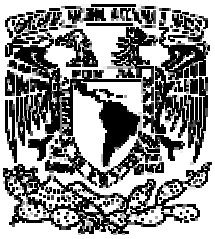


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN PARA EL BANCO DE IMÁGENES ASTRONÓMICAS DEL INSTITUTO DE ASTRONOMÍA UNAM

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

GARCÍA MARTÍNEZ ILIANA

DIRECTORES DE TESIS:

ING. ORLANDO ZALDíVAR ZAMORATEGUI

ING. GILBERTO ZAVALA PÉREZ

CIUDAD UNIVERSITARIA **2004**

Dedico esta tesis a...

A Dios, por la vida que me ha dado.
Por la libertad que no da para decidir nuestro camino... para equivocarnos, para aprender, para superarnos.
Todas las cosas pasan por algo... porque? ... no lo sé, pero pasan por algo.

A mis padres por todo lo que me han dado, por mi familia, por el apoyo que he recibido cuando he estado en problemas y por las alegrías compartidas. Por las marcas indelebles que han dejado en mi alma.
Apolinar y Guadalupe, es un honor ser su hija. Los amo.

A mis hermanas por ser mis amigas, confidentes, maestras y muchas veces un sostén en mi vida.
Mónica y Lupita... gracias por ser mis hermanas. Las quiero mucho.

A mis sobrinos
Ulises Santos y Héctor Elías, por el simple hecho de existir. Y por supuesto a Daniel por todo el apoyo que he recibido de su parte.

A mis abuelos, tíos y primos, agradecida estoy con uds. por los regaños, el apoyo, el cariño... por todo.
Santos, Ricarda, Álvaro, Micaela, Adrián y todos mis tíos y demás primos!,
Con todo cariño, besos.

A mis amigos de la Facultad que siempre me han apoyado y sacado de tantas crisis.
Mar, Leti, Marco Polo, Pedro, Erik, Herverth, Adri, Liz, Ara, Marian, Ross, Lalo, Sofi, Poncho, Miri, Iris, Vic, Fab, Marco Antonio ^2 y a todos que me faltan!!,
muchas, muchas gracias por todas las risas, las preocupaciones, las lágrimas, la euforia y las travesuras que vivimos en nuestros años dentro de la facultad, todo esta guardado junto con cada uno de sus nombres, en un lugar muy especial de mi corazón.

A mis amigos de Astronomía. Y con todo cariño escribo, mis amigos...
Isidro, Sinhue, Alex, Jorge, Nacho, Fabio, Miguel Ángel, Mario, Fausto y Jess.
No saben de que túnel tan obscuro me sacaron. Los quiero mucho chicos.

A los de maestría: Janet, Tomas, Primoz, Geovanni, Memo, Eduardo y Lirio, que me enseñaron que una maestría no esta peleada con la vida.

A todo el personal del instituto.
Gil, Pepe Peña, Daniel Flores, Dn. Flemón, Dn. Eduardo, Sra. Ma. Eugenia, Elizabeth, Gloria y Juan Manuel, por todos sus consejos y por el apoyo brindado.

A los amigos de toda la vida
Male, Miri, Lupe, Ara y Ame. El tiempo ni las distancias importan.

Y a toda la gente que me ha apoyado en mi vida y en mi trabajo, y que no he podido mencionar.

A todo lo vivido, dejado y aprendido con este trabajo.

Iliana

Agradecimientos

Un sincero agradecimiento a todos mis sinodales que me brindaron sus conocimientos, y especialmente a mi codirector y director:

Ing. Orlando Zaldívar Zamorategui por su ayuda, correcciones y paciencia en lo escrito de la tesis... Gracias.

Ing. Gilberto Zavala Pérez, por la ayuda a la realización del proyecto y por todo lo que me apoyo en el Instituto de Astronomía... Gracias.

A todos los miembros del Instituto de Astronomía
Por haber colaborado en mi formación, y por haberme dado la oportunidad de conocer una infinidad de cosas.

Y un sincero agradecimiento a Juan Carlos Yustis, por ser parte de este proyecto.

A la Facultad de Ingeniería y a sus catedráticos,
por permitirme lograr una de mis más grandes metas... Gracias.

Y sobre todo a la Universidad Nacional Autónoma de México
Por brindarnos las bases para ser mejor cada día y hacer mejor a nuestro país, MÉXICO.

Orgullosamente PUMA!

Por mi raza hablara el espíritu.

México, pumas, universidad!!.
Gooooya, gooooya, cachun cachun ra, ra, cachun cachun ra, ra, gooooya universidad !!!

Índice

PRÓLOGO

Prólogo

El Instituto de Astronomía ha tenido una repercusión muy marcada en el ámbito científico a nivel nacional e internacional, desde sus comienzos hasta nuestros días generando científicos de la más alta calidad.

Los inicios de esta sociedad figuran en el Palacio Nacional en 1867 cuando se funda el Observatorio Astronómico Nacional (OAN), sin embargo, se tiene una gran cantidad de información de lo realizado por las culturas prehispánicas, como fueron los aztecas (precisamente en donde se ubica el castillo de Chapultepec), los mayas, teotihuacanos, entre muchos otros.

Con un pequeño observatorio en la azotea del Palacio Nacional dieron inicio las primeras investigaciones sobre astronomía en la era actual, donde continuaron hasta 1878, que debido al gran crecimiento de la ciudad, la contaminación lumínica y la trepidación del edificio, el observatorio fue traslado al castillo de Chapultepec en el año de 1878 y, subsecuentemente por los mismos motivos en 1908 fue traslado al ex-arzobispado en Tacubaya.

