron mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.

2.2.2.2 Características de PostgreSQL

Los sistemas de mantenimiento de bases de datos relacionales tradicionales soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos. El modelo relacional sustituyó modelos previos en parte por su "simplicidad". Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. PostgreSQL ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema

clases
herencia
tipos
funciones

Otras características aportan potencia y flexibilidad adicional:

Restricciones (Constraints)
Disparadores (triggers)
Reglas (rules)
Integridad transaccional

Estas características colocan a PostgreSQL en la categoría de las Bases de Datos identificadas como *objeto-relacionales*. Nótese que éstas son diferentes de las referidas como *orientadas a objetos*, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales. PostgreSQL tiene algunas características que son propias del mundo de las bases de datos

orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera.

2.2.3 Lenguaje PHP

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

No es lo mismo que un script CGI escrito en otro lenguaje de programación como Perl o C. En vez de escribir un programa con muchos comandos para crear una salida en HTML, se escribe el código HTML con cierto código PHP embebido (introducido) en el mismo, que produce cierta salida. El código PHP se incluye entre etiquetas especiales de comienzo y final que permiten entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

2.2.3.1 Breve historia de PHP

PHP fue concebido en otoño de 1994 por Rasmus Lerdorf. Las primeras versiones no distribuidas al público fueron usadas en sus páginas web para mantener un control sobre quien consultaba su currículum. La primera versión disponible para el público a principios de 1995 fue conocida como "Herramientas para páginas web personales" (Personal Home Page Tools). Consistían en un analizador sintáctico muy simple que sólo entendía unas cuantas macros y una serie de utilidades comunes en las páginas web de entonces, un libro de visitas, un contador y otras pequeñas cosas. El analizador sintáctico fue reescrito a mediados de 1995 y fue nombrado PHP/FI version 2. FI viene de otro programa que Rasmus había escrito y que procesaba los datos de formularios. Así que combinó las "Herramientas para páginas web personales", el "intérprete de formularios", añadió soporte para mSQL y PHP/FI vio la luz. PHP/FI creció a gran velocidad y la gente empezó a contribuir en el código.

Es difícil dar estadísticas exactas, pero se estima que a finales de 1996 PHP/FI se estaba usando al menos en 15.000 páginas web alrededor del mundo. A mediados de 1997 este número había crecido a mas de 50.000. En ese tiempo el desarrollo del proyecto sufrió un profundo cambio, dejó de ser un proyecto personal de Rasmus, al cual habían ayudado un grupo de usuarios y se convirtió en un proyecto de grupo mucho más organizado. El analizador sintáctico se rescribió desde el principio por Zeev Suraski y Andi Gutmans y este nuevo analizador

estableció las bases para PHP versión 3. Gran cantidad de código de PHP/FI fue portado a PHP3 y otra gran cantidad fue escrito completamente de nuevo.

2.2.3.2 Características de PHP

PHP tiene soporte para la programación orientada a objetos, es decir, se pueden crear clases para la construcción de objetos, con sus constructores etc. El soporte de objetos que se tiene en PHP no es tan potente como el de C++, pero cada vez es más completo y en versiones recientes de PHP4 este soporte ha sido mejorado.

La sintaxis de PHP es muy similar a la de C o C++. Quizá lo más destacado ocurre a nivel semántico: el tipado es muy poco estricto, algo típico en los lenguajes "script", y cuando se crea una variable en ella se puede guardar el tipo de datos que se desee. Esto es muy flexible y cómodo para el desarrollador, aunque los errores que se cometen pueden ser mucho más graves y difíciles de corregir. Las posibilidades del intérprete para detectar incompatibilidades entre variables se reducen mucho y pueden ocurrir cosas extrañas en determinadas circunstancias si no se tiene cuidado.

Existe un gran número de manejadores de bases de datos a los que se puede acceder utilizando PHP, entre los que están:

- Adabas D
- dbm
- dBase
- filePro
- Hyperwave
- Informix
- InterBase
- LDAP
- Microsoft SQL server
- mSQL
- MySQL
- ODBC
- Oracle
- PostgreSQL
- Solid
- Sybase

Entre otras características de PHP están el soporte para el(la):

- Acceso a servidores IMAP
- Envío de correo con SMTP
- Acceso a servidores de FTP
- Generación dinámica de gráficos y documentos PDF

- Análisis de documentos XML
- Corrector de ortografía
- Generación de datos en WDDX (Intercambio Web de Datos Distribuidos)

PHP se ha diseñado de forma muy modular y ha sido sencillo, conforme han surgido, utilizar las librerías. Toda esta funcionalidad está basada en librerías que en su mayor parte no han sido desarrolladas por el equipo de PHP. La gente de PHP lo que ha implementado han sido los cabos necesarios para poder acceder a las librerías.

2.2.3.3 PHP4

La principal novedad de PHP4 fue el interprete del lenguaje PHP. Gracias a Zend, desarrollado por Andi Gutmans y Zeev Suraski, se logró aumentos de entre 5 y 10 veces en la velocidad de ejecución de páginas PHP. Este nuevo rendimiento lo puso por delante de ASP, la tecnología de Microsoft, por lo que le permitió competir directamente con ASP en la plataforma de esta compañía. La plataforma de desarrollo Apache+PHP en entornos Microsoft está logrando ya rendimientos superiores a IIS+ASP, por lo que aumentó su uso en éstas plataformas. Esto significará que en el futuro el desarrollador en plataformas NT, por ejemplo, no se verá obligado al uso de IIS+ASP para el desarrollo de sistemas web, tal y como había sucedido hasta ahora. Por otro lado, PHP estaba hasta el momento pensado básicamente para Apache. Era el único que tenía soporte para tener a PHP como un módulo del servidor. Todos los demás servidores de web sólo podían utilizar a PHP como cgi. En PHP4 se añadió soporte de la API de Apache, el de Netscape (NSAPI), el soporte para ISAPI (API de Internet Explorer) y para la API del servidor de AOL. De esta forma todos estos servidores pueden lograr rendimientos similares a Apache en su ejecución de PHP.

Además del aumento en rendimiento en lo referente al soporte de objetos en PHP, la sobrecarga de estos se mejoró. De esta forma se añadió la posibilidad de métodos polimórficos, métodos que según desde qué referencia a un objeto se llamen, se comportan de una forma o de otra.

Aunado a lo citado anteriormente, el nuevo intérprete de PHP es capaz de liberar los recursos que se reserven de forma automática con lo que el programador se puede olvidar de liberarlos.

2.2.4 INTERNET

Cuando se habla de herramientas de software a utilizar en el desarrollo de un sistema de información, y si se pretende que éste funcione en línea, una de las que resulta imprescindible es precisamente una de las herramientas de mayor difusión y crecimiento en el mundo entero, Internet.

Internet es una red de redes, es decir, muchas redes de área local de empresas, universidades, centros de investigación, organismos públicos, etc., están interconectados entre sí por medio de protocolos comunes (TCP/IP).

La información contenida en cada una de las computadoras de esta red es accesible desde cualquier otra conectada a ésta; es decir, aquellas organizaciones que quieren conectar sus computadoras a Internet, no tienen más que enlazarse a otra computadora que, a su vez, esté en Internet. Sin embargo cada organización es responsable de sus propias computadoras y de sus conexiones. No existe ninguna organización que imponga reglas, aunque existen grupos y organizaciones que se dedican a controlar de alguna forma el tráfico en ella.

2.2.4.1 Breve historia de internet

Internet fue creada a partir de un proyecto del departamento de defensa de los Estados Unidos llamado ARPANET (Advanced Research Project Network). Iniciado en 1969, el propósito principal era la investigación y desarrollo de protocolos de comunicación para redes de área amplia, a fin de enlazar redes de transmisión en paquetes informáticos de diferentes tipos, capaces de resistir las condiciones de operación más difíciles y continuar funcionando aún con la pérdida de una parte de la red (por ejemplo en caso de guerra).

Estas investigaciones dieron como resultado el protocolo *TCP/IP*. Durante el desarrollo de este protocolo se incrementó notablemente el número de redes locales de agencias gubernamentales y de universidades que participaban en el proyecto, las funciones militares se separaron y se permitió el acceso a la red a todo aquel que lo requiriese sin importar de que país provenía la solicitud, siempre y cuando fuera para fines académicos o de investigación (y por su puesto que pagara sus propios gastos de conexión). Los usuarios pronto encontraron que la información que había en la red era por demás útil y si cada quién aportaba algo se enriquecería aún más con el acervo de información existente.

2.2.4.2 Servicios en internet

Internet puede ser adquirido si se tiene acceso a las organizaciones antes mencionadas o bien contratando el servicio con proveedores de Internet. Estos negocios establecen una conexión con Internet y posteriormente venden el

servicio a personas que lo soliciten, pagando una cuota ya sea por hora, mes o año.

Internet tiene los siguientes servicios disponibles:

- Correo electrónico
- FTP
- Usenet
- Gopher
- Listas de correo
- Telnet
- World Wide Web

Para el sistema en cuestión (SAILFI) el servicio de Internet que se utiliza es el servicio de World Wide Web (WWW), ya que éste trabaja bajo dicho esquema de funcionamiento.

World Wide Web

La World Wide Web (WWW), también llamada Web, es el servicio más nuevo de Internet y que permite compartir información a todo el público en general que tenga acceso a Internet. Es la parte más importante de Internet, por que es aquí donde se conjuntan todas las páginas.

La Web o telaraña mundial utiliza una nueva tecnología que permite acceder a los recursos de Internet de una manera sencilla en un ambiente gráfico interactivo, donde la información es fácil de encontrar y de acceder.

Algo que ha sido revolucionario en la Web es el hipertexto. Los hipertextos relacionan información (enlaces) en un documento, con información referente al mismo tema, por medio de códigos de dirección. Un usuario sencillamente hace clic en un texto resaltado para alcanzar mayor detalle sobre un tema o saltar a otro relacionado. Se puede viajar por la red de un documento a otro, dispersos por todo el mundo, mediante referencias cruzadas, lo que significa que la selección de un término en una página, puede conducir hacia otro documento situado en otra máquina a miles de kilómetros.

Las claves de cómo funciona WWW son el protocolo *HTTP* y el *HTML*. De una manera más sencilla, http es el protocolo que gobierna los hipertextos a lo largo de la red y el código HTML crea y da formato a las páginas de un documento Web.

2.3 Análisis de requerimientos de Hardware

Cuando se trata de elegir el hardware es cuestión de mezclar conocimientos técnicos con el sentido común. El enfoque técnico se refiere a que cada aspecto del hardware de una computadora influye en la velocidad y la eficiencia de sus servicios. Por otro lado, el sentido común indica que se debe conseguir el mejor hardware que se pueda pagar, enfocándose en las piezas que dan mayor valor por el dinero, después de evaluar que servicios se quieren tener disponibles.

Precisar el tipo de hardware necesario está sujeto a muchas variables. Si se determina que lo requerido es más de lo que se puede pagar en ese momento, es necesario decidir por lo más esencial al principio y dejar para después lo menos importante, tomando en consideración que cierto hardware es más fácil de actualizar o de mejorar que otro.

Dependiendo de los servicios a ofrecer, hay tanto máquinas *cliente* como *servidor*. La máquina servidor brinda servicios como FTP, World Wide Web, entre otros. La máquina cliente accede al servidor y utiliza sus servicios. No es obligatorio que el servidor cuente con todas las capacidades necesarias para ejecutar el servicio, a menos que éste sea un cliente que también use el servicio.

La velocidad del CPU afecta de manera directa la rapidez de procesamiento de una computadora, ésto es, se debe pensar en la velocidad del CPU como la velocidad de referencia desde la cual se miden todos los desempeños. Esto significa que para un mayor número de clientes, el procesador deberá tener más velocidad.

Por otra parte, la cantidad de RAM que tenga una computadora es un factor significativo con respecto a cuántos procesos a la vez puede manejar un servidor, y qué tan rápido puede avanzar por ellos. Esto significa que los requerimientos múltiples se procesan más rápido, porque en lugar de ir uno a uno, van de dos, cinco o más a la vez.

El aumento de la capacidad requerida para un número creciente de usuarios simultáneos, depende de lo que estos hagan en el sistema. Así mismo, cuando se incrementa tanto el número de usuarios como la demanda de servicios, el tamaño y tipo del disco duro, se convierten en algo importante.

El tamaño del disco duro se debe escoger dependiendo de cuánto se quiere almacenar en él.

Considerando las herramientas de software elegidas para la implementación del sistema, descritas en el punto anterior de este capítulo, se decidió utilizar como servidor del mismo, uno de los equipos servidores con que cuenta la Unidad de Cómputo Académico (UNICA) de la facultad, aunado a que como se detallará más

adelante, el sistema de acopio de información no tendrá una demanda excesiva de usuarios, y éste soporta perfectamente ese aspecto.

En este servidor se dispone del sistema operativo Linux, el DBMS PostgreSQL y el lenguaje de programación PHP que son los que se utilizaron para desarrollar el SAILFI.

En cuanto a recursos de hardware se trata de un equipo HP Netserver 2000 con procesador Pentium III, memoria RAM de 312 Mb y disco duro de 9 Gb.

Para las máquinas clientes (usuarios del sistema) debido a las ventajas que proporciona PHP sólo se necesitan un navegador o browser y una conexión a internet, recomendándose una memoria RAM de 128 Mb y procesador Pentium II o superior para un óptimo desempeño del sistema.

2.4 Identificación de usuarios del sistema

Como se ha mencionado anteriormente, la idea de contar con un sistema de acopio de información surge del hecho de que en la Facultad de Ingeniería se produce una gran cantidad de información concerniente con las actividades académicas de la institución.

El órgano dentro de la Facultad de Ingeniería encargado de recopilar y manipular esta información es el Departamento de Información y Estadística. Por esta razón una persona perteneciente a este departamento será la encargada de asignar las cuentas para los usuarios del sistema.

Con base en el organigrama de la facultad, figura 2.4.1, se observa que cada división es independiente por lo que también lo son la información y las actividades que éstas realizan. Así cada una de las divisiones será un usuario del sistema, designando a la(s) persona(s) que consideren conveniente, para acceder a él.

De esta manera se tendrán nueve usuarios, los cuales se listan a continuación:

1. División de Ciencias Básicas (DCB)
2. División de Ciencias Sociales y Humanidades (DCSH)
3. División de Ingeniería Civil, Topográfica y Geodésica (DICTyG)
4. División de Ingeniería en Ciencias de la Tierra (DICT)
5. División de Ingeniería Mecánica e Industrial (DIMEI)
6. División de Ingeniería Eléctrica (DIE)
7. División de Estudios de Posgrado (DEP)
8. División de Educación Continua (DEC)
9. Secretaría General (SG)

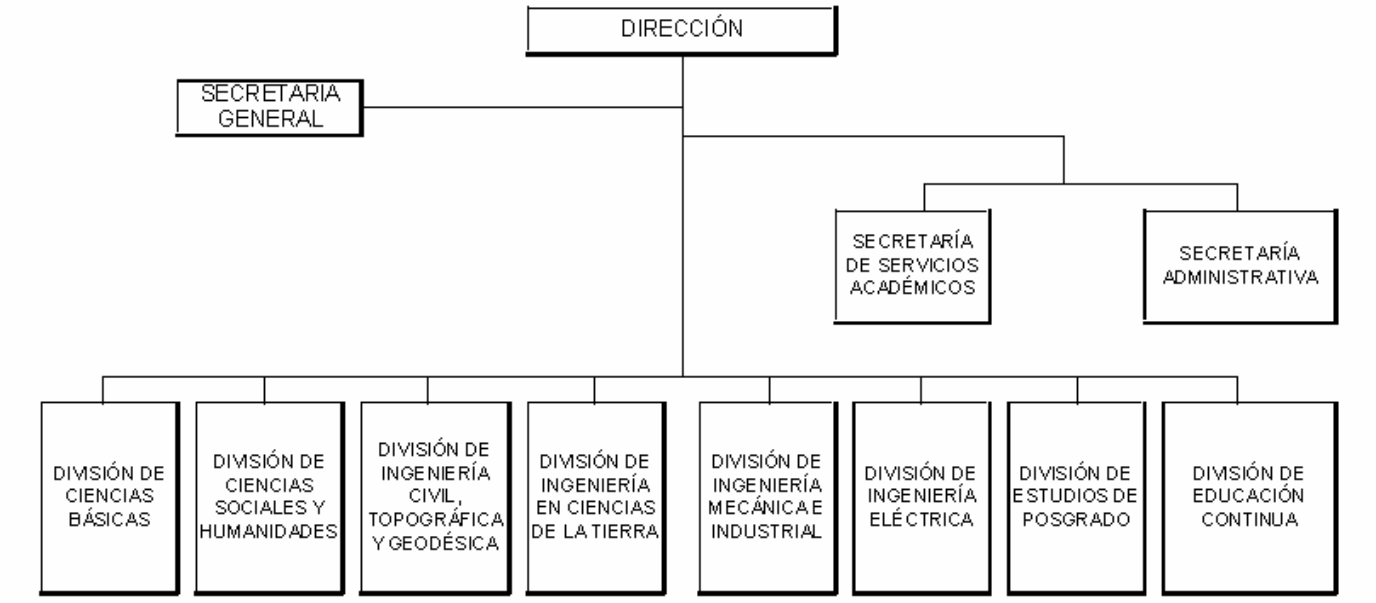


figura 2.4.1 Organigrama de la Facultad de Ingeniería

Es importante mencionar que entre las características del sistema se tiene que al momento de que un usuario es validado, éste sólo puede acceder a la información concerniente con su división, no existiendo posibilidad de que agregue, consulte, elimine y/o modifique la de otras divisiones.

Capítulo 3

Diseño del sistema

Diseño del sistema

El diseño del sistema es un proceso y un modelado a la vez. El proceso de diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del sistema a construir. Algunas de las características principales a considerar son:

El diseño debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acumular todos los requisitos implícitos que desea el cliente.

Debe ser una guía que puedan leer y entender los programadores del código y los que prueban y mantienen el software.

El diseño debe proporcionar una idea completa de lo que es el sistema, enfocando los datos a la parte funcional y al comportamiento de la implementación de éste.

Se debe tener presente que la mayoría de los sistemas actúan de manera recíproca con su medio ambiente recibiendo entradas y produciendo salidas.

3.1 Diagrama de flujo

Un diagrama de flujo representa la esquematización gráfica de un algoritmo, el cual muestra los pasos o procesos a seguir para alcanzar la solución de un problema ya que nos ayuda a determinar cómo funciona realmente el proceso para producir un resultado. Éste puede ser un producto, un servicio, información o una combinación de los tres.

Su correcta construcción es sumamente importante porque a partir del mismo se escribe un programa en algún lenguaje de programación. Si el diagrama de flujo está completo y correcto, el paso del mismo a un lenguaje de programación es relativamente simple y directo.

Es importante resaltar que el diagrama de flujo muestra el sistema como una red de procesos funcionales conectados entre sí por "tuberías" y "depósitos" de datos que permite describir el movimiento de los datos a través del sistema.

Éste describirá:

- Lugares de origen y destino de los datos
- Transformaciones a las que son sometidos los datos
- Lugares en los que se almacenan los datos dentro del sistema
- Los canales por donde circulan los datos.

Al examinar cómo los diferentes pasos en un proceso se relacionan entre sí, se pueden descubrir con frecuencia las fuentes de problemas potenciales. Los diagramas de flujo detallados describen la mayoría de los pasos en un proceso. Con frecuencia este nivel de detalle no es necesario, pero cuando se necesita, el equipo completo de desarrollo normalmente realizará una versión de arriba hacia abajo, luego grupos de trabajo más pequeños pueden agregar niveles de detalle según sea necesario durante el proyecto.

Realizar un diagrama de flujo es importante porque identifica pasos innecesarios y círculos de duplicación de trabajo.

La metodología para preparar un diagrama de flujo es:

1. Propósito. Analizar cómo se pretende utilizar el diagrama de flujo.

2. Determinar el nivel de detalle requerido.

3. Definir los límites.

4. Utilizar símbolos apropiados.

5. Hacer preguntas. Para cada input (entrada), hacer preguntas como:
   ¿Quién recibe el input?
   ¿Qué es lo primero que se hace con el input?

6. Documentar. Documentar cada paso en la secuencia, empezando con el primero (o último). Para cada uno, hacer preguntas como:
   ¿Qué produce este paso?
   ¿Quién recibe este resultado?
   ¿Qué pasa después?
   ¿Alguno de los pasos requiere inputs que actualmente no se muestran?

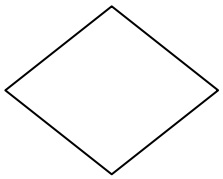
7. Completar. Continuar la construcción del diagrama hasta que se conecten todos los outpouts (resultados), definidos en el diagrama.

8. Revisión. Preguntar:
   ¿Todo el flujo de información encaja en los inputs y outputs del proceso?
   ¿El diagrama muestra la naturaleza serial y paralela de los pasos?
   ¿El diagrama capta de forma exacta lo que realmente ocurre, a diferencia de la forma cómo se piensa que las cosas deberían pasar o cómo fueron diseñadas originalmente?

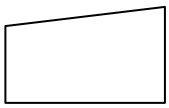
9. Determinar oportunidades.

Aunque hay literalmente docenas de símbolos especializados utilizados para hacer diagramas de flujo, se utilizan con más frecuencia los siguientes:

Un paso o tarea del proceso. Una descripción breve del paso se representa dentro del símbolo.



Punto de verificación o de decisión. Este diamante indica un punto de la rama en el proceso. La descripción está escrita dentro del símbolo, generalmente en la forma de una pregunta. La respuesta a la pregunta determina el camino que debe tomarse desde el símbolo de decisión. Cada camino está identificado para que corresponda a una respuesta.

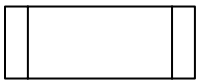


Entrada manual de datos.

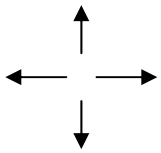
Cola o punto de espera.

Punto de Almacenamiento

Sub-proceso

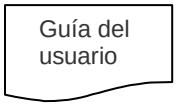


Proceso predefinido

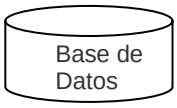


Las líneas de flujo son utilizadas para representar el progreso de los pasos en la secuencia. La punta de la flecha indica la dirección del flujo del proceso.

Otros dos símbolos que no son utilizados tan comúnmente y que pueden ser útiles son:



El símbolo del documento representa la información escrita pertinente al proceso.

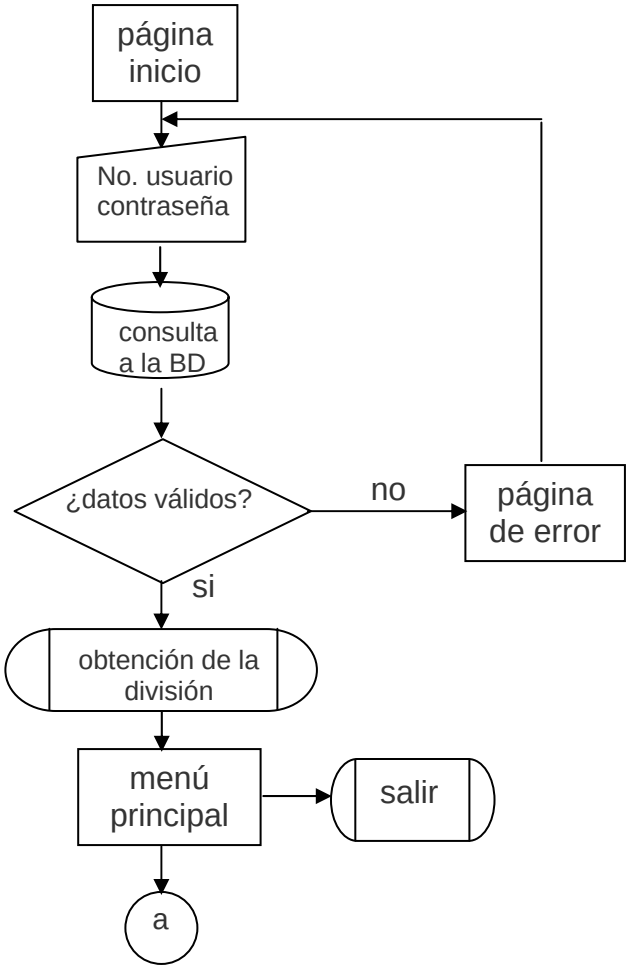


El símbolo de la base de datos representa información almacenada electrónicamente con respecto al proceso.

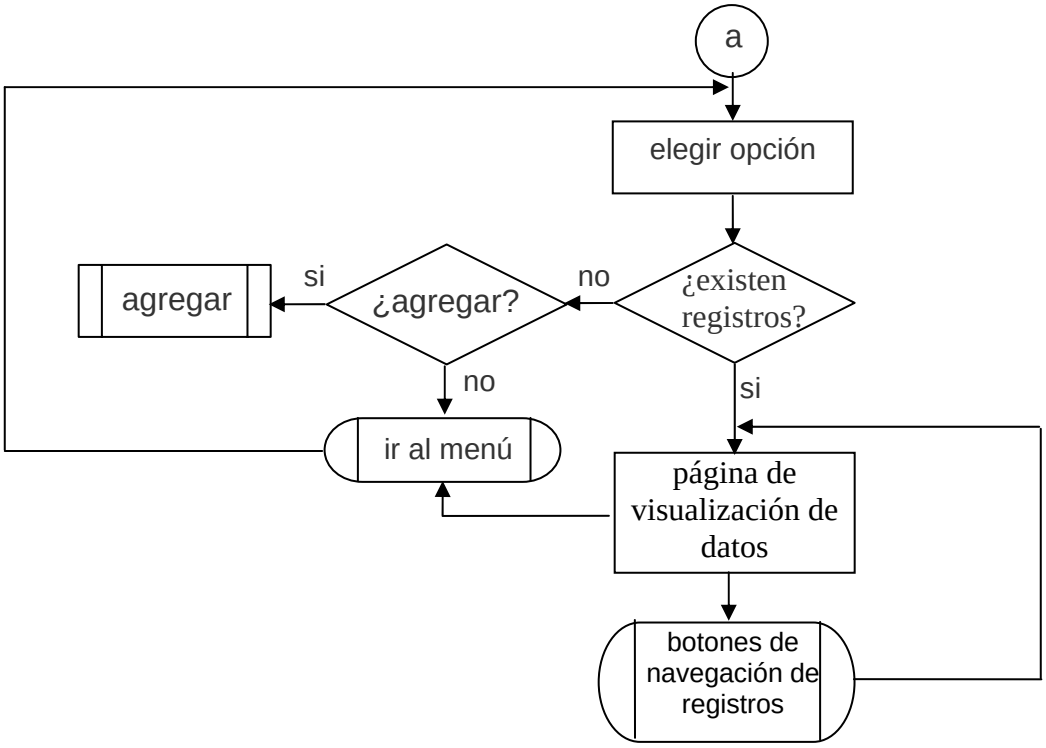
En estos dos últimos, el título o la descripción aparece dentro del símbolo.

Los diagramas de flujo para describir los procesos principales del SAILFI: validación de contraseñas, agregar, modificar, consultar y eliminar registros, se muestran a continuación:

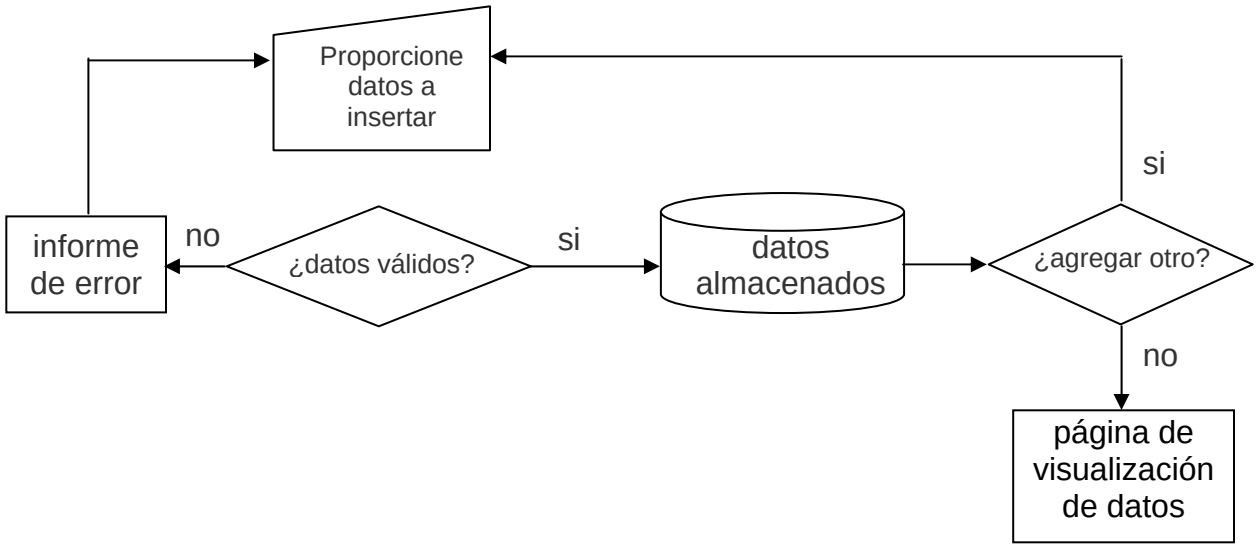
Para validar contraseñas



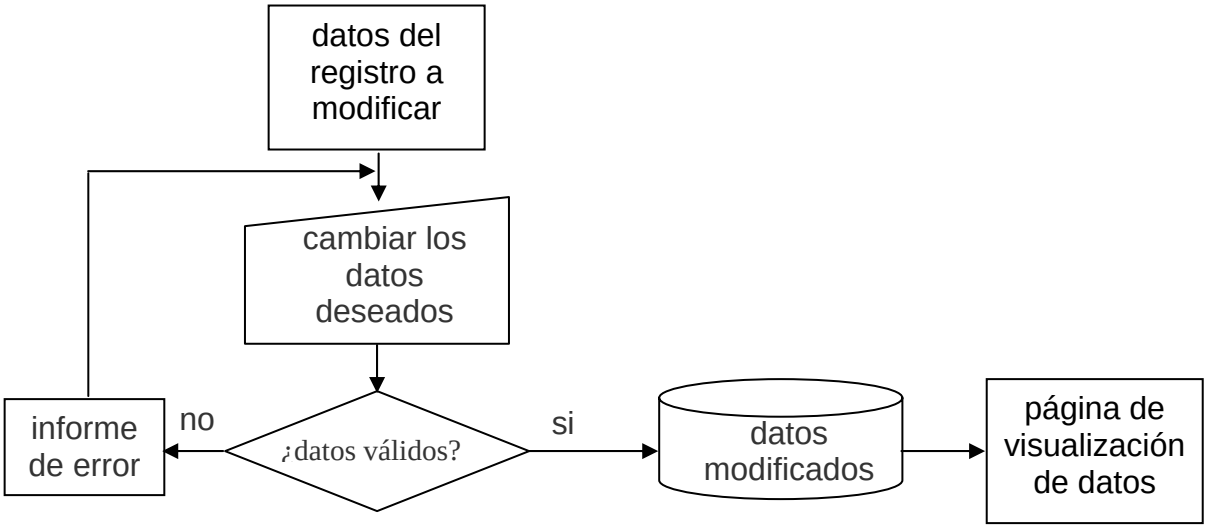
Consultar



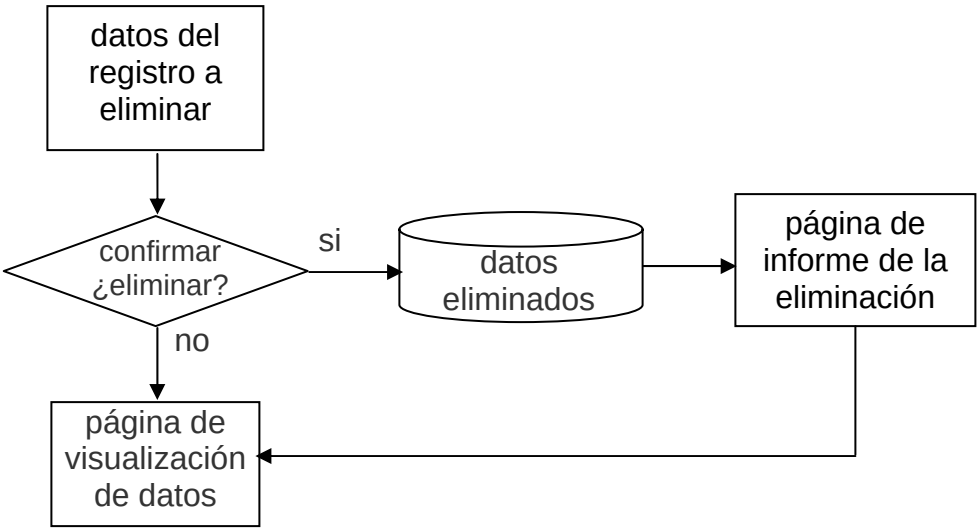
Agregar



Modificar



Eliminar



3.2 Modelo de la base de datos

Un Modelo de base de datos es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Una base de datos es el conjunto de datos almacenados con una estructura lógica. Es decir, tan importante como los datos, es la estructura conceptual con la que se relacionan entre ellos. En la práctica podemos pensar esto como el conjunto de datos más el programa (software) que hacen de ellos un conjunto consistente. Si no tenemos los dos factores unidos, no podemos hablar de una base de datos, ya que la combinación de ambos da la coherencia necesaria para poder trabajar con los datos de una manera sistemática.