En el año de 1929 cuando la Universidad Nacional Autónoma de México es declarada por decreto presidencial autónoma, el OAN es incorporado a la universidad.

En el pueblo de Tonantzintla que se encuentra en las cercanías de la ciudad de Puebla, en el año de 1951 se funda la estación del OAN (UNAM) y en el año de 1961 la universidad inaugura el telescopio de 1 m. de diámetro en la óptica principal del observatorio.

En 1967 se reconoce la categoría de instituto de investigación al OAN, por lo que se crea el Instituto de Astronomía de la UNAM con sede en Ciudad Universitaria (IAUNAM) y el OAN se reserva para las estaciones de observación que están a cargo del nuevo instituto.

En la Sierra de San Pedro Mártir, Baja California Norte, se establece un nuevo lugar para el telescopio, ya que en Tonantzintla, Puebla, la contaminación lumínica comienza a dar muchos problemas y en el año de 1971 se instalan en San Pedro los telescopios de 1.5 m. y de 84 cm. de diámetro, convirtiéndose la UNAM en el principal rector de éstos.

Actualmente, se tiene el telescopio de 2.1 m., que se cataloga entre los mejores de su clase en el mundo y donde se han realizado diversos estudios, dando lugar a un gran número de investigaciones que han sido publicadas y reconocidas en revistas especializadas del más alto nivel en el plano internacional.

En la institución han surgido múltiples personalidades no sólo para el instituto sino para la ciencia en México, como lo fueron los iniciadores de la astronomía moderna en México: Dr. Guillermo Haro (1913–1988) y la Dra. Paris Marie Pishmish (?-1999), respaldados por el

Ing. José de los Ángeles Anguiano Limón (1840-1921), el Ing. Francisco Díaz Covarrubias (1833-1889), Dr. Joaquín Gallo Monterrubio (1882-1965), el Ing. Valentín Gama y Cruz (1868-1942), entre otros.

En el instituto se encuentra el posgrado en ciencias (Astronomía). Los programas de maestría y doctorado se iniciaron en 1989 y es catalogado como posgrado de excelencia por parte de CONACyT.

El instituto es una base fundamental para la ciencia en México y un gran apoyo para el extranjero, ya que mantiene una gran vinculación con proyectos de investigaciones fuera y dentro del país de gran importancia, uno de éstos es el que da pie a realizar este proyecto, para solucionar problemas que se tienen en la actualidad, como el apoyo a la comunidad científica en el desarrollo de sus trabajos, tener una cierta organización de las imágenes que se obtienen en el instituto por parte de los investigadores y estudiantes, y tratar de dar un mayor auge a la divulgación en México sobre la astronomía, mostrando las imágenes de las investigaciones realizadas.

INTRODUCCIÓN

Introducción

El objetivo general del trabajo de tesis es realizar el análisis, diseño y desarrollo de la base de datos, así como de las interfaces de usuario para el Banco de Imágenes Astronómicas (BIA) del Instituto de Astronomía de la UNAM, con acceso desde la red LAN del instituto y del exterior, usando una interfaz del World Wide Web.

Lo que el Instituto de Astronomía obtiene con este proyecto es:

1) Recopilar un acervo de imágenes astronómicas.
2) Integrar y organizar el acervo en un banco de imágenes.
3) Difundirlo ante la comunidad en general.

La organización escrita de la tesis se basa en la metodología que se utiliza para la realización del proyecto, que en nuestro caso es una ***metodología en espiral*** compuesta básicamente por cinco tareas, detalladas tanto en el capítulo segundo como en el capítulo tercero:

1. Comunicación con el cliente.
2. Análisis.
3. Propuesta de solución.
4. Programación del sistema.
5. Evaluación del cliente.

En el que cada tarea representa un capítulo, más el capítulo dos que son fundamentos teóricos para el desarrollo del sistema.

Los capítulos se describen a continuación.

Capítulo primero. Análisis preliminar.

En este capítulo, se toma el planteamiento del problema, las delimitaciones y sus alcances, la situación actual respecto al software, hardware y las características de la comunidad del Instituto, además de otros requerimientos para el sistema.

Capítulo segundo. Fundamentos teóricos.

Aquí se exponen los fundamentos teóricos para el desarrollo del sistema, principalmente divididos en dos partes:

a) Ingeniería del software, que se refiere a tipos de metodologías, clasificaciones del proceso como grados de rigor y tipo de proyecto, planificación temporal, entre otros. Además de las

teorías de los diferentes modelos y flujos de información, así como sus estructuras.

b) Teorías de las bases de datos; diagramas, clasificaciones, procesos, etc.

Capítulo tercero. Propuesta de solución.

Se define la estructura del diseño y la propuesta de solución que en base al capítulo anterior se establece.

Se declara el tipo de metodología a utilizar, sus regiones de tareas y el porqué fue tomada, así como grados de rigor y tipo de proyecto, estudios de tiempo, modelado del sistema y los diccionarios de datos.

Capítulo cuarto. Diseño del proyecto.

Se detalla el diseño del proyecto tanto para el sistema, como para la base de datos; para el primero conlleva el diseño de datos, el diseño arquitectónico, el diseño de interfaz y el diseño procedimental; por lo tanto, aquí se muestran los diagramas a bloques, diagramas de estructuras de datos, diagramas entidad-relación y las estructuras básicas de cada una de las secciones y sus principales funciones, además de las relaciones entre éstas.

Para la base de datos se determinan el diseño de datos conceptual, el diseño lógico y el diseño físico.