La forma física como estén almacenados los datos es independiente del concepto que tengamos de ellos. El conjunto de programas que saben como traer, unir y mostrar los datos, así como aquellos encargados de almacenarlos, son los que le dan coherencia al concepto base de datos.

Los modelos de datos se dividen en tres grupos:

- Modelos lógicos basados en objetos.

Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.

- Modelos lógicos basados en registros.

Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los tres modelos de datos más ampliamente aceptados son:

- Modelo Relacional

  Como su nombre lo dice, su estructura principal es la relación, es decir una tabla bidimensional compuesta por líneas y columnas.

Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos. Entidades con características comunes, es decir descritas por el mismo conjunto de atributos, formarán parte de la misma relación.

- Modelo Jerárquico

En el modelo jerárquico los datos se organizan en grupos unidos entre ellos por relaciones de "posesión", en las que un conjunto de datos puede tener otro conjunto de datos, pero un conjunto puede pertenecer sólo a otro conjunto. La estructura resultante es un árbol de conjuntos de datos.

- Modelo de Red

El modelo de red o reticular es muy parecido al jerárquico, y de hecho nace como una extensión de este último. También en este modelo los conjuntos de datos están unidos por relaciones de posesión, pero cada conjunto de datos puede pertenecer a uno o más conjuntos. La estructura resultante es una red de conjuntos de datos

Sin embargo en los últimos años, se han propuesto algunas extensiones al modelo relacional como respuesta a la creciente complejidad de las aplicaciones que requieren bases de datos y han surgido dos nuevos modelos: el modelo de datos orientado a objetos y el modelo relacional extendido cuyo objetivo es capturar mejor el significado de los datos, para disponer de los conceptos de la orientación a objetos y de capacidad deductiva. Sin embargo, a diferencia de los modelos que los preceden, la composición de estos modelos no está clara. Esta evolución representa la tercera generación de los *DBMS* (Data Base Management System, Sistema de Administración de Base de Datos).

Modelo de datos orientado a objetos (object-oriented)

El esquema de una base de datos por objetos está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que poblarán la base de datos. La diferencia principal respecto a los modelos examinados anteriormente radica en que con una base de datos tradicional (entendiendo con este término cualquier base de datos no por objetos), las operaciones que se tienen que efectuar en los datos se les piden a las aplicaciones que los usan y en una base de datos object-oriented, los objetos memorizados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. Esto les permite saber cómo comportarse, sin tener que apoyarse en aplicaciones externas.

- Modelos físicos de datos.

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos.

Existen dos clasificaciones de este tipo que son:
- Modelo unificador
- Memoria de elementos

Para el SAILFI se utilizó el modelo relacional por lo cual se describe a continuación con un poco más de detalle.

El *modelo relacional* fue propuesto originariamente por E.F. Codd en un artículo de 1970. Gracias a su coherencia y facilidad de uso, el modelo se convirtió en los años 80 en el más usado para la producción de DBMS.

La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tupla) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

El modelo relacional representa la segunda generación de los DBMS. En él, todos los datos están estructurados en el ámbito lógico como tablas formadas por filas y columnas, aunque de manera física pueden tener una estructura completamente distinta.

Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los DBMS de la primera generación, lo que constituye otro punto a su favor.

Dada la popularidad del modelo relacional, muchos sistemas de la primera generación se han modificado para proporcionar una interfaz de usuario relacional, con independencia del modelo lógico que soportan. (de red o jerárquico)

El término "Relacional" en *RDBMS (*Relational Database Management System) hace referencia al hecho de que grupos parecidos de información son almacenados en distintas tablas que luego pueden ser "juntadas" (relacionadas) basándose en los datos que tienen en común.

Los sistemas jerárquico y de red constituyen la primera generación de los DBMS. Pero estos sistemas presentan algunos inconvenientes:

- Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.
- La independencia de datos es mínima.
- No tienen un fundamento teórico.

3.3 Diseño de la base de datos

Uno de los pasos más importantes en la construcción de una aplicación que maneja una base de datos, es sin duda, el diseño de la base de datos. Si las tablas no son definidas apropiadamente, podemos tener muchos problemas al momento de ejecutar consultas a la base de datos para tratar de obtener algún tipo de información.

Una base de datos es una colección de datos que es manejada y organizada por un software específico, el DBMS[1].

Independientemente del tipo de base de datos, las funciones principales que se pueden esperar de un DBMS son:

- Permitir el acceso a los datos a través de un esquema conceptual, en lugar de hacerlo a través de un esquema físico.
- Compartir e integrar los datos entre aplicaciones diferentes.
- Controlar el acceso compartido a los datos.
- Garantizar la seguridad e integridad de los datos.

3.3.1 Diseño conceptual

En esta etapa se debe construir un esquema de la información que utiliza un sistema, independientemente de cualquier consideración física. A este esquema se le denomina *esquema conceptual*. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos a utilizar: encuentran entidades, atributos y relaciones.

El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

[1] Un DBMS es sustancialmente un software que se coloca entre el usuario y los datos.

En este nivel se describe que datos son almacenados realmente en la base de datos y las relaciones que existen entre los mismos, describe la base de datos completa en términos de su estructura de diseño.

El nivel conceptual de abstracción lo usan los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Consta de las siguientes definiciones:

Definición de los datos: Se describen el tipo de datos y la longitud de campo de todos los elementos direccionables en la base, cada uno de éstos presenta sus propias variantes, por lo que la elección del tipo de dato correcto no sólo influye en el tipo de información que se puede almacenar en cada campo, sino que afecta al rendimiento global de la base de datos. Los elementos por definir incluyen artículos elementales (atributos), datos y registros conceptuales (entidades).

Relaciones entre datos: Se definen las relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples.

En el nivel conceptual la base de datos aparece como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad los archivos conceptuales no existen físicamente. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema.

El esquema conceptual se puede utilizar para que el diseñador transmita al cliente lo que ha entendido sobre la información que éste maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación.

Se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario y es completamente independiente de los aspectos de implementación, como puede ser el *DBMS* que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física, además de ser una fuente de información para el diseño lógico de la base de datos.

3.3.2 Diseño lógico

El diseño lógico es el proceso de construir un esquema de la información que utiliza un sistema, basándose en un modelo de base de datos específico, independiente del *DBMS* concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el *DBMS* que se va a utilizar, como puede ser el modelo relacional, el modelo de red,

el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La *normalización* es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional.

Se puede decir que éstos son los principales objetivos de la normalización:

- Controlar la redundancia de la información.
- Evitar pérdidas de información.
- Capacidad para representar toda la información.
- Mantener la consistencia de los datos.

Uno de los objetivos de una estructura de tabla normalizada es minimizar el número de "celdas vacías".

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

3.3.3 Diseño Físico

Es aquí donde se determinan las estructuras de almacenamiento físico y los métodos usados para tener acceso efectivo a los datos de la base de datos. El esquema físico es una descripción de la implantación de una base de datos en memoria. Una etapa de suma importancia es que el modelado de la base se adapta a un DBMS específico, usando el lenguaje de definición de datos para dar origen a la base de datos operacional.

Como se mencionó en el capítulo anterior, se utilizará PostgreSQL como manejador de la base de datos para el SAILFI y se diseñó una base de datos relacional que es el tipo de bases de datos actualmente más difundido. Los motivos del éxito de éstas bases son fundamentalmente dos:

1. Ofrecen sistemas simples y eficaces para representar y manipular los datos.
2. Se basan en el modelo relacional, con sólidas bases teóricas.

A continuación se ejemplifica con la tabla de proyectos como se diseñó la base de datos para el SAILFI:

En la base de datos se tienen que representar proyectos, se podrá definir una relación llamada "Proyectos", cuyos atributos describen las características de los proyectos (figura 3.3.3.1). Cada tupla de la relación "proyectos" representará un proyecto específico.

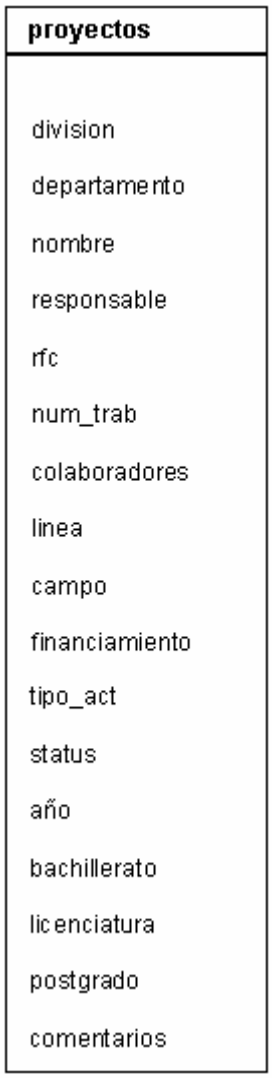


figura 3.3.3.1

Basándonos en el concepto de normalización se observa que llegando al nível 1 de normalización se cumple con la no redundancia y consistencia en los datos, los cuales son los factores de mayor importancia.

3.4 Diccionario de datos

Un Diccionario de datos contiene metadatos, es decir, datos acerca de los datos. El esquema de una tabla es un ejemplo de metadatos. Un sistema de base de datos consulta el Diccionario de Datos antes de leer o modificar los datos reales.

El Diccionario de Datos es una forma de documentación para el diseñador de bases de datos, guarda los detalles y descripciones de todos los datos que forman parte del sistema.

Si los analistas desean conocer cuántos caracteres abarca un determinado dato, qué otros nombres recibe en distintas partes del sistema, o dónde se utiliza, encontrarán las respuestas en un diccionario de datos desarrollado en forma apropiada.

Su utilidad básica es:

- Documentar la estructura interna de cada tabla, incluyendo todos sus campos y sus tipos de datos con comentarios
- Proveer una detallada descripción de cada tabla dentro de la base de datos
- Describir reglas como pueden ser el uso de valores no nulos, valores únicos.
- Incluye también documentación respecto al uso de herramientas para respaldar la información

El Diccionario de Datos del SAILFI quedará como se muestra a continuación, se ejemplifica con las tablas *proyectos* y *actividades:*

TABLA *proyectos*

En ésta tabla se almacenan lo datos que son capturados por el formulario proyectos, los campos son:

id_reg	Identificador del dato o registro varchar(20) NOT NULL
division	División a la que corresponde el proyecto varchar(10) NOT NULL
departamento	Departamento a cargo del proyecto varchar(50) NOT NULL
nombre	Nombre del proyecto de investigación varchar(100) NOT NULL

responsable	Nombre del responsable del proyecto varchar(50) NOT NULL
rfc	RFC del responsable del proyecto varchar(10) NOT NULL
num_trab	Número de trabajador UNAM del responsable del proyecto (INTEGER)
colaboradores	Nombre de los colaboradores del proyecto varchar(300)
linea	Línea de investigación del proyecto varchar(70) NOT NULL
campo	Campo de la ciencia en el cual se desarrolla el proyecto varchar(30) NOT NULL
financiamiento	Fuente de financiamiento del proyecto varchar(60) NOT NULL
tipo_act	Tipo de actividad que se lleva a cabo en el proyecto varchar(25) NOT NULL
status	Status del proyecto (nuevo, continuación, terminado) varchar(12) NOT NULL
año	Año de inicio del proyecto de investigación (SMALLINT)
bachillerato	Número de alumnos participantes de bachillerato (SMALLINT)
licenciatura	Número de alumnos participantes de licenciatura (SMALLINT)
posgrado	Número de alumnos participantes de posgrado (SMALLINT)
comentarios	Información relevante para ampliar o aclara algún concepto varchar(300)

Llave primaria (id_reg)

El código en PostgreSQL es:

```
create table proyectos

( id_reg varchar(20),
 division varchar(10) NOT NULL,
 departamento varchar(50) NOT NULL,
 nombre varchar(150) NOT NULL,
 responsable varchar(50) NOT NULL,
 rfc varchar(10 NOT NULL),
 num_trab integer,
 colaboradores varchar(300),
 linea varchar(70) NOT NULL,
 campo varchar(30) NOT NULL,
 financiamiento varchar(60) NOT NULL,
 tipo_act varchar(25) NOT NULL,
 status varchar(12) NOT NULL,
 año smallint,
 bachillerato smallint,
```

```
licenciatura smallint,
posgrado smallint,
comentarios varchar(300),
PRIMARY KEY(id_reg));
```

Otro factor importante que definir para el diccionario de datos es la manera de respaldar la información.

Para realizar el respaldo de la base de datos se utiliza la siguiente expresión en PostgreSQL:

```
pg_dump SAILFI > Repaldo_BaseSAILFI_fechaderespaldo.bak
```

Esto crea un script que contiene todas las instrucciones necesarias para regenerar la base de datos del SAILFI, con todo y los datos almacenados en las tablas.

Es importante destacar que esto se puede hacer mientras la base de datos permanece en línea y disponible para ser consultada.

Para regenerar la base se utiliza la siguiente expresión:

```
psql SAILFI_NUEVABASE < Repaldo_BaseSAILFI_fechaderespaldo.bak
```

Esto creará todas las tablas que se tenían en la base SAILFI en la nueva base, que en este caso se llamaría SAILFI_NUEVABASE.

Si lo que se quiere es respaldar únicamente los datos de una tabla específica de la base de datos la instrucción que se tiene que ejecutar estando dentro de la base es:

```
\copy NOMBRE_TABLA to 'ARCHIVO_SALIDA'
```

por ejemplo:

```
\copy proyectos to 'respaldo_proyectos.txt'
```

Para volver a cargar los datos de una tabla que se había respaldado de la manera que se mencionó anteriormente en una tabla vacía.

```
\copy NUEVA_TABLA from 'ARCHIVO_TEXTO'
```

es decir:

```
\copy proyectos_nuevatabla from ‘respaldo_proyectos.txt'
```

Capítulo 4

Desarrollo del sistema

Desarrollo del sistema

Es en esta etapa en donde se pasa del diseño conceptual al diseño físico del sistema. Dicho en otras palabras, en esta etapa se conjunta el análisis y definición de requerimientos, diseño del sistema y del programa y llegar así a la codificación (implementar los programas en el lenguaje de programación).

Los programadores tienen un papel principal en esta etapa, ya que son los encargados de la codificación de los módulos correspondientes, así como también de la verificación de sintaxis en el código, para encontrar errores que son resueltos por ellos mismos. Como se verá más adelante, el programador también valida cada uno de los módulos programados y realiza pruebas necesarias al sistema para su exitosa implantación.

4.1 Descripción de los procesos

Antes de desarrollar el sistema se deben identificar los procesos que éste debe llevar a cabo.

Se puede decir que un proceso es una serie de pasos o tareas ordenadas que involucran actividades, restricciones y recursos que producen una determinada salida.[2]

Además un proceso tiene criterios de entrada y salida, de modo que se conoce cuándo comienza y termina una actividad.

A continuación se describe un proceso que es muy importante en el desarrollo del sistema porque da la pauta para poder utilizarlo, debido a que es la parte que los usuarios manipularán.

4.1.1 Diseño de la interfaz de usuario

El diseño de la interfaz de usuario puede tener dificultades, dado que personas diferentes tienen estilos diferentes de percepción, comprensión y trabajo. Por ejemplo, un usuario puede hacer más uso del teclado y otro confiar plenamente del mouse.

[2] Ingeniería de Software, Teoría y Práctica. Shari Lawrence Pfleeger.