Capítulo quinto. Programación del sistema.

Aquí se explica la programación tanto de la base de datos como de las interfaces del usuario. Se presenta cada una de las tablas y una descripción detallada de *una* de las secciones del sistema.

Capítulo sexto. Análisis de resultados y especificaciones finales.

Se analizan las últimas partes del sistema: la estructura final de hardware donde es instalado el sistema, el arreglo de software del sistema y algunas especificaciones finales.

Al final se encuentran las conclusiones, las aportaciones útiles tanto para la comunidad científica como para la divulgación de la ciencia; el Apéndice con las tablas e imágenes que sirven como ayuda o complemento del proyecto y por último la bibliografía que se utiliza para la realización de este trabajo.

I

ANÁLISIS PRELIMINAR

1. Análisis preliminar

1.1 Entorno social

El avance de la astronomía en los últimos años, ha traído la necesidad del empleo de nuevos sistemas para el manejo y difusión de los datos obtenidos; la construcción de telescopios, lentes, cámaras, etc., son una parte fundamental del estudio de la astronomía, pero también se requieren más y mejores sistemas para el intercambio de información en cualquier nivel de estudio.

Un punto importante por parte de los investigadores y por cualquier persona que esté realizando algún proyecto, es el dar a conocer a la comunidad en general lo que se ha logrado y en lo que se ha destacado.

La divulgación científica en esta área, tiene una gran importancia tanto para el investigador como para el público en general. En los científicos, el poder intercambiar conocimientos entre ellos y entre la comunidad, el fomentar una cultura científica que es necesaria para el desarrollo en nuestro país, ya que el escaso interés por parte de la comunidad dificulta aspectos que pueden ir desde que no haya gente que estudie astronomía, hasta la falta de fondos económicos.

A pesar de la carencia de muchos recursos, el Instituto ha sabido mantenerse y dar excelentes frutos a la ciencia, tanto a nivel nacional como internacional al lograr realizar estudios en muy diversas áreas de la astronomía.

En el instituto se cuenta con un gran acervo de imágenes astronómicas (obtenidas principalmente del Observatorio Astronómico Nacional de San Pedro Mártir, B.C.N. y Tonanzintla, Puebla) las cuales, por falta de un medio adecuado para su difusión no están disponibles a la comunidad.

Actualmente en nuestro país, no se tiene un sistema de consulta de imágenes para la investigación astronómica que sea eficiente y accesible por las personas, en cualquier parte del país o del extranjero y que pueda ser visto a través del internet; por lo que se inicia el proyecto del Banco de Imágenes Astronómicas del Instituto de Astronomía UNAM (BIA) y con esto, la construcción de la base de datos, las interfaces máquina–usuario, tablas, etc., con el fin de que las personas tengan acceso al BIA, sin importar su formación académica ni el lugar en que se encuentre.

El alcance del proyecto es el de proveer a la comunidad científica de un medio eficiente para la consulta de imágenes astronómicas, que ayude a los trabajos de investigación en una forma mucho más amplia y eficaz, además de que el banco cuenta con una sección de divulgación astronómica para el público en general.

En resumen, el recopilar un gran acervo de imágenes astronómicas, integrar este acervo en un banco y difundirlo ante la comunidad son los principales objetivos del proyecto.

1.2 Banco de Imágenes Astronómicas (BIA)

Una de las características principales del proyecto debe ser su automatización, esto es, que requiera de poco tiempo por parte del personal de cómputo en el mantenimiento del banco; por lo tanto, en el agregado de imágenes (lo que lleva más tiempo), lo podrá realizar cualquier persona acreditada y sólo el borrar información o el modificar algunos registros, así como el agregar nuevos usuarios lo realiza el administrador del sistema, lo cual, se efectúa en pocos minutos.

El sistema se debe componer por tres secciones:

I. Búsqueda de información.
II. Agregar información.
III. Administración del banco.

La primera consistirá en *obtener la información*. Donde se pide elegir un criterio de búsqueda para un tema en general de entre cinco opciones; éstas son:

a) Clave de la imagen. La clave la proponemos en particular para el BIA y se le asignará automáticamente dependiendo del autor del proyecto y de su número de trabajo. Está compuesta por las iniciales del primer nombre y los apellidos del primer autor, además del número de trabajo que se tenga en el banco.
b) Autor. Se buscará por algún nombre o apellido de uno o varios autores, además de que se presentan en segundo plano las opciones que tengan un parecido gramatical.
c) Nombre en catálogo. Es el nombre del objeto celeste estudiado y será propuesto por el autor, de acuerdo al catálogo que elija.
d) Título del proyecto. Es el título de la investigación.
e) Tipo de objeto. Se clasifica por el tipo de objetos celestes, por ejemplo: galaxias, sistema solar, nebulosas, etc.

Además del criterio seleccionado, se pedirá una o más palabras para poder delimitar la búsqueda tomando en cuenta, que entre éstas se pueden utilizar los operadores lógicos: *and* y *or*.

Los reportes al usuario se presentarán en bloques de información que cumplan con las características pedidas, esto es, los bloques tendrán la imagen, el criterio de búsqueda, la clave y el número de identificación (No.Id.), y donde finalmente se creará la liga para acceder a la página final, la cual mostrará toda la información de la imagen en particular.

La segunda sección, a*gregar información* es dónde el banco se retroalimenta. Se pedirá un login y un password, si es acreditado el usuario puede entrar; en caso contrario, se le especificará el

porqué no lo puede hacer, por ejemplo, el login o el password incorrectos.

Al entrar se observará el login, la presentación del formulario y la clave que se le asigna automáticamente por parte del programa. No es necesario hacer el llenado del formulario completo, pero datos como la imagen misma, el nombre del autor, la descripción, un lugar donde se pueda localizar el investigador, el nombre de la imagen y el título del proyecto, tienen un carácter obligatorio. Para la imagen se tendrá una sección para adjuntar el archivo, que realizará la transferencia de la imagen a los archivos del sistema.

La presentación de la página final se podrá observar en una "vista preliminar" y se modificarán los datos regresando al formulario ya descrito. Cuando se está de acuerdo con toda la información se guardarán los registros en la base de datos, al guardarlos se presentan los datos finales y se envía un email de notificación al administrador.

En la última sección, el administrador tendrá la función de agregar a los nuevos usuarios y de modificar o borrar registros en la base; después de acreditarse como tal, se presentará la sección dividida en dos partes, la primera para borrar o modificar datos y la segunda al añadir usuarios.

En la parte de *borrar o modificar datos* se pedirá el número de identificación y se presentará en un formulario toda la información de acuerdo a éste. Aquí se realizará la selección de borrar la información o modificarla, en esta página es donde se realizan los cambios.

Para agregar usuarios, se tomarán cuatro datos en un formulario, el login y el password para la base, el nombre completo del autor y las iniciales para formar la clave de la imagen en sus proyectos.

Se incluirán enunciados de ayuda cuando no se use adecuadamente el sistema o se cometa un error en éste, además de un correo electrónico para tener comunicación con los encargados.

Es probable que surja algún problema cuando se realice una *búsqueda* de información al banco, ya que puede entrar cualquier persona que tenga acceso a la red, sean investigadores, estudiantes o público en general. Para tratar de darle solución a esto, como ya se mencionó, el BIA tendrá un sencillo manejo, ya que seguirá un formato de búsqueda como en una biblioteca, con una selección del tema en general y algunas palabras para hacer una búsqueda más delimitada, además de poder utilizar operadores lógicos entre éstas.

En las otras dos secciones se tendrán sistemas muy accesibles, con ayuda e indicaciones para su uso. Cabe mencionar que para tener acceso a ellas, se requiere de una cuenta (login y password), que serán dados por el administrador del sistema.

1.3 Hardware

En el Instituto se cuenta con diverso equipo en cómputo destinado al trabajo en las investigaciones. Específicamente para el proyecto se tienen tres computadoras con las siguientes características:

Pentium IV 1.2 GHz	**AMD 1800**
Disco Duro 20 Gb	Disco Duro 20 Gb
Tarjeta de Red	Tarjeta de Red
Tarjeta de video 32 Mb	Tarjeta de sonido 32 bits
Monitor 19“	Tarjeta de video 32 Mb
CD CREATIVE 52X	Monitor 19“
CDWRITER	CD CREATIVE 52X
	CDWRITER

Silicon Graphics 320
Pentium III 500 Mhz
Disco Duro 20 Gb
Tarjeta de Red
Tarjeta de sonido 32 bits
Tarjeta de video 32 Mb
Monitor 19”
CD CREATIVE 52X
CDWRITER

También se tienen seis discos de 20 Gb haciendo un total de 120 Gb., que se utilizarán para el almacenamiento de la información, tanto de los registros como de las imágenes.

Y por último, se tienen dos cdwriters más, un scanner y una impresora a color.

II

FUNDAMENTOS TEÓRICOS

2. Fundamentos teóricos

La teoría es la base para todo sistema, ya que no se puede comenzar un proyecto sin antes haber sido documentado ni saber lo que hay detrás de él. Estar informado y saber lo que se ha establecido anteriormente es un buen inicio para cualquier meta, independientemente del área de desarrollo.

En nuestro caso, para el progreso de la tesis es necesario establecer los fundamentos teóricos generales en los cuales se basa para implementar el sistema, las diferentes opciones que existen, las ventajas y desventajas en los temas involucrados, además que en los capítulos siguientes se irán detallando un poco más, si así se requieren.

2.1 Ingeniería del software

En la actualidad, los países dependen de grandes sistemas computacionales que utilizan en todos los ámbitos, desde la bolsa de valores hasta la industria textil; se gastan una gran cantidad de recursos para desarrollar el software de calidad junto con sus herramientas.

Fue en 1968 cuando el término de ingeniería del software se establece por la necesidad de dar solución a lo que en ese entonces se llamó “la crisis del software”, que no fue otra cosa que la construcción sin control de líneas de código que no llevaban una planeación de diseño, una metodología establecida y tampoco una construcción seria del software.

Las primeras conferencias internacionales sobre ingeniería del software fueron organizadas por la *NATO Software Engineering Conference* a finales de 1968 y principios de 1969 en Garmisch y Roma respectivamente. En las reuniones pusieron de manifiesto que la mayoría de los grandes proyectos tenían problemas similares de retraso en los plazos de entrega, un elevado costo y una gran cantidad de fallas y defectos.

La experiencia previa en la construcción de grandes proyectos de software dio a conocer que un *enfoque informal* causaba grandes desventajas en el sistema, los proyectos se retrasaban, tenían un costo superior, eran difíciles de mantener, la tecnología era más lenta, etc. y, por lo tanto, se observó la urgencia de crear y establecer nuevas técnicas para solucionar estos problemas.

En la primera conferencia se convocó a un grupo aproximado de cincuenta expertos y el objetivo de la conferencia era trazar el rumbo que permitiera salir de la crisis del software, lo que estableció el principio para que la construcción desorganizada del software se convirtiera en una ingeniería con teoría y principios.