Marcus[3] representa muchas de las cuestiones involucradas en el diseño de la interfaz de usuario. Puntualiza que una interfaz debe atender algunos elementos clave, como son:

- metáforas: términos fundamentales, imágenes y conceptos que deben ser reconocidos y aprendidos.
- un modelo mental: la organización y representación de datos, funciones, tareas y roles.
- las reglas de "navegación" para el modelo: cómo moverse dentro de los datos, funciones, actividades y roles.
- aspecto: las características de la presentación del sistema que transmiten información al usuario.
- sensación: las técnicas de interacción que proporcionan una experiencia atractiva para el usuario.

La meta de estos elementos y de la interfaz de usuario es contribuir a que éste consiga un rápido acceso al contenido de sistemas complejos, sin perder la comprensión mientras se desplaza a través de la información[4].

Para el desarrollo del SAILFI se utiliza lenguaje PHP y HTML que permiten hacer una interfaz amigable para los usuarios. Además se tratan de cubrir la mayoría de los puntos requeridos para una interfaz eficiente.

Es importante recordar que el SAILFI está diseñado para ser consultado a través de la web, por lo que requiere de seguridad para su acceso, por otro lado la información que manipulará un usuario sólo será la de su correspondiente división.

Debido a esta característica se utiliza el modelo cliente / servidor, el cual se explica brevemente.

Modelo cliente / servidor

El funcionamiento de las páginas web es básicamente el siguiente. Al introducir una dirección web lo que estamos haciendo es pedir un archivo localizado en una computadora (que actuará de servidor). El servidor nos enviará este archivo y nuestro navegador (el programa cliente) se encargará de interpretarlo para que nos aparezca la página web (que será más o menos vistosa) en pantalla.

Cuando las *páginas* son *dinámicas* el servidor es más complejo, ya que consta de dos bloques: uno que envía la página web y otro bloque que genera la información dinámicamente, por ejemplo, una base de datos.

[3] Ingeniería de Software, Teoría y Práctica. Shari Lawrence Pfleeger.
[4] Es importante recordar que, generalmente, el manual del usuario se basa en la presentación de pantallas para una fácil descripción de las tareas.

Generalmente cuando el servidor cuenta con un bloque inferior que permite generar servicios web se le conoce como servidor de aplicaciones.

Es decir, el modelo cliente / servidor es una relación entre procesos corriendo en máquinas separadas, donde:

- El servidor (s) es un proveedor de servicios.
- El cliente (c) es un consumidor de servicios.

ambos interactúan por un mecanismo de pasaje de mensajes: Pedido de servicio / respuesta.

Es una forma de dividir y especializar programas a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificar las actualizaciones y mantenimiento del sistema.

En la figura 4.1.1.1 se muestra un diagrama del modelo c/s.

figura 4.1.1.1 Modelo cliente / servidor

Para utilizar este modelo 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 (interfaz)
- Procesos
- Almacenamiento de la Información (BD)

Los cuales se suelen distribuir como se muestra en la figura 4.1.1.2

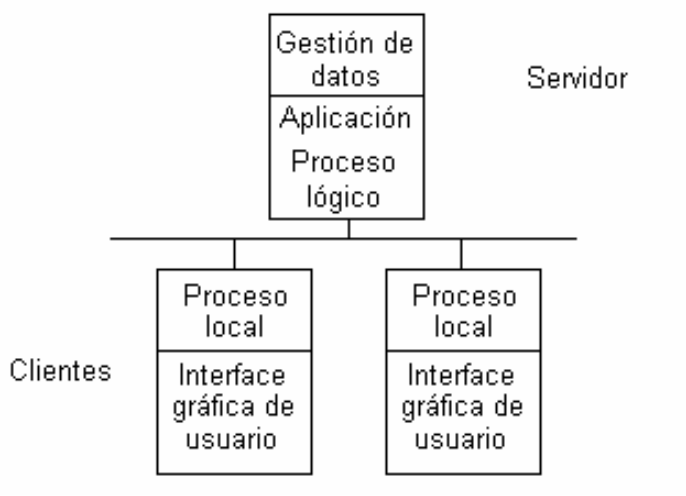


figura 4.1.1.2 Aplicaciones cliente / servidor

La ventaja de un sistema cliente / servidor es que los usuarios extraen la información que necesitan sólo cuando la utilizan, reduciendo considerablemente el tráfico de información en la de red. Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre.

Por otro lado un servidor puede atender las peticiones y controlar el acceso a los recursos de varios clientes a la vez.

Uno de los aspectos que más ha promovido el uso de sistemas cliente / servidor, es la existencia de plataformas de hardware cada vez más económicas. Ésta constituye a su vez una de las más importantes ventajas de este esquema, porque da la oportunidad de utilizar máquinas considerablemente más económicas que las requeridas por otros modelos.

El modelo cliente / servidor facilita el suministro de información a los usuarios. Esto es, por un lado proporciona una mayor consistencia a la información de la organización al contar con un control centralizado de los elementos compartidos, y por otro, facilita la construcción de interfaces gráficas dinámicas, las cuales pueden hacer que los "datos" se conviertan en "información".

En los sistemas c/s, el acceso es frecuentemente limitado a proteger ciertos datos en un servidor dado, ya que, no hay ningún control físico para el control de los datos usados por múltiples PCs, excepto para controlar el medio físico de almacenaje, sin embargo los usuarios no tienen acceso a esta opción de tal manera que la validación de contraseñas es uno de los procesos de protección del sistema.

En el capítulo anterior se representaron con diagramas los principales procesos para el manejo de datos del sistema que, además de la validación de contraseñas, son: agregar, modificar, consultar y eliminar. A continuación se describen dichos procesos.

4.1.2 Validación de contraseñas

La validación es muy importante porque es el proceso que permite el acceso al sistema, los datos requeridos son: Número de usuario y Contraseña. Ambos serán proporcionados por el administrador del sistema. Por otra parte, como dato de salida del proceso se tendrá la división correspondiente al usuario que ingresa.

En este proceso lo que se hace es presentar una pantalla al usuario pidiéndole su número de usuario y su contraseña. La pantalla es la mostrada en la figura 4.1.2.1.

figura 4.1.2.1 Interfaz para la validación de usuarios.

Cuando el usuario de clic en continuar se procede a:

- Verificar que ambos datos hayan sido proporcionados.
- Validar que sean del tipo correcto; es decir, que el número de usuario sea un número y no otro carácter.
- Consultar en la base de datos si la información proporcionada puede acceder al sistema.

En caso de no cumplir con uno o más de estos puntos se muestra una página de aviso de error. En caso contrario, se identifica a que división corresponden, ya que el usuario sólo puede ver la información referente a ésta, y se muestra el menú principal.

4.1.3 Consulta

El proceso de consulta se efectúa cuando el usuario desea ver la información que existe en la base de datos referente a un apartado específico, en esta parte no es necesario que proporcione datos, sólo debe acceder la opción que le interesa para poder visualizar los datos correspondientes. Los datos de salida de este proceso son los referentes a la consulta.

Una vez que el usuario ingreso al sistema deberá seleccionar en el menú principal la información que desea consultar. Para esto sólo necesita hacer clic sobre la opción que requiera en las pantallas que se le irán presentando.

El sistema se conecta al servidor de base de datos y ejecuta la consulta para la búsqueda de registros; al obtener los datos cierra la conexión al servidor (esto ayuda a que disminuya el tráfico de información en la red) y carga los datos en el formulario.

Cuando los datos son consultados no pueden ser modificados desde esta pantalla.

La figura 4.1.3.1 muestra la pantalla de consulta para el caso Publicaciones Editoriales Institucionales (Boletines).

figura 4.1.3.1 Pantalla para seleccionar la consulta de boletines

Los datos se presentarán en pantalla como se muestra en la figura 4.1.3.2.

figura 4.1.3.2 Pantalla de consulta

Los botones Primero, Siguiente, Anterior y Último, permiten al usuario "*navegar*" entre los registros.

También se tienen botones para los procesos Modificar, Eliminar y Agregar.

En caso de no existir datos en la opción requerida, se visualizará una pantalla indicando que no hay registros para la consulta (figura 4.1.3.3) se tendrá un botón de Agregar, para ingresar datos.

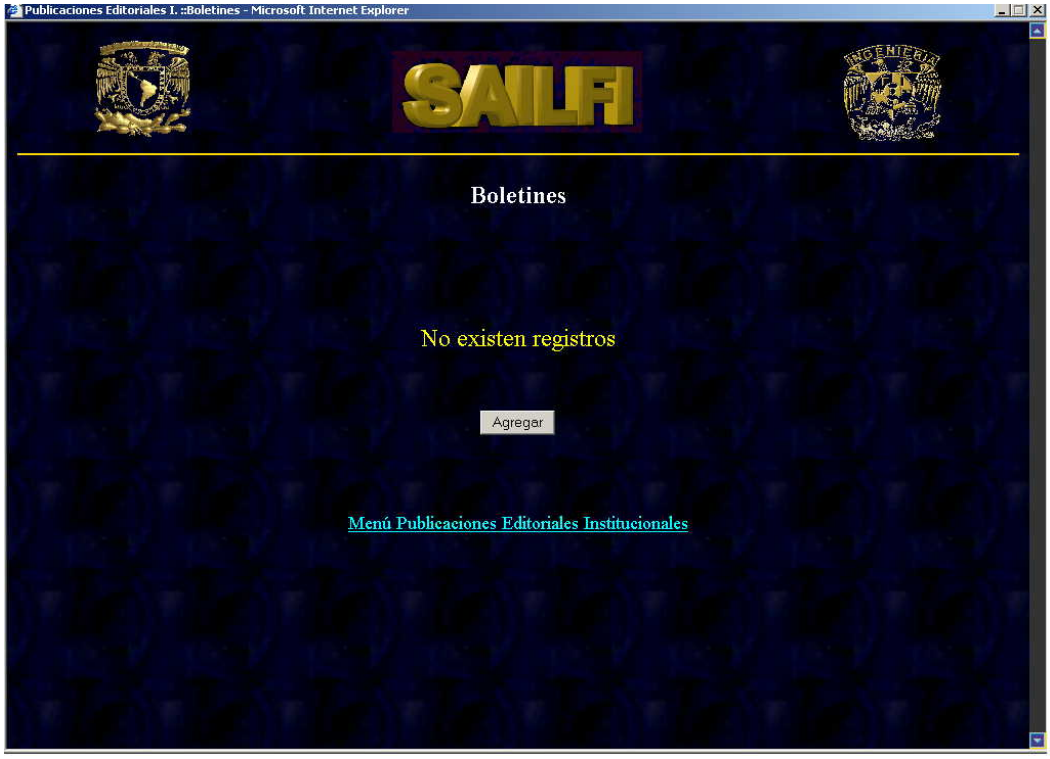


figura 4.1.3.3 Consulta sin registros para mostrar.

4.1.4 Agregar

Este proceso tiene como finalidad capturar datos, referente a las instancias que maneja el sistema. Es necesario que el usuario proporcione los datos requeridos por el formulario (datos de entrada) para poder ser almacenados exitosamente en la base de datos.

El proceso para agregar datos comienza cuando el usuario da clic sobre el botón Agregar o Agregar otro y se lleva a cabo de la siguiente manera:

- Se presenta el formulario con los campos en blanco para que se introduzcan los datos, excepto el campo división, que ya viene con el dato correspondiente de acuerdo a la contraseña que el usuario proporcionó a su ingreso. Ver figura 4.1.4.1.

- Se muestra un botón de ayuda[5] para que el usuario consulte, si tiene duda, el tipo de dato que debe introducir en los campos del formulario.

- Cuando el usuario haga clic en el botón Restaurar los datos que haya ingresado se perderán y el formulario estará limpio.

[5] Este botón de ayuda es el que se refiere a la ayuda en línea, ver capítulo 6.3.

- Si el usuario opta por el botón Guardar los datos son validados para que no se introduzca algún tipo de dato incorrecto en la base de datos. En caso de encontrar incorrecto uno o más datos se muestra un botón para corregir.

figura 4.1.4.1 Pantalla para agregar registros

Al ser válidos los datos se hace la conexión al servidor y son enviados para almacenarse, se cierra la conexión y se muestran un botón de Agregar otro y una liga para regresar.

4.1.5 Modificar

El proceso de modificación de datos consiste en efectuar cambios en los valores de algún (nos) campo (s) de un registro en la base de datos, los datos de entrada al proceso son los provenientes de la consulta. Los de salida son los datos modificados después de ejecutar la sentencia de actualización.

Al dar clic sobre el botón Modificar se presentará en pantalla el formulario con los datos correspondientes al registro seleccionado y los botones Guardar Cambios y Restaurar. (ver figura 4.1.5.1)

El usuario realizará los cambios en los datos que requiera modificar.

Si presiona Restaurar aparecerán en el formulario los datos que tenia el registro cuando se hizo la consulta.

Cuando se presiona el botón Guardar Cambios, los datos son validados de la misma manera que cuando se agrega y el proceso es el mismo.

figura 4.1.5.1 Pantalla para modificar datos.

4.1.6 Eliminar

El proceso eliminar tiene como objetivo “quitar” permanentemente de la base de datos el registro que el usuario desea. Los datos del registro a eliminar son considerados datos de entrada al proceso.

Para eliminar un registro se deberá hacer clic sobre el botón Eliminar en la pantalla de consulta.

Aparecerá en pantalla el formulario con los datos y los botones de Cancelar y Confirmar. (ver figura 4.1.6.1)

Cuando se seleccione Cancelar el usuario regresará a la pantalla donde estaba realizando la consulta y podrá seguir navegando entre los registros o realizar otro proceso.

Cuando se oprime el botón Confirmar se abre la conexión a la base de datos, se envía la instrucción SQL correspondiente a la eliminación y después de ello se

cierra nuevamente la conexión a la base de datos. Por último notifica al usuario la eliminación del registro.

figura 4.1.6.1 Pantalla para eliminar un registro.

Los pasos que lleva a cabo un sistema que se apoya en un servidor de aplicaciones y uno de base de datos (como es el caso del SAILFI), para ser visualizados por un usuario en un *navegador web* se resumen en la figura 4.1.6.2:

4.2 Pruebas e implantación

Durante el proceso de implantación y prueba se deben desarrollar todas las estrategias posibles para garantizar que en el uso inicial el sistema se encuentre libre de problemas, lo cual se puede descubrir durante este proceso y llevar a cabo las correcciones pertinentes para su buen funcionamiento.

Las pruebas de los sistemas no siempre reciben la atención que merecen, sin embargo cuando se llevan a cabo de manera adecuada proporcionan mucha información que puede ayudar a mejorar la efectividad de los esfuerzos de desarrollo de aplicaciones futuras.

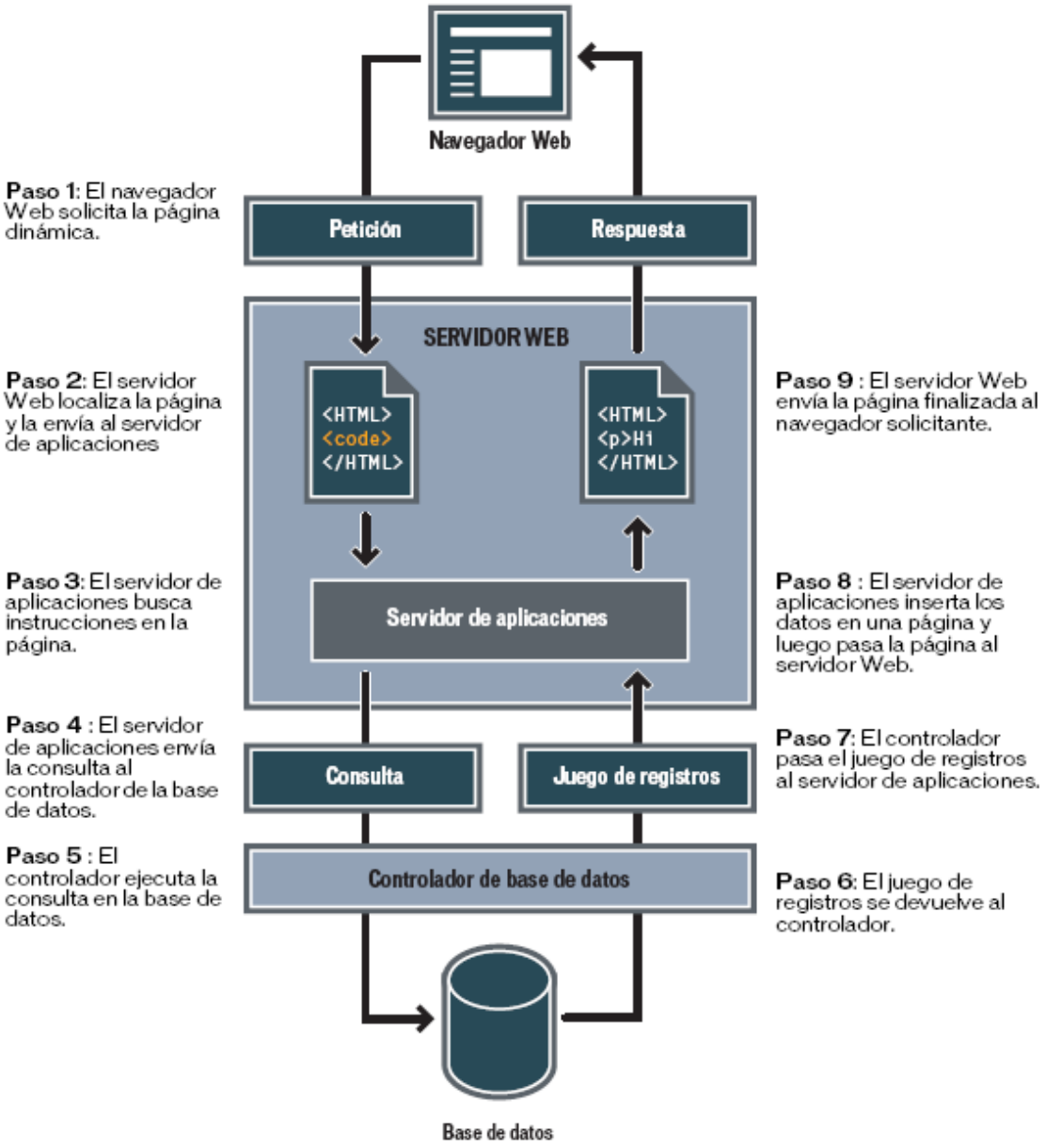


figura 4.1.6.2 Proceso para visualizar una página Web dinámica. [6]

4.2.1 Pruebas al Sistema

El ciclo de vida del desarrollo requiere varios niveles de prueba, comenzando con la prueba unitaria y de integración para posteriormente demostrar la funcionalidad completa del sistema.