Entre sus conclusiones se destacó que los principales puntos que hicieron palpable la situación fueron por una parte, la introducción de computadoras de tercera generación y por otra, la falta de metodología que llevaban, por ejemplo:

1. Una gran proporción de los sistemas no llenaban las *expectativas* que de ellos tenían los usuarios.
2. Los *costos* del software eran muy difíciles de prever y a menudo eran muy superiores a lo esperado.
3. Las *modificaciones del software* eran una tarea compleja, costosa y tendía a muchos errores.
4. El software solía tener un mayor *tiempo de entrega* y con menos opciones de las que fueron establecidas en un principio.
5. Era difícil *cambiar* un programa de su entorno hardware, incluso cuando las tareas a realizar eran las mismas.
6. Regularmente no se realizaba un *aprovechamiento* óptimo de los recursos accesibles.
7. Los programas *fallaban* constantemente.

En México es a partir de los años ochenta cuando se le comienza a dar una mayor importancia al proceso del software y un tiempo mayor al manejo de herramientas y de teorías en beneficio de la *ingeniería de software*.

Actualmente, la ingeniería de software ha sido el foco de atención para el desarrollo de todo sistema en cualquier ámbito de trabajo, ya que define el enfoque que se toma cuando el software es tratado por la ingeniería y se acompaña de las tecnologías que existen en el proceso (métodos técnicos y herramientas automatizadas).

El *proceso de ingeniería de software* se entiende como un trabajo en equipo donde participan personas de diferentes áreas, como administradores, gerentes de producto, jefes de proyecto, analistas, programadores de desarrollo y mantenimiento, usuarios finales, entre otros; el número de personas depende de situaciones como el tamaño y especificaciones del sistema, clientes y hasta los fondos económicos disponibles.

Para obtener una definición más universal, tomemos los siguientes párrafos:

“La ingeniería del software es una disciplina que comprende los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza.”[1]

“Ingeniería del software es la aplicación de las ciencias y las matemáticas por las cuales las capacidades del equipo de cómputo se hacen útiles para el hombre por medio de programas de

[1]Sommerville, Ian. *Software Enginnering.* Addison Wesley. Harlow. 2002. Pág. 6.

computación, procedimientos y documentación asociada."[2]

Por lo tanto, definimos que *la ingeniería de software* es el área en la ingeniería que trata el análisis, diseño y la producción de programas de cómputo. Es un *método* que sigue reglas y mecanismos fundamentados en conocimientos científicos, esto es, sigue un modo estructurado y ordenado para sistematizar los conocimientos y obtener un resultado, además de realizar la documentación necesaria para el funcionamiento y mantenimiento del proyecto.

Los ingenieros deben adoptar un enfoque sistemático y organizado para su trabajo, usar las herramientas y técnicas apropiadas dependiendo del problema que tienen que solucionar, las restricciones de desarrollo y los recursos disponibles, estudiar el medio ambiente que va a rodear el sistema y analizarlo de una manera detallada para poder obtener diversas soluciones.

2.2 Desarrollo del proceso del software

Al ser el proceso de ingeniería de software un método (conjunto de actividades y resultados asociados que producen un producto de software), se basa en ciertos procesos y herramientas que lo hacen un sistema de *calidad.* Pressman lo divide en cuatro capas que se ilustran en la Fig. 2-1.

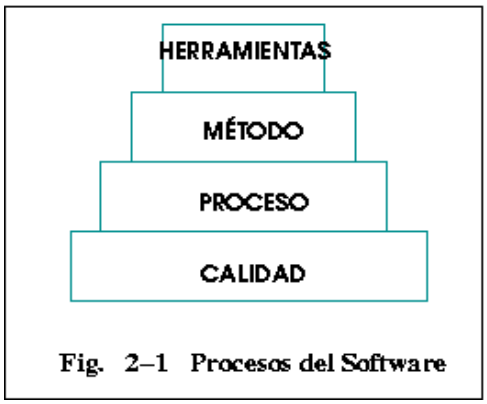


Fig. 2–1 Procesos del Software

A) Todo el enfoque de la ingeniería debe descansar en un camino seguro y confiable, esto es, sobre un enfoque de ***calidad***. Toda la gestión de la calidad forma una cultura para la realización de los proyectos y esto es, lo que da comienzo para el desarrollo de nuevas y mejoras en los sistemas.
B) El ***proceso*** de ingeniería mantiene juntas las capas de la tecnología (herramientas) y

[2]Barry W. Bohem *Software Engineering – As it is.* Proceedings of the 4th international conference on Software engineering. Redondo Beach, CA. 1979.

permite un desarrollo racional y oportuno de la ingeniería del software. Define un marco de trabajo para un conjunto de áreas del proceso que son las bases del control de la gestión de proyectos de software y establece el contexto donde se aplican los métodos técnicos: donde se encuentran los resultados (datos obtenidos y administrados, resultados prácticos, etc.).

C) Los ***métodos*** indican cómo construir técnicamente el proyecto. Abarca el análisis de los requisitos, el diseño, la construcción de los programas, las pruebas y el mantenimiento. La forma de seguir los pasos en la ingeniería del software depende de la metodología que se adopte.

D) Las ***herramientas*** dan un soporte automático o semiautomático para todo el proceso y para los métodos. Cuando se utilizan herramientas creadas por otras herramientas que se pueden utilizar entre sí, se entra a lo que se denomina Ingeniería del Software Asistido por Computadora (CASE).