[6] Tomado del Manual Primeros pasos en DW MX

La etapa de prueba se concentra en la búsqueda de defectos, en este apartado se explica la prueba individual de componentes, después la de integración para comprobar las interfaces y el sistema en su totalidad.

La prueba conocida como **prueba de módulo, prueba de componente o prueba unitaria,** verifica que el componente funciona correctamente con los tipos de entrada esperados a partir del estudio del diseño del componente. La prueba unitaria se hace, siempre que sea posible, en un ambiente controlado de modo que el responsable de realizar la prueba pueda ingresarle al componente que se está probando un conjunto predeterminado de datos y observar que acciones y datos de salida se producen.

Cuando el conjunto de componentes del sistema (o subsistema) ha superado la prueba unitaria, el paso siguiente es asegurar que las interfaces entre los componentes están definidas y se manejan correctamente, además de efectuarse la **prueba de integración.** En ésta se desarrolla el proceso de verificar que los componentes del sistema trabajen juntos conforme a lo descrito en las especificaciones de diseño.

En la figura 4.2.1.1 se ilustra esta etapa de prueba

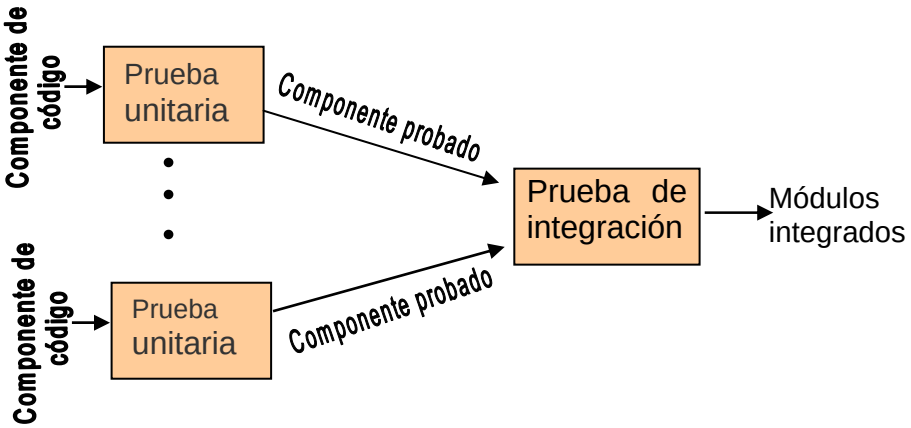


figura 4.2.1.1 Pasos para la prueba de integración

Sin embargo, ya integrados los módulos se realizan otras pruebas para revisar el sistema en conjunto, entre éstas están:

La **prueba de función** verifica que el sistema integrado realice las funciones especificadas en los requerimientos. Una vez que se considera que las funciones trabajan correctamente se continua con las demás pruebas.

La **prueba de rendimiento** compara el rendimiento de los componentes integrados con los requerimientos no funcionales, estos requerimientos que incluyen la seguridad, exactitud, velocidad y confiabilidad, restringen la manera en que se realizan las funciones del sistema. Para el SAILFI se realizaron las pruebas

de seguridad y de restricción de funciones.

Cuando la prueba se lleva a cabo exitosamente y el sistema trabaja como se estableció en el diseño, se le denomina **sistema verificado** y luego se compara con las expectativas del cliente. Si se comprueba satisfactoriamente que el sistema que se ha construido cumple los requerimientos, se tiene un sistema **validado.**

La **prueba de aceptación** se hace en conjunto con el cliente; el sistema se comprueba contra la especificación de requerimientos realizada anteriormente.

La prueba de aceptación en ocasiones se realiza en su ambiente real pero a menudo se ejecuta en un una ubicación diferente a la definitiva. Es por esta razón que al completar la prueba de aceptación el sistema aceptado se instala en el ambiente en el que será utilizado y se ejecuta una última **prueba de instalación** para permitir que los usuarios trabajen con él ya puesto el sistema en el sitio real.

En la figura 4.2.1.2 se ilustra la relación entre los pasos de prueba mencionados anteriormente.

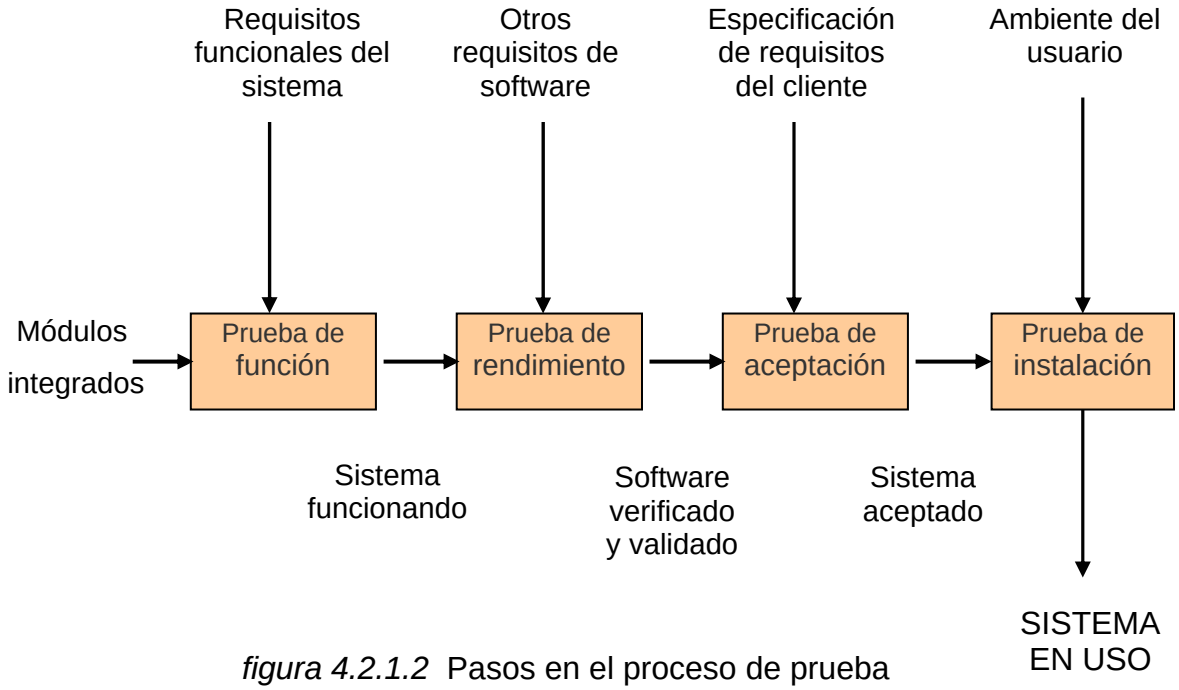


figura 4.2.1.2 Pasos en el proceso de prueba

Sin importar el tamaño del sistema, el tipo de prueba descrito en cada paso es necesario para asegurar su correcto funcionamiento.

Dependiendo del tamaño de la empresa que usará el sistema y el riesgo asociado a su uso, puede hacerse la elección de comenzar la operación del sistema sólo en un área de la empresa (como una prueba piloto), que puede llevarse a cabo en un departamento o con una o dos personas. Cuando se implanta un nuevo sistema lo

aconsejable es que el viejo y el nuevo funcionen de manera simultánea o paralela con la finalidad de comparar los resultados que ambos ofrecen en su operación, además dar tiempo al personal para su capacitación y adaptación al nuevo sistema.

4.2.2 Implantación

El proceso de implantación constituye la última etapa de la metodología de desarrollo del proyecto y es posterior al proceso de prueba. A pesar de todo el trabajo requerido para llegar a este punto, la fase de implantación puede ser la más difícil.

La implantación es el proceso en el cual se instala el software en los equipos, como resultado de un análisis y diseño previo.

Es importante señalar que, si se trata de un producto estándar, la implantación puede ser relativamente fácil debido a que los usuarios también pueden estar relativamente familiarizados con el sistema si no difiere mucho del que se utilizaba con anterioridad.

Sin embargo, cuando se trata de una nueva tecnología, que no ha sido aplicada con anterioridad o difiere sustancialmente de las prácticas previas, el proceso de implantación debe ser manejado con extremo cuidado y mucha atención en los detalles.

Al implantar el nuevo sistema lo primero que debemos hacer es asegurarnos que el sistema es operacional, es decir que funciona de acuerdo a los requerimientos del análisis y permite a los usuarios operarlo adecuadamente.

Entre los objetivos de la implantación se encuentran:

- Darle responsabilidad al grupo encargado de administrar el sistema.
- Formular medidas de desempeño con las cuales evaluar a los usuarios.
- Entregar formalmente la versión definitiva del sistema.

Para que el sistema funcione correctamente es necesario llevar a cabo una buena implantación y ejecución.

Para la implantación del SAILFI en los servidores (o cuentas) definitivos se consideraran los siguientes pasos:

- Verificar que todos los archivos necesarios para el funcionamiento del sistema, se encuentren en una misma carpeta. La cual será copiada al servidor donde se alojará el sistema.

- Revisar que el servidor cuente con todos los componentes necesarios, en su versión requerida, para la ejecución del sistema. Dichos componentes son: Apache y PHP por la parte del servidor web; y PostgreSQL en el servidor de base de datos.
- Crear una cuenta en ambos servidores (el de base de datos y el de web) para la instalación del sistema, el administrador será quien utilice estas cuentas.
- Una vez implantado el sistema verificar que funcione correctamente en el servidor y que este listo para ser utilizado. (Prueba de instalación)
- Mostrar la operación del nuevo sistema a los interesados o implicados.

Capítulo 5

Mantenimiento

Mantenimiento

Aún cuando el mantenimiento es de las últimas etapas en el ciclo de vida del software, las actividades realizadas en esta etapa no son las menos importantes. Por el contrario, se ha convertido en la principal actividad en cuanto a recursos necesarios y costos, además de ser dependiente de los requerimientos del usuario y empezar con ellos.

Según la terminología ANSI-IEEE, el mantenimiento del software es: *"la modificación de un producto software después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de entorno".*[7]

Las actividades del mantenimiento son similares a aquéllas del desarrollo: análisis de los requerimientos, evaluación del diseño del sistema y del programa, elaboración y revisión del código, comprobación de los cambios y puesta al día de la documentación.

La aparición de nuevas tecnologías, productos y servicios, así como las constantes transformaciones en el mundo de los negocios han generado que el mantenimiento del software cobre una enorme importancia haciendo necesario no sólo el adaptar los sistemas a los nuevos requerimientos de las empresas, sino también la continua vigilancia de su correcto funcionamiento.

Generalmente un grupo separado de analistas, programadores y diseñadores (a veces se incluye uno o dos miembros del equipo de desarrollo) se designa como equipo de mantenimiento. Un equipo nuevo, puede ser más objetivo que los diseñadores originales. A un equipo separado le puede resultar más fácil distinguir entre cómo debe trabajar un sistema y cómo funciona realmente. El personal de mantenimiento no sólo debe entender el software tal como es en un momento dado, sino también como ha sido antes y hacia dónde evolucionará en el futuro.

Todos los problemas que se manifiestan al mantener un sistema contribuyen al elevado costo de mantenimiento del software. En los años setenta, la mayoría del presupuesto de un sistema de software se gastaba en el desarrollo. La relación entre el dinero invertido en desarrollo y el dinero destinado al mantenimiento se revirtió durante los años ochenta estimando para el mantenimiento entre un 40 y un 60 por ciento del costo del ciclo de vida completo de un sistema. Las estimaciones para un sistema en el año 2004 sugieren que los costos de mantenimiento pueden haberse incrementado hasta alcanzar el 80 por ciento del costo del ciclo de vida. Cuanto más complejo es el código, es mayor el esfuerzo requerido para mantenerlo.

[7] http://alarcos.inf-cr.uclm.es/doc/mso/

Una manera de reducir el esfuerzo de mantenimiento es implementar la calidad desde el comienzo. Tratar de imponer buen diseño y estructura dentro de un sistema ya construido no resulta tan exitoso como lo es construir el sistema correctamente desde el comienzo.

5.1 Revisiones periódicas

Después que los sistemas han sido implantados, se les debe seguir dando mantenimiento para asegurar que continúen operando en el nivel mostrado durante la etapa de prueba. Las rutinas de mantenimiento variarán de acuerdo con el tipo y complejidad de la tecnología. Los fabricantes o proveedores suelen indicar en muchos productos el programa o calendario de mantenimiento requerido. El mantenimiento también puede ser realizado por el fabricante o el proveedor como parte del acuerdo de compra.

Además las instrucciones técnicas complementarias especificarán las inspecciones con carácter oficial exigidas por el programa y las revisiones no oficiales, necesarias para comprobar que siguen conservándose las condiciones de seguridad en el sistema.

Deberá por otra parte quedar constancia, en la institución en el lugar donde está instalado el servidor (la máquina) además de en los lugares donde se ejecuta el programa (clientes), de las revisiones no oficiales requeridas, con indicación del técnico que las ha llevado a efecto y de sus resultados.

Las revisiones periódicas detectan desvíos y proponen mejoras, así como la elaboración y actualización de la documentación técnica de la aplicación, para reflejar en todo momento la situación real del software, las modificaciones realizadas y las adiciones incorporadas al sistema. Además aseguran la aplicación del nuevo proceso y su realimentación para una mejora continua.

Puede ser necesario instalar sistemas de monitoreo o prueba para asegurar que las necesidades de mantenimiento sean identificadas y satisfechas cuando resulte necesario. Cuando los sistemas son de uso prolongado, se puede establecer un mecanismo para recibir retroalimentación de los usuarios como otra forma de determinar las necesidades de mantenimiento y modificación.

Cuando se realicen modificaciones a los programas como resultado de ejercicios de mantenimiento o actualización, puede ser necesario promover rondas adicionales de verificación y prueba del sistema para asegurarse que siguen cumpliendo las normas exigidas.

5.2 Medidas preventivas y correctivas

El mantenimiento se puede dividir en:

Mantenimiento preventivo

El mantenimiento preventivo involucra la modificación de algún aspecto del sistema a fin de *prevenir las fallas*. Por lo general el mantenimiento preventivo se realiza a partir del momento en que el programador o el a analizador del código encuentran un defecto real o potencial, que todavía no se ha manifestado como falla, y encara las acciones necesarias para corregir el defecto antes de que se produzca el daño.

El mantenimiento preventivo consta de:

- Revisión y configuración de sistemas operativos monousuario y de red.
- Prevención de posibles infecciones causadas por virus informáticos.
- Diagnóstico de posibles fallas en el software.