Tomando ahora en cuenta la *evolución de los procesos*, se observan para éstos, cuatro actividades principales, que son:

1. **Especificación del software**, son las funcionalidades, requisitos y restricciones del sistema, en otras palabras, la *delimitación del sistema.*
2. **Desarrollo del software,** es la planeación y desarrollo del sistema, la construcción de las interfaces, la generación del código, etc., todo esto cumpliendo con los requisitos de la parte anterior. Es el *cómo* se realiza el proyecto.
3. **Validación del software**, es la *acreditación* por parte del cliente y verificación de todos los objetivos, aquí se definen las pruebas finales a realizar.
4. **Evolución del software**, es el *mantenimiento o corrección* del sistema, que se refiere a la adaptación del software respecto al tiempo de funcionamiento.

Dentro de esta última actividad se encuentran las siguientes fases:

- *Corrección*, en cualquier momento el usuario puede encontrar errores, los cuales en el sistema deben ser corregidos de la mejor manera.
- *Adaptación*, conforme pasa el tiempo las características iniciales del sistema cambian y habrá que ir adaptándolo a los cambios del entorno actual.
- *Mejora*, el sistema puede mejorar los requisitos iniciales.
- *Prevención* (reingeniería del software), hace cambios en los programas de computación a fin de que se puedan corregir, adaptar y mejorar antes de ocasionar un problema.

Por lo tanto, en general para el desarrollo del software, podemos destacar los siguientes puntos:

a) Todo el producto del software se basa en los programas desarrollados y en su documentación.
b) Todo sistema de software debe tener como base esencial la mantenibilidad, fiabilidad, utilidad y su rendimiento.
c) Las actividades básicas para la implementación de productos de software son la especificación, el desarrollo, la validación y la evolución de software.

d) Los métodos son las maneras organizadas de producir el software, donde se incluyen las notaciones para ser utilizados, las reglas en las descripciones del sistema que son producidos y algunos lineamientos generales.
e) Existen las herramientas CASE, que son sistemas de software que están diseñadas para soportar actividades rutinarias en el proceso de software, como editar diagramas de diseño, asegurar la consistencia de un diagrama y mantenerse al tanto de las pruebas del programa que han sido ejecutadas.

2.3 Modelos del proceso del software

Para cada sistema se debe seleccionar una estrategia de desarrollo, usualmente se le denomina *modelo de proceso* o *paradigma de ingeniería del software*, que es, una perspectiva abstracta y simplificada de la realidad.

El desarrollo de los *modelos de proceso*, se pueden dividir en cuatro secciones, según define Pressman, la primera: "...*status quo*, estudio del medio externo, desarrollo técnico e integración de soluciones", esto es, el **status quo** representa la situación actual; la segunda es la **definición del problema** (que es similar a la primera etapa que Sommerville establece para las actividades del software: la **especificación del software)**, donde se define la funcionalidad y las restricciones del sistema; la tercera etapa es el **desarrollo del problema** que es la aplicación de alguna tecnología para la resolución de los problemas, (para Sommerville sería la segunda etapa: el **diseño e implementación del software**) y la última para Pressman, **la integración de soluciones** que es la presentación de todas las soluciones viables.

Los últimos puntos que establece Sommerville para las actividades del software, son similares a la última etapa de Pressman, pero toma una perspectiva un poco más hacia la parte del cliente, ya que tiene dos de cuatro puntos que lo involucra directamente: el tercer punto, la **validación del software**, que es la aprobación del sistema por parte del cliente y el cuarto punto, la **evolución del software**, que es el mantenimiento general del sistema.

Existen diferentes tipos de estrategias o modelos, pero principalmente se dividen en *modelos de procesos secuenciales* y *modelos de procesos evolutivos*.

Los **modelos de procesos secuenciales** son aquellos que se les pueden representar gráficamente en una línea, con un principio y un fin relativo, no así los **evolutivos** que tienen una estructura que tiene vueltas o varios caminos a un mismo tiempo en cada una de sus fases.

2.3.1 Modelos de procesos secuenciales

Un objetivo importante de un proceso secuencial es tratar en la medida de lo posible, de no regresar una vez que se ha terminado una actividad y asegurar que cada actividad de desarrollo no introduzca requerimientos no deseados o elimine los esenciales.

2.3.1.1 Modelo de vida clásico y el modelo de cascada

Entre los modelos secuenciales se encuentra *el modelo lineal secuencial* conocido también como el *modelo de vida clásico,* ya que tiene los puntos mínimos para seguir un proceso adecuado:

1. Análisis.
2. Diseño.
3. Código.
4. Prueba.

Desarrollando el modelo lineal secuencial se puede delimitar e*l modelo de cascada*, el cual se basa en lo que realizó Winston Royce[3] aunque omite varios pasos, ya que su proceso se forma básicamente por las siguientes etapas: requerimientos del sistema, requerimientos del software, diseño del programa preliminar, análisis, diseño del programa, código, pruebas y operativilidad. Y también toma en cuenta otras etapas, como la generación de los requerimientos del sistema y las revisiones de software preliminar, la revisión crítica del software y la del software final.