Mantenimiento perfectivo

Los cambios en la especificación, normalmente debidos a cambios en los requerimientos de un producto software, implican un nuevo tipo de mantenimiento llamado perfectivo. La razón es muy variada, desde algo tan simple como cambiar el formato de impresión de un informe, hasta la incorporación de un nuevo módulo funcional. Se puede definir el mantenimiento perfectivo como el conjunto de actividades para *mejorar o añadir nuevas funcionalidades* requeridas por el usuario.

Algunos autores dividen este tipo de mantenimiento en dos:

- Mantenimiento de ampliación: orientado a la incorporación de nuevas funcionalidades.
- Mantenimiento de eficiencia: que busca la mejora de la eficiencia de ejecución.

Este último tipo de mantenimiento consiste en la modificación del software para *mejorar sus propiedades* (por ejemplo, aumentando su calidad y/o su mantenibilidad) sin alterar sus especificaciones funcionales.

Algunas maneras de hacerlo son:

- incluir sentencias que comprueben la validez de los datos de entrada
- reestructurar los programas para mejorar su legibilidad, o
- incluir nuevos comentarios que faciliten la posterior comprensión del programa

En algunos casos se ha planteado el mantenimiento para la reutilización, el cual consiste en modificar el software (buscando y modificando componentes para incluirlos en bibliotecas) para que sea mas fácilmente reutilizable. En realidad este tipo de mantenimiento es preventivo, especializado en mejorar la propiedad de reusabilidad del software.

El mantenimiento perfectivo es el tipo más habitual (ver figura 5.2.1).

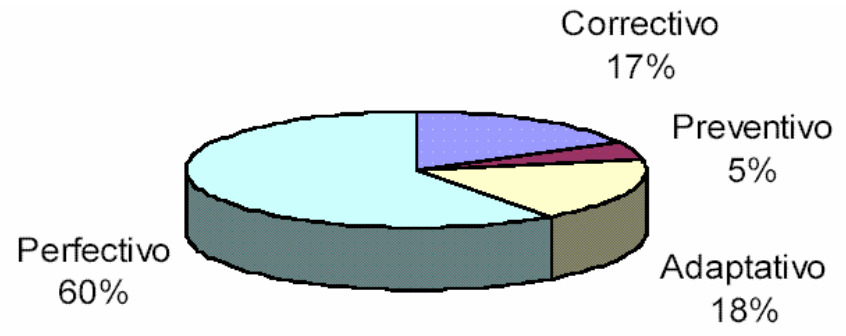


figura 5.2.1. Distribución porcentual de los diferentes tipos de mantenimiento

Mantenimiento correctivo

A medida que ocurren fallas, se ponen a consideración del equipo: se encuentra la causa de la falla y se hacen las correcciones y los cambios a los requerimientos, el diseño, el código, las sesiones de prueba y la documentación, según sea necesario. A menudo, la reparación inicial es temporal: como para mantener al sistema funcionando, pero no el arreglo óptimo. Los cambios de largo alcance pueden llevarse a cabo después, para corregir los problemas más generales con el diseño o el código.

El mantenimiento correctivo consta de:

- Sustitución de piezas o componentes dañados (discos duros, tarjetas de red, CD-ROM, etc)
- Reconfiguración de componentes.
- Reinstalación de software y componentes de hardware.
- Vacunación y recuperación de archivos infectados por virus informáticos.
- Instalación de nuevas versiones de software en caso de ser necesario.
- Instalación de nuevos paquetes de software a pedido del cliente.

A pesar de las pruebas y verificaciones que aparecen en etapas anteriores al mantenimiento, los programas pueden tener defectos. El mantenimiento correctivo tiene por objetivo *localizar y eliminar los posibles defectos* de los programas.

Un defecto en un sistema es una característica del sistema con el potencial de causar un fallo el cual ocurre cuando el comportamiento de un sistema es diferente del establecido en la especificación. Entre otros, los fallos en el software pueden ser de:

- Procesamiento, por ejemplo, salidas incorrectas de un programa.
- Rendimiento, por ejemplo, tiempo de respuesta demasiado alto en una búsqueda de información.
- Programación, por ejemplo, inconsistencias en el diseño de un programa.
- Documentación, por ejemplo, inconsistencias entre la funcionalidad de un programa y el manual de usuario.

En la figura 5.2.2 se muestra una distribución de causas de fallo, según un estudio estadístico realizado.

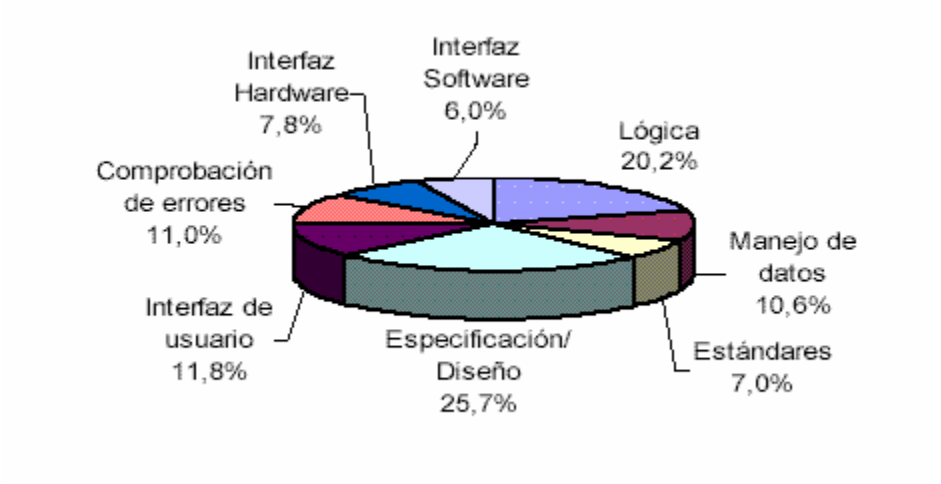


figura 5.2.2. Distribución de causas de los defectos de software

Mantenimiento Adaptativo

Este tipo de mantenimiento consiste en la modificación de un programa debido a *cambios en el entorno* (hardware o software) en el cual se ejecuta.

Los cambios en el entorno hardware pueden afectar a :

- el sistema operativo (cambio a uno más reciente)
- la arquitectura física del sistema (paso de una arquitectura de red de área local a Internet/Intranet)
- al entorno de desarrollo del software (incorporación de nuevos elementos o herramientas como ODBC)

La importancia del cambio necesario puede ser muy diferente: desde un pequeño retoque en la estructura de un módulo hasta tener que reescribir prácticamente todo el programa para su ejecución en un ambiente distribuido en una red.

Los cambios en el entorno software pueden ser de dos clases:

- En el entorno de los ***datos***, por ejemplo, al trabajar con una base de datos diferente.
- En el entorno de los ***procesos***, por ejemplo, migrando a una nueva plataforma de desarrollo con componentes distribuidos como Java, ActiveX, etc.

Este tipo de mantenimiento es cada vez más frecuente debido principalmente al cambio, cada vez más rápido, en los diversos aspectos de la informática: nuevas generaciones de hardware, nuevos sistemas operativos -o versiones de los antiguos-, y mejoras en los periféricos o en otros elementos del sistema.

5.3 Nuevas tecnologías

La web ha sufrido grandes cambios en muy poco tiempo, ya que en apenas una década ha pasado de ser algo desconocido a convertirse en un fenómeno social. Palabras como internet o web forman ya parte del vocabulario habitual debido a que la mejora en infraestructura ha permitido acceder a las “autopistas de la información” a todo público. Por esta razón son muchas las empresas que ofrecen a través de sus sitios web todo tipo de servicios, desde el correo electrónico hasta la compra electrónica. Por desgracia, los sitios web siguen siendo hoy en día islas digitales con poca comunicación o colaboración entre sí. La nueva tecnología de servicios web ofrece las herramientas necesarias para que esta situación cambie. Desde un punto de vista funcional, se trata prácticamente de los mismos servicios que conocemos hoy en día, aunque con la posibilidad de conectarse entre sí y colaborar utilizando protocolos y lenguajes estándares en internet.

Si bien el software libre ha existido desde los comienzos de la informática, sólo la popularización de internet y la accesibilidad global a la red han permitido que el software libre haya podido llegar a donde está hoy. Sólo la existencia de un gran número de colaboradores en el modelo garantiza su supervivencia.

Según los expertos, lo más probable es que en un futuro las aplicaciones basadas en software libre convivan con las de código propietario, aunque éstas últimas seguirán siendo mayoritarias en grandes empresas. La popularización de los programas basados en software libre tendrá como consecuencia una bajada de precios en el software propietario y un aumento de la competitividad en la prestación de servicios.

Estudios que se han realizado demuestran que los centros de cómputo donde se trabaja con software libre se ahorran hasta un 70% del costo que supondría implantar los mismos equipos de cómputo con programas de Microsoft al no tener

que pagar licencias. Así, aumenta considerablemente el número de equipos en éste.

El crecimiento que se está dando de la base instalada de infraestructuras sobre software libre va a tener como consecuencia un aumento de las necesidades de personal calificado en ésta área. El perfil más demandado será el de administrador de sistemas, y las principales aplicaciones que debería dominar un candidato serían las relacionadas con: administración básica de linux, administración de red, servidores de web (Apache), servidores de archivos en red (NFS, SAMBA) y servidores de correo (Postfix, Sendmail, etc.).

En la web se pueden encontrar, o construir, dos tipos de páginas:

Las que se presentan sin movimiento y sin funcionalidades más allá de los enlaces, llamadas también páginas estáticas.

Las páginas que tienen efectos especiales y en las que se puede interactuar, llamadas también páginas dinámicas, para crearlas es necesario utilizar otros lenguajes de programación, aparte de HTML.

Las páginas dinámicas de cliente se escriben en dos lenguajes de programación principalmente: javascript y visual basic script (VBScript), que se verán posteriormente un poco más a detalle. También se mencionará el concepto de DHTML y de las CSS.

Con Flash también se pueden hacer páginas dinámicas del lado del cliente. Flash es una tecnología, y un programa, para crear efectos especiales en páginas web.

Las *páginas dinámicas del cliente* son muy dependientes del sistema donde se están ejecutando y esa es su principal desventaja, ya que cada navegador tiene sus propias características, incluso cada versión, y lo que puede funcionar en un navegador puede no funcionar en otro.

Como ventaja se puede decir que estas páginas descargan al servidor algunos trabajos, ofrecen respuestas inmediatas a las acciones del usuario y permiten la utilización de algunos recursos de la máquina local.

Las *páginas dinámicas del servidor*, son reconocidas, interpretadas y ejecutadas por el propio servidor. Con ellas se puede hacer todo tipo de aplicaciones web. Son muy útiles cuando se tiene que acceder a información centralizada, situada en una base de datos en el servidor, y cuando por razones de seguridad los movimientos requeridos no se pueden hacer en la máquina del usuario. Un ejemplo de esto es un banco: no tiene ningún sentido que el cliente tenga acceso a toda la base de datos, solo a la información que le concierne.

El código para generar las páginas dinámicas del servidor se suelen escribir en el mismo archivo HTML, al igual que ocurre en las páginas del cliente. Cuando una

página es solicitada por parte de un cliente, el servidor ejecuta los *scripts* y se genera una página resultado, que solamente contiene código HTML. Este resultado final es el que se envía al cliente y puede ser interpretado sin lugar a errores ni incompatibilidades, puesto que sólo contiene HTML, posteriormente es el servidor el que maneja toda la información de las bases de datos y cualquier otro recurso, como imágenes o servidores de correo y luego envía al cliente una página web con los resultados de todas las operaciones.

Para escribir páginas dinámicas de servidor se utilizan los CGIs creados en varios lenguajes; los más comunes son en Perl, ASP, PHP y JSP.

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

Es necesario un servidor más potente y con más capacidades que el requerido para las páginas de cliente. Además, estos servidores soportan menos usuarios concurrentes, porque se requiere más tiempo de procesamiento para cada uno.

Javascript

Se trata de un lenguaje de programación del lado del cliente, gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Javascript es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

Visual Basic Script

Es un lenguaje de programación de scripts del lado del cliente, pero sólo compatible con internet explorer. Está basado en visual basic, un popular lenguaje para crear aplicaciones Windows. Tanto su sintaxis como la manera de trabajar están muy inspirados en él. Sin embargo, no todo lo que se puede hacer en visual basic se puede hacer en visual basic script, pues este último es una versión reducida del primero.

El modo de funcionamiento de visual basic script para construir efectos especiales en páginas web es muy similar al utilizado en javascript y los recursos a los que se puede acceder también son los mismos: el navegador.

DHTML

DHTML no es precisamente un lenguaje de programación. Se trata de una nueva capacidad de la que disponen los navegadores modernos, por la cual se puede tener un mayor control sobre la página.

Cualquier página que responde a las actividades del usuario y realiza efectos y funcionalidades se puede englobar dentro del DHTML, pero en este caso se refiere más a efectos en el navegador por los cuales se pueden mostrar y ocultar elementos de la página, se pueden modificar su posición, dimensiones, color, entre otros.

Las fronteras del DHTML quedan poco definidas. Las que se marcan anteriormente son sólo las que engloban a los procesos en el cliente, pero también se puede decir que DHTML es cualquier cosa que hace una página dinámica, ya sea en el cliente, el servidor o las dos cosas; es decir, dentro del concepto de DHTML se engloban también las *CSS.*

CSS

CSS, es una tecnología que permite crear páginas web de una manera más exacta.

Una de las características más potentes de la programación con hojas de estilo consiste en definir los estilos de todo un sitio web. Esto se consigue creando un archivo donde se colocan las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas ellas comparten una misma declaración de estilos y, por tanto, si se cambia una, cambiarán todas las páginas.

Sólo los navegadores de Netscape versiones de la 4 en adelante y de Microsoft a partir de la versión 3 son capaces de comprender los estilos en sintaxis CSS.

Java

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. Actualmente es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de internet como en el de la informática en general.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que si se hace un programa en Java podrá funcionar en cualquier computadora del mercado.

Ésta es una de las razones por las que Java es interesante para internet, ya que muchas personas deben tener acceso con computadoras distintas.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en otro lenguaje se puede hacer también en Java.

Applets de Java

Es otra manera de incluir código a ejecutar en los *clientes* que visualizan una página web. Se trata de pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.
Los applets de Java están programados en Java y precompilados, es por ello que la manera de trabajar de éstos varía un poco con respecto a los lenguajes de script como Javascript. Los applets son más difíciles de programar que los scripts en Javascript y requieren de conocimientos básicos o medios del lenguaje Java.

En general, cualquier aplicación que se desee hacer con acceso a través de la web se puede hacer utilizando applets de Java.

Como desventajas en relación con Javascript cabe señalar que los applets son más lentos de procesar y que tienen espacio muy delimitado en la página donde se ejecutan. Es por ello que con los applets de Java no se pueden abrir ventanas secundarias, controlar frames, formularios, capas, etc.

JSP

JSP es un acrónimo de Java Server Pages (Páginas de Servidor Java). Es una tecnología orientada a crear páginas web con programación en Java.

Con JSP se pueden crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Las JSP se pueden escribir en el editor HTML/XML habitual. En JSP se crean páginas de manera parecida a como se crean en *ASP* o *PHP*. Se generan archivos con extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutarse en el servidor.

Perl

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el web. Perl es un acrónimo de Practical Extracting and Reporting Languaje, que indica que se trata de un lenguaje de programación muy práctico para extraer información de archivos de texto y generar informes a partir del contendido de los mismos.

Perl es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de internet como *javascript* o *ASP*. Esto quiere decir que el código de los scripts en Perl no se compila sino que cada vez que se quiere ejecutar se lee el código y se pone en marcha interpretando lo que hay escrito. Además es

extensible a partir de otros lenguajes, ya que desde Perl se pueden hacer llamadas a subprogramas escritos en otros lenguajes. También desde otros lenguajes se puede ejecutar código *Perl*.

ASP

ASP (Active Server Pages) es la tecnología desarrollada por Microsoft para la creación de páginas dinámicas del servidor. ASP se escribe en la misma página web, utilizando el lenguaje Visual Basic Script o Jscript (Javascript de Microsoft).

Con las ASP se pueden realizar muchos tipos de aplicaciones distintas. Permite acceso a bases de datos, al sistema de archivos del servidor y en general a todos los recursos que tenga el propio servidor.

Actualmente se ha presentado ya la segunda versión de ASP, el ASP.NET, que comprende algunas mejoras en cuanto a posibilidades del lenguaje y rapidez con la que funciona. ASP.NET tiene algunas diferencias en cuanto a sintaxis con el ASP, de modo que se debe de tratar de distinta manera uno de otro.

Capítulo 6

Capacitación y soporte

Capacitación y soporte

Aunque el sistema este bien diseñado y desarrollado correctamente, su éxito depende de su implantación y ejecución, por lo que es importante capacitar al usuario con respecto a su uso y mantenimiento.

El término *capacitación de usuarios del sistema* se refiere a enseñar a los usuarios qué pueden hacer con el sistema y cómo lo pueden hacer, esto con el fin de que lo conozcan y se sientan cómodos con él. Si esta etapa no es exitosa, los usuarios no utilizarán correctamente el sistema y no estarán conformes con su funcionamiento.

La responsabilidad de la capacitación a los usuarios y administradores es del desarrollador, debido a que a medida que se diseña el sistema se planifican y desarrollan las ayudas que contribuirán a que el usuario aprenda a utilizarlo, pero una vez implementado la parte de soporte corresponderá al Administrador.

Los sistemas son utilizados por dos tipos de personas: los usuarios y los administradores, aunque en algunas ocasiones la misma persona es, a la vez, usuario y administrador. Sin embargo las tareas de usuarios y administradores tienen metas muy diferentes, de modo que la capacitación para cada trabajo enfatiza los diferentes aspectos del sistema.

Un usuario pone en práctica las funciones principales del sistema, por lo tanto para el usuario, el administrador es un solucionador de problemas. Sin embargo un sistema tiene a menudo tareas complementarias que le dan soporte a sus funciones superiores. Entre las funciones de este tipo se encuentran: definir quién tiene acceso al sistema y crear copias de respaldo de los archivos esenciales de datos. Normalmente estas funciones auxiliares no son realizadas directamente por los usuarios. De hecho son los administradores quienes llevan a cabo estas operaciones de soporte del trabajo principal.

6.1 Capacitación a los usuarios del sistema

Como se mencionó anteriormente el Sistema de Acopio de Información en Línea será usado por dos tipos de personas:

- Las indicadas por cada división para acceder al sistema, a quienes se nombrará usuarios, y
- Las que quedarán como encargadas del sistema, a quienes se llamará administradores.

La capacitación para los usuarios estará basada primariamente sobre las funciones del sistema a las cuales necesitarán acceder, es decir, los usuarios podrán navegar entre los registros que su contraseña les permita para agregar otro, borrar uno existente o acceder a alguno en particular para modificarlo.

Los usuarios no necesitarán tener conocimiento de las operaciones internas del sistema, sólo se presentarán para él las funciones primarias de modo que las comprenda claramente y aprenda a ejecutarlas.

Por otro lado, el propósito de la capacitación para el administrador será la familiaridad con las operaciones para el soporte del sistema, la creación de usuarios, saber como darles soporte y el manejo de la base de datos; este entrenamiento apunta a la forma en que el sistema trabaja, en lugar de apuntar a lo que hace

Si no se lleva a cabo la capacitación a los usuarios y los administradores, estos tienden a poner en práctica solamente aquellas funciones con las que se sienten cómodos y pueden no utilizar otras funciones que pueden hacer el sistema más eficiente y productivo.

Ayudas a la capacitación

La capacitación puede hacerse de diversas maneras. Sin importar cómo se da, se proporciona en todo momento información a los usuarios y administradores. De este modo, sí en algún momento el usuario olvida como acceder a la información o como llenar correctamente los campos requeridos, la capacitación incluye métodos para aprender a encontrar la información correspondiente.

Algunas de las ayudas a la capacitación son:

- *Documentos*. Todo sistema va acompañado de documentación formal que es el soporte de la capacitación. Los documentos contienen toda la información que se necesita para utilizar el sistema correcta y eficientemente. Al existir en manuales separados o disponible en línea, los documentos están accesibles para los usuarios y los administradores cuando el sistema está funcionando.

  En muchos casos los usuarios prefieren iconos bien definidos, ayuda en línea, demostraciones y clases para aprender cómo trabaja el sistema.

- *Iconos y ayuda en línea*. Un sistema puede estar diseñado de manera que su interfaz con el usuario haga que sus funciones resulten claras y fáciles de comprender. La mayoría de los sistemas siguen el ejemplo de Apple y Xerox al utilizar iconos para representar la opción de un usuario de funciones del sistema. Un clic del mouse selecciona un icono y un segundo clic invoca la función. La ayuda en línea se trata más adelante en este capítulo, debido a que es la que se ocupa en el sistema (SAILFI).

- *Demostraciones y cursos*. Las demostraciones y los cursos agregan personalización al entrenamiento, además hacen que los usuarios y administradores respondan positivamente. Las necesidades de los usuarios están puestas en primer término, y la demostración o el curso se enfoca sobre un aspecto particular del sistema. La demostración puede ser una clase formal de presentación del sistema.

La capacitación es exitosa solamente cuando satisface las necesidades de los usuarios y se adapta a sus capacidades. Las preferencias personales, los estilos de trabajo y las presiones organizacionales juegan un papel muy importante en este éxito. Hay que considerar que algunas personas prefieren aprender leyendo, otras escuchando y otras utilizando una combinación de técnicas.

Finalmente, la ubicación física de las personas que utilizarán el sistema determinará el tipo de capacitación.

6.2 Soporte técnico

El soporte técnico ha sido creado para ayudar al usuario a eliminar los problemas específicos que pueden ocurrir ya sea al ejecutar, instalar o configurar un programa, así como en la instalación o cambio de algún componente o del equipo, eliminación de virus, etc. Todo esto con la finalidad de resolver el problema y satisfacer las necesidades del usuario brindándole un servicio de calidad.

En el servicio de soporte técnico se guía, paso a paso, a través del proceso de solución de un problema, ya sea de hardware o software, así como también se ayuda a interpretar cualquier terminología compleja que le pueda ser poco comprensible.

En este caso, el soporte de hardware será dado por el personal de cada división encargado de esta tarea y se enfocará en atender las solicitudes de usuarios que tengan problemas de funcionamiento con sus equipos o que tengan problemas técnicos para acceder al sistema.

El SAILFI tiene como soporte técnico la ayuda en línea, que formar parte del sistema. Por otro lado, las personas que requieran una cuenta nueva necesitarán comunicarse con el administrador del sistema, para este fin se cuenta con una liga que hará referencia a su dirección de correo electrónico.

En la figura 6.2.1 se muestra un diagrama ajustado al sistema a partir de una metodología general de soporte[8] que se utiliza para resolver problemas de los usuarios.

[8] Diseño de Sistemas de Información (Teoría y Práctica), John G. Burch et. al, Gpo. Noriega Editores, 5ta edición.

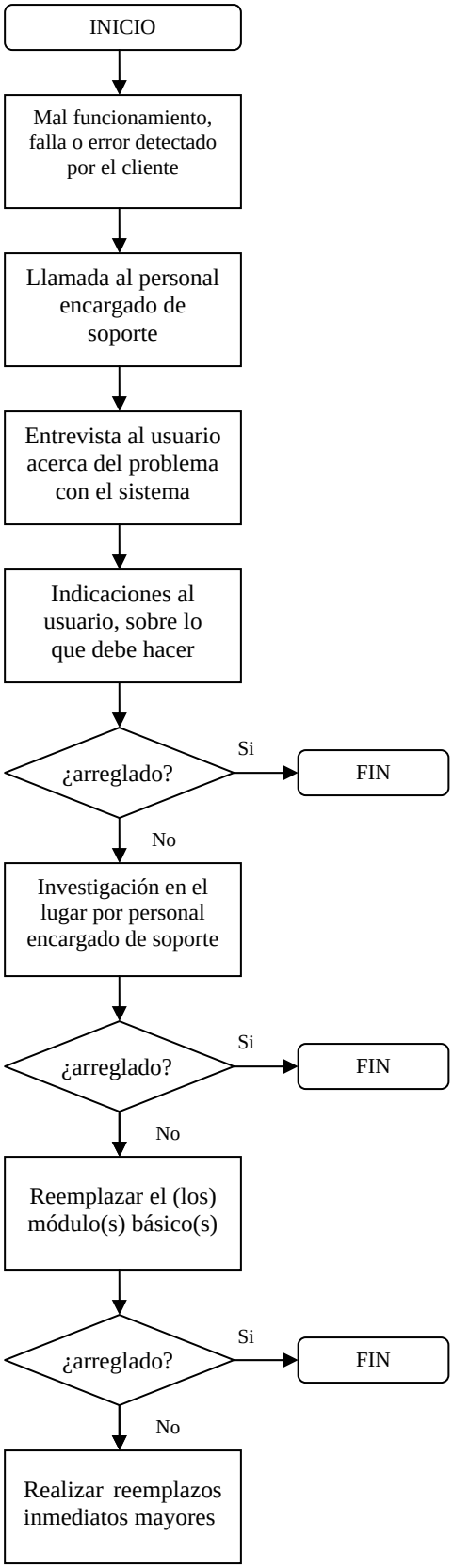


figura 6.2.1 Metodología para soporte

6.3 Ayuda en línea

Anteriormente se dijo que el sistema esta diseñado de manera que su interfaz con el usuario hace que las funciones resulten claras y fáciles de comprender. De esta manera la ayuda en línea hace que la capacitación resulte más fácil. El usuario puede explorar información adicional acerca del propósito o utilización de un campo requerido, sin ocupar el tiempo en la búsqueda a través del documento en papel. La asistencia en línea más sofisticada, que aprovecha la tecnología multimedia, le permite al usuario obtener todo el detalle que necesite de una forma agradable y fácil de usar.

El SAILFI tiene ayuda en línea, la cual consiste en varias páginas[9] WEB con la información que es requerida para llenar los formularios (se especifica el tipo de dato esperado por el sistema para almacenarse en la base de datos), dicha ayuda se despliega al dar clic sobre el icono de ayuda, en una ventana del navegador independiente a la que se esta utilizando.

[9] Tomando como referencia la ayuda del SAP de la UNAM.

Conclusiones

El tratar de aprovechar al máximo los recursos humanos y materiales, con que cuenta la universidad debe ser siempre un objetivo primordial de toda actividad de índole académico y administrativo, en busca de un crecimiento adecuado no sólo de esta entidad, sino del país mismo.

En este sentido la realización de proyectos de esta naturaleza buscan cumplir con ese objetivo, en el entendido claro, de buscar la calidad en él que la Facultad no sólo espera, sino que además requiere y merece.

El desarrollo de éste sistema surgió como una necesidad de actualizar los procedimientos que se tenían para el manejo de la información estadística que se produce en la Facultad, desde su recopilación, pasando por su almacenamiento y hasta llegar a una disponibilidad de ella de forma tan eficiente como sea posible, lo cual fue parte de los objetivos que se propusieron.

Así, se cumplió con el hecho de buscar una interfaz sencilla, de fácil comprensión y manipulación por parte del usuario, pero no por ello ineficiente.

Con el sistema se puede almacenar de una manera estandarizada la información, lo que garantiza que su recuperación también se realice en esta forma.

Se cuenta ahora con un medio de consulta, el cual permite hacerlo en cualquier momento, desde cualquier máquina con acceso a internet y de una manera confiable, de las diversas actividades académicas que se realizan en la Facultad lo que facilitará la presentación de dicha información cuando ésta sea requerida por las distintas instancias.

Se podrá también con la utilización del sistema disminuir la carga de trabajo que se genera al realizar la recopilación de la información hasta que va a requerirse. Se pretende que el acopio se pueda llevar a cabo casi durante todo el año, lo que ayudará a evitar también posibles pérdidas en la información (lo que puede suceder cuando ésta se tiene almacenada en medios impresos) si el proceso se realiza a su debido tiempo.

La utilización de una metodología en el desarrollo de un sistema es muy importante porque permite avanzar de una manera ordenada y eficiente a través de cada una de las etapas de desarrollo. Dentro de éste, las pruebas realizadas son especialmente importantes.

La elección de software libre para su desarrollo se basó en una comparación de ventajas, y desventajas por supuesto, que éste ofrece contra el ya tradicional y todavía mayormente utilizado software propietario, pero que actualmente se encuentra en un punto de difusión y crecimiento de importante consideración.

Es necesario tomar en cuenta que el proceso puede no terminar aquí. Esto significa que cambios en la estructura organizativa y administrativa de la propia Facultad e incluso de la Universidad, pudieran requerir una evolución del sistema, lo que comúnmente se denomina versión, en el lenguaje computacional.

Por ello, se reconoce que el sistema es perfectible aún y por lo mismo las actividades de mantenimiento, tanto de software como de hardware, son necesarias para un óptimo desempeño de éste.

Apéndices

Apéndice I (Metodología empleada)

Definición de un Modelo de Ciclo de Vida

La ingeniería de software está compuesta por una serie de pasos que comprenden los métodos, las herramientas y los procedimientos para crear software de calidad. Habitualmente estos pasos reciben el nombre de *paradigmas de la ingeniería de software o modelos de ciclo de vida de software*.

Existen actualmente distintos paradigmas que pueden adoptarse para guiar el proceso de desarrollo de un producto software, no obstante, todos ellos se basan en tres paradigmas básicos:

- Paradigma de Ciclo de Vida Clásico
- Paradigma de Prototipos Evolutivos
- Paradigma del Modelo en Espiral

La selección del paradigma a utilizar en un proyecto de desarrollo de software está directamente relacionada con la naturaleza del proyecto y de la aplicación, los métodos y herramientas a utilizar y los controles e informes requeridos.

Estos modelos están compuestos por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto de software, hasta aquel en que el producto deja definitivamente de ser utilizado por el último de sus usuarios.

La elección de un cierto modelo para un determinado tipo de proyecto es de vital importancia, ya que el orden de las etapas es un factor importante.

Entre las principales funciones de un modelo de ciclo de vida del software se encuentran las siguientes:

- Describir las fases principales de desarrollo de software
- Definir los resultados esperados al término de las fases primarias
- Ayudar a administrar el progreso del desarrollo
- Proveer un espacio de trabajo para la definición de un detallado proceso de desarrollo de software

Así, los modelos por una parte suministran una guía para los ingenieros dedicados al desarrollo de software con el fin de ordenar las diversas actividades técnicas en el proyecto y por otra un marco para la administración del desarrollo y el mantenimiento, en el sentido en que permiten estimar recursos.

Modelos de Ciclo de Vida y de Desarrollo

Como se ha mencionado, no existe un único paradigma o modelo de ciclo de vida que pueda definir todas las fases por las que deba pasar un producto de software y que se adapte a todas las necesidades y problemas. Esto es así porque existe un amplio espectro de aplicaciones para las que deben desarrollarse productos de diferente naturaleza, por ejemplo: software de simulación, software de administración, entre otros.

El carácter de estas aplicaciones hace que cada una de ellas requiera soluciones particulares y específicas, aún más, puede suceder que dentro de un mismo tipo existan necesidades diferentes. Por otra parte, los centros de desarrollo que deben implementarlas poseen, a su vez, estructuras organizativas y modalidades de trabajo particulares.

También forman parte de esta problemática otros factores, entre los que pueden mencionarse:

- El tipo de cliente o usuarios para el que se desarrolla el sistema
- La volatilidad de requisitos
- La antipatía al riesgo por parte del cliente y de la organización de desarrollo
- El área donde se utiliza la aplicación

Estos factores suelen combinarse, por lo que puede considerarse lógico pensar que exista la necesidad de aplicar diferentes métodos para diferentes tipos de producto. Por tal motivo, resulta indispensable realizar, previo al inicio de un proyecto de desarrollo, la identificación y análisis de los diferentes modelos de ciclo de vida, adoptando aquél que más se ajuste a las necesidades del proyecto, del cliente y de la organización de desarrollo.

A continuación se describen brevemente los modelos de ciclo de vida de mayor aplicación:

Modelo de Ciclo de Vida Clásico (o en Cascada)

Es el paradigma más antiguo y ampliamente difundido, fue presentado por primera vez por Royce en 1970 y aplicado con éxito para estructurar y gestionar grandes proyectos de software en importantes compañías de desarrollo.

Éste es el más básico de todos los modelos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. La visión del modelo en cascada del desarrollo de software es muy simple; dice que el desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuyen a la satisfacción de metas de esa fase o a una sub-secuencia de metas de la misma.