Pero los más manejados por la mayoría de los autores y por lo tanto, los puntos que tomamos para la estructura de cascada son:

a) ***Análisis***, se *establecen* los requisitos de todo el proyecto, se delimitan alcances y limitaciones y la mayor importancia es la reunión con el usuario del proyecto. El usuario analiza y explica los requisitos del sistema.
b) ***Diseño***, establece el *cómo* se va a realizar el proyecto, tiene cuatro atributos que son: estructura de datos, arquitectura del software, representaciones de interfaz y detalle procedimental. La documentación como el diccionario de datos, la ayuda integrada al sistema o manuales escritos, que por parte del ingeniero juegan un rol muy importante.
c) ***Código***, es la *generación del código* para la máquina, siguiendo los pasos que se delimitaron en la etapa anterior, su documentación puede ser dentro del programa o fuera de éste, según las características del sistema.
d) ***Prueba***, al concluirse el proyecto se realizan *pruebas finales* por parte de los ingenieros,

[3] Royce, Winston W. *International Conference on Software Engineering*. Monterey, California, United States. 1987.

para comprobar si los objetivos y expectativas se cumplen satisfactoriamente. Las pruebas alternas que realizan los usuarios finales cuando se ha instalado el sistema, varían significativamente de acuerdo al medio en el que se desenvuelvan.

e) ***Mantenimiento***, el entorno de cualquier sistema cambia conforme pasa el tiempo y por lo tanto hay que ir *modificando y adaptando* el sistema a los cambios.

El diagrama que muestra el Modelo de Cascada se muestra en la Fig. 2-2.

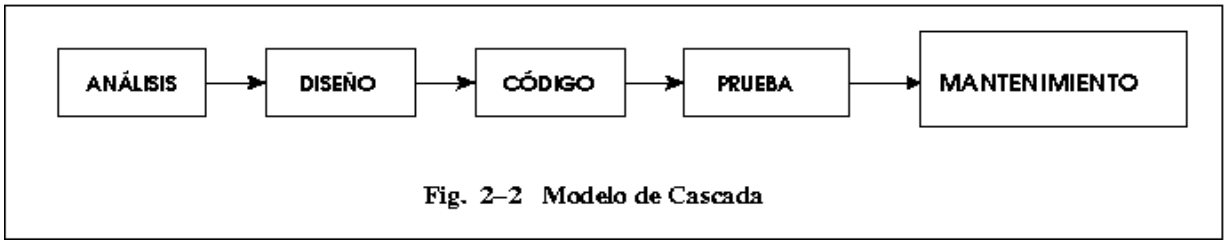


Fig. 2–2 Modelo de Cascada

2.3.1.2 Modelo de construcción de prototipos

Otro modelo secuencial es *el modelo de construcción de prototipos*, surge cuando el sistema no está definido del todo, se tienen sólo algunas ideas y la primera entrevista puede ser vaga y superficial, el ingeniero responsable no tiene la seguridad de la eficacia de un algoritmo ni tampoco de la capacidad de adaptación del sistema operativo o de la forma de las interfaces al usuario. Se realiza un prototipo y funciona el modelo por el mejoramiento de éstos. El modelo de construcción de prototipos se muestra en la Fig. 2-3.

Tiene tres pasos a seguir:

- *Escuchar al cliente*. Se estipula el análisis y la planificación del proyecto, se definen los objetivos y ciertas características. Es similar a la etapa de *análisis* en el modelo anterior, pero con menor detalle.
- *Construcción y revisión de la maqueta*. Aparece un diseño rápido, un prototipo, con las variables principales y más generales del sistema.
- *Revisión del prototipo*. El cliente prueba el sistema y lo utiliza para refinar los requisitos del software.

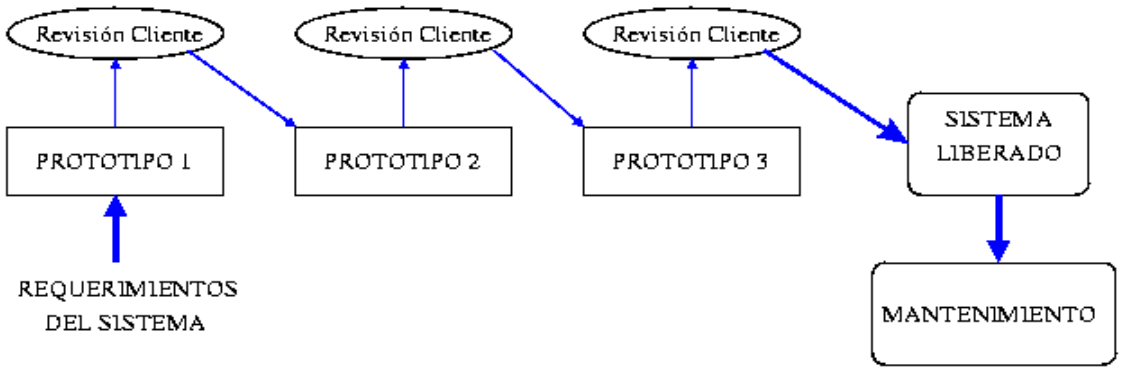


Fig. 2–3 Modelo de Construcción de Prototipos

Lo ideal es que el último paso sea una manera de identificar los requisitos siguientes, pero al terminar el *primer* prototipo muchas veces se comprueba que es demasiado lento, grande y con poca calidad y posiblemente tendrá que ser desechado.

La principal desventaja es que una vez que el cliente ha dado su aprobación final al prototipo y cree que está a punto de recibir el proyecto final, se encuentra con que es necesario rescribir buena parte del prototipo para hacerlo funcional, porque lo más seguro es que el desarrollador haya hecho compromisos de implementación para hacer que el prototipo funcione rápidamente. Por lo tanto, es posible que ahora el último prototipo sea no muy amigable en su uso o incluso, que esté escrito en un lenguaje de programación inadecuado.