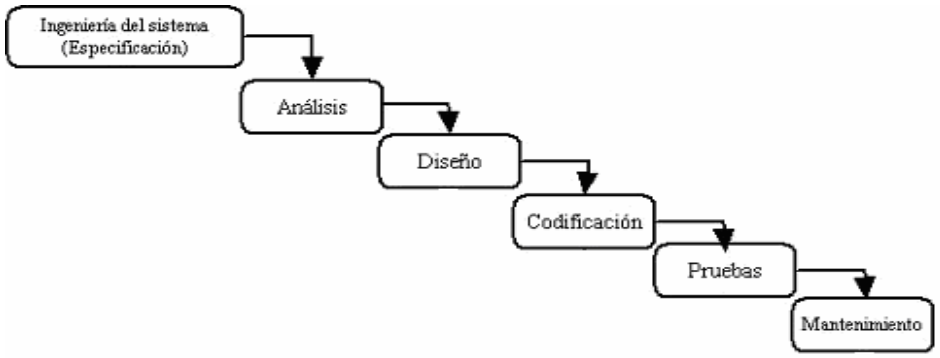


figura AI.1 Modelo de Ciclo de Vida en Cascada

Modelo de Prototipado Evolutivo

En este modelo se construye una serie de grandes versiones sucesivas de un producto. Sin embargo, mientras que la aproximación incremental presupone que el conjunto completo de requerimientos es conocido al comenzar, el modelo evolutivo asume que los requerimientos no son completamente conocidos al inicio del proyecto.

En el modelo evolutivo, los requerimientos son cuidadosamente examinados, y sólo esos que son bien comprendidos son seleccionados para el primer incremento. Los desarrolladores construyen una implementación parcial del sistema que recibe sólo estos requerimientos.

El sistema es entonces desarrollado, los usuarios lo utilizan y proveen retroalimentación a los desarrolladores. Basada en esta retroalimentación, la especificación de requerimientos es actualizada, y una segunda versión del producto es desarrollada y desplegada. El proceso se repite indefinidamente.

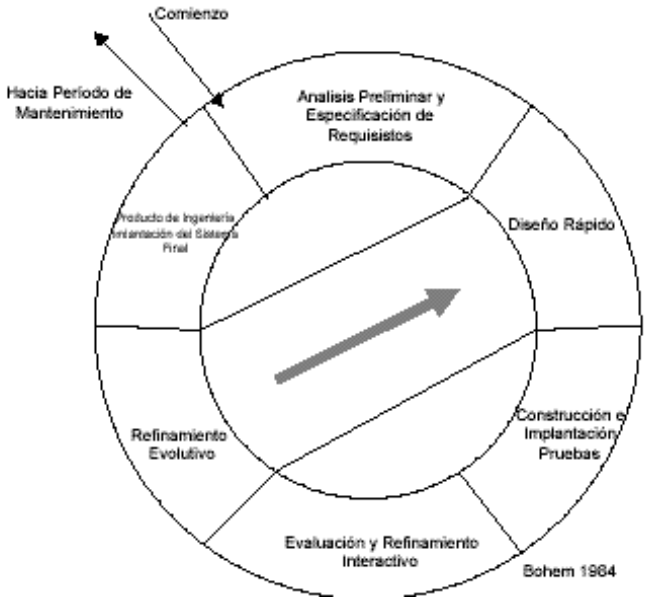


Figura AI.2 Modelo de Prototipado Evolutivo

Modelo de Ciclo de Vida en Espiral

Este modelo, de relativamente reciente surgimiento (Böem 1986), incorpora métodos de proceso que están influenciados por el control y gestión del riesgo para el análisis y estructuración del proceso de desarrollo.

Esta nueva filosofía se encuentra representada por ciclos de desarrollo evolutivo e iterativo en forma de espiral, cuyo avance angular representa el progreso del desarrollo, en tanto que el desplazamiento radial desde el centro hacia fuera indica el incremento de los costos de desarrollo en forma acumulativa. La siguiente figura permite visualizar gráficamente el comportamiento de este tipo de ciclo de vida, en ella se pueden distinguir en sus cuatro cuadrantes las cuatro actividades principales del modelo:

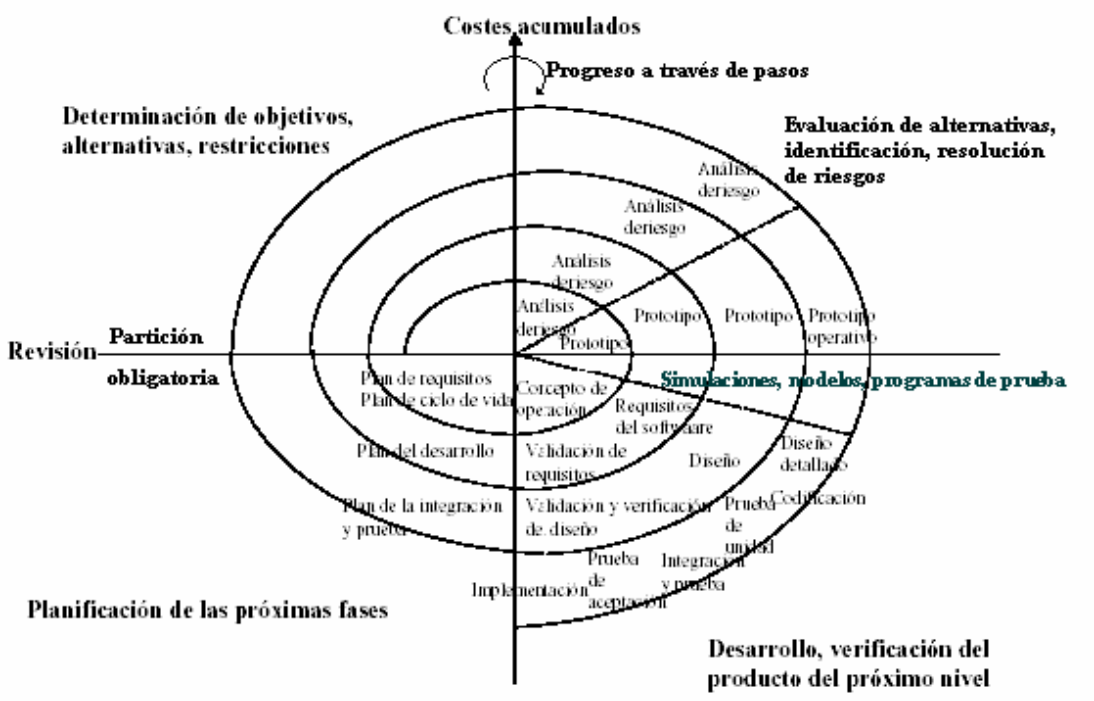


Figura AI.3 Modelo de Ciclo de Vida en Espiral

Los elementos a desarrollar en las distintas etapas de los modelos descritos anteriormente son bastante semejantes, pero cada uno de los modelos se obtiene empleando distintas técnicas.

Se eligió el modelo de ciclo de vida clásico para desarrollar el SAILFI, porque al ser el más sencillo permite explicarlo de una manera fácil al cliente, debido a que consiste en descomponer la actividad global del proyecto en fases que se suceden de manera lineal; es decir, cada una se realiza una sola vez, después de la anterior y antes que la siguiente. No obstante se admite cierta realimentación correctiva para mejoras del sistema.

Apéndice II (Glosario de términos)

ActiveX

Los controles ActiveX son la versión actualizada de la especificación sobre controles OLE. Los controles son una arquitectura básica para crear componentes de software programables que pueden utilizarse en diferentes contenedores, incluidos los exploradores Web compatibles en Internet. Cualquier control ActiveX puede ser un control de Internet y aportar su funcionalidad a un documento activo o formar parte de una página web; es decir, controles OLE + internet = controles ActiveX.

Apache

Es un servidor web libre, es decir, el encargado de construir y devolver las páginas web que solicitan los navegadores. Su nombre procede de "a patchy server", por ser una versión "parcheada" de uno de los primeros servidores web, el NCSA HTTPD, y actualmente corre en muy diversas plataformas (Unix, Windows, etc.).

ASP (Active Server Pages) Páginas Activas del Servidor.

Awk Es un lenguaje de búsqueda y procesamiento de patrones, además de ser una herramienta con las que se cuenta en Unix. El nombre *awk* proviene de las iniciales de sus desarrolladores: Alfred V. Aho, Brian W. Kerningan y Peter J. Weinberger, de los Laboratorios Bell de AT&T.

C El C se encuentra en la jerarquía de lenguajes en un nivel intermedio entre Pascal y el Ensamblador. Pretende ser un lenguaje de alto nivel con la versatilidad del bajo nivel. Se diseñó junto con el sistema operativo UNIX y está muy orientado a trabajar en su entorno.

Chilisoft

ChiliSoft ASP es un software que permite ejecutar aplicaciones ASP en distintas plataformas (UNIX, Linux, etc), utilizando diferentes servidores web, como Apache, Planet, Zeus, etc.

Cliente

Cualquier elemento de un sistema de información que requiere un servicio mediante el envío de solicitudes al servidor a través de una red, por ejemplo: internet

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

CSS (Cascading Style Sheets) Hojas de estilo en Cascada.

DBMS
(Database Management System, Sistema Manejador de Base de Datos) es sustancialmente un software que se coloca entre el usuario y los datos para facilitar funciones como: almacenar físicamente los datos, garantizar consistencia e integridad además de manejar vistas a la información.

HTML
son las siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto). El hipertexto es una palabra o frase en especial que “apunta” hacia otra página.

HTTP es el acrónimo de HyperText Transfer Protocol

IIS (Internet Information Server). El IIS engloba un conjunto de herramientas destinadas al control de servicios de Internet como el Web, FTP, correo y servidores de noticias. Además incluye el soporte necesario para la creación de páginas dinámicas en el servidor mediante el lenguaje ASP.

JSP (Java Server Page) Páginas de Servidor Java. Se refiere a un tipo especial de páginas HTML, en las cuales se insertan scripts, se procesan en línea para finalmente desplegar un resultado final al usuario en forma de HTML. Por lo general dichos programas hacen consultas a bases de datos y dependiendo del resultado que se despliegue será la información que se muestre a cada usuario de manera individual. Los archivos de este tipo llevan la extensión "jsp".

Lenguaje de Bajo Nivel
Lenguaje de programación que la computadora puede entender a la hora de ejecutar programas, lo que aumenta su velocidad de ejecución, pues no necesita un intérprete que traduzca cada línea de instrucciones.

Lenguaje de Alto Nivel
Lenguaje de programación en el que las instrucciones enviadas para que la computadora ejecute ciertas órdenes son similares al lenguaje humano. Dado que la computadora no es capaz de reconocer estas órdenes, es necesario el uso de un intérprete que traduzca el lenguaje de alto nivel a un lenguaje de bajo nivel que el sistema pueda entender.

Navegar
Es un término que se utiliza para referirse a la acción de acceder a los registros. También se refiere a ir de una página a otra en la web.

ODBC
son las siglas de Open DataBase Connectivity, un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de

ODBC es hacer posible el acceder a los datos de cualquier aplicación, sin importar qué DBMS almacene los datos, ODBC logra esto al insertar una capa intermedia entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda. Para que esto funcione tanto la aplicación como el DBMS deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el DBMS debe ser capaz de responder a ellos.

OLE Un objeto OLE (Object Linking and Embedding) significa el estándar de vinculación e incrustación de objetos. OLE es un entorno unificado de servicios basados en objetos con la capacidad de personalizarlos y ampliar arbitrariamente la arquitectura, con la finalidad global de permitir una integración rica entre los componentes.

Páginas dinámicas del cliente

Son llamadas así porque es el navegador del cliente el que soporta la carga de procesamiento. El navegador es el encargado de interpretar las instrucciones y ejecutarlas para realizar efectos e interactividades con el usuario, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Páginas dinámicas del servidor

Son llamadas así ya que se ejecutan en el servidor web, justo antes de que se envíe la página a través de internet al cliente

Personal Web Server

Se trata de la versión light del Internet Information Server utilizada en entorno Windows 95 y 98. Ambos son aplicaciones que permiten hacer de las PCs auténticos servidores web.

PHP (Acrónimo de "PHP: Hypertext Preprocessor"), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

POSIX

*(*Portable Operating System Interface for Computer Environments) conjunto de estándares que definen una serie de aspectos de un sistema operativo, que se diseñó para garantizar portabilidad entre distintos sistemas UNIX.

RISC (Reduced Instruction Set Computing) Procesadores con conjuntos de instrucciones de complejidad reducida.

Scripts

Un conjunto de comandos escritos en un lenguaje interpretado para automatizar ciertas tareas de aplicación.

Sed Es una de las herramientas más peculiares de Unís, significa Stream Editor (editor de flujo). Los editores suelen aceptar de forma interactiva las

modificaciones que se desean insertar. Permite crear pequeños programas "shell scripts" y da la capacidad de modificar de forma automática el contenido de un archivo.

Servidor

Dispositivo de un sistema que resuelve las peticiones de otros elementos del sistema, denominados clientes.

TCP/IP

(Transmission Control Protocol / Internet Protocol) es un sistema de comunicaciones muy sólido y robusto bajo el cual se integran todas las redes que conforman lo que se conoce actualmente como Internet.

Usuario

Son las personas que van a acceder al servicio, usualmente las restricciones de acceso al sistema se aplican de acuerdo a usuario y contraseña.

Bibliografía

1. KORTH, Henry ; et. al.
   Fundamentos de bases de datos
   4a. Edición
   Madrid
   McGraw Hill, 2002

2. PFLEEGER, Shari
   Ingeniería de software. Teoría y práctica
   Buenos Aires
   Pearson Education, 2002

3. TENENBAUM
   Sistemas operativos modernos
   México
   Prentice Hall Hispanoamericana, 1993

4. LeBLANC
   Construya un site perfecto de internet con linux
   México
   Prentice Hall Hispanoamericana, 1996

5. CORTES ROSAS, Jesús ; GONZÁLEZ CÁRDENAS, Miguel ; et.al.
   Algoritmos numéricos
   México
   Facultad de Ingeniería, UNAM, 2002

6. BURCH G., John ; et. al.
   Diseño de Sistemas de Información (Teoría y Práctica)
   5a. Edición.
   Gpo. Noriega Editores

7. ACHOUR, Mehdi ; BETZ, Friedhelm ; et.al
   Manual de php
   PHP Documentation Group, 2004

8. *Manual de Postgresql*
   Postgresql Global Development Group, 2003

Referencias de sitios de internet

9. Sitio oficial de PHP
   http://www.php.net

10. http://www.gnu.org/philosophy/philosophy.es.html

11. LUSKY, Raul
    Linux, otra opción en sistemas operativos
    http://www.maestrosdelweb.org/editorial/extra.asp

12. http://www.monografias.com/trabajos12/diflu/diflu.shtml

13. http://www.calidad.org/s/flujo.pdf

14. MARQUÉS, Mercedes
    Apuntes de ficheros de bases de datos (6, 45, 69 y 70)
    http://www3.uji.es/~mmarques/f47/apun/

15. http://www.programacion.com/bbdd/articulo/bbdd_disenyo/#bbdd_disenyo_pn

16. http://atenea.udistrital.edu.co/profesores/jdimate/basedatos1/tema1_3.htm

17. http://alarcos.inf-cr.uclm.es/doc/bda/doc/teo/bda-t3.pdf

18. http://window.to/concepcion.com.do

19. http://www.monografias.com/trabajos/anaydisesis/anaydisesis.shtml

20. http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/adproysisinf.htm

21. Microsoft ibérica
    Ingeniería del software
    http://www.microsoft.com/spanish/MSDN/estudiantes/ingsoft/default.asp

22. http://es.wikipedia.org/wiki/Cliente-servidor

23. http://www.mastermas.com/reportajes/linux/P2.asp

24. RmyA
    Consultoría en mejora de procesos
    http://www.rmya.com.ar/Download/Folleto%20MP.pdf

25. ACE website
Verificación, prueba y mantenimiento de los programas
http://www.aceproject.org/main/espanol/et/ete05c.htm

26. Desarrollo web
Artículos en desarrollo web
http://www.desarrolloweb.com/articulos/712.php
http://www.desarrolloweb.com/articulos/1328.php

27. CARRERO FERNÁNDEZ, David
Diccionario informático
http://www.glosarium.com/list/14/

28. Asociación Española de Internet
Ciberaula
http://www.ciberaula.com/curso/iis/

29. Cibertecanet. Aprender en internet
http://www.ciberteca.net/articulos/programacion/net/

30. http://www.manycomics.com/se/ciclo.htmx#INICIO

31. http://www.itba.edu.ar/capis/webcapis/trabajosfinalesdeespecialidad