Aún así, el método de prototipos es un modelo que siguiendo estrictamente algunos lineamientos (por ejemplo trabajar en un lenguaje de programación o tomar las mismas referencias) puede dar resultados muy satisfactorios.

2.3.1.3 Modelo DRA

El último modelo secuencial descrito es el *modelo DRA (Rapid Application Developement)*, que enfatiza un ciclo de desarrollo extremadamente corto y por lo tanto, rápido.

La velocidad con la que trabaja se debe a que su construcción está basada en componentes a diferentes niveles, esto es, el proyecto debe poder modularizarse de forma que permita completarse cada una de las funciones en un tiempo corto, por un equipo específico y al final poder complementarlas en un conjunto.

En la Fig. 2-4 muestra el modelo DRA.

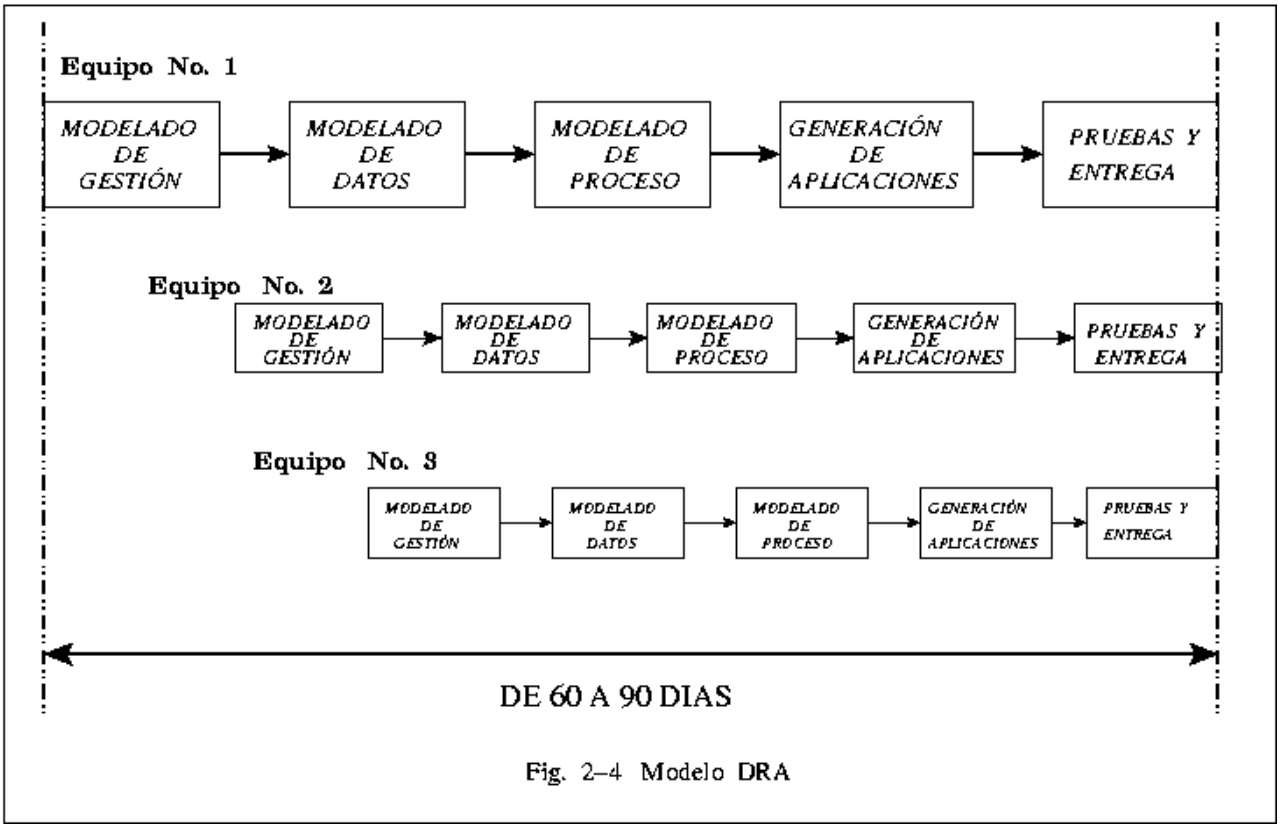


Fig. 2–4 Modelo DRA

Si se lleva un adecuado seguimiento y si se utiliza para aplicaciones de sistemas de información, tiene las siguientes fases: el *modelado de gestión*, el *modelado de datos*, el *modelado de proceso*, la *generación de aplicaciones* y por último, *pruebas y entrega.*

Sus ventajas son:

a) Se comprenden bien los requisitos y se limita el ámbito del proyecto.
b) Es fácil dividir al sistema en módulos y así trabajarlos.
c) Se utiliza un enfoque de construcción basado en objetos reutilizables.

Y las desventajas:

a) Requiere recursos humanos suficientes como para crear el número correcto de equipos.
b) Necesita que el cliente y el desarrollador se comprometan en las actividades necesarias para completar un sistema en un tiempo corto.
c) Se hace muy complicado si el sistema no se puede dividir en módulos.

2.3.2 Modelos de procesos evolutivos

Los modelos de procesos evolutivos conjuntan la información de los elementos de los modelos lineales en una forma de crecimiento, esto es, tienen una vida basada en ciclos que van mejorando ya sea por incrementos, por diversos caminos o regenerándose en un mismo diagrama de trabajo conforme transcurre el tiempo.

Los modelos son útiles cuando el sistema tiende a cambiar continuamente su entorno y las especificaciones iniciales. El primer incremento regularmente es un prototipo que el cliente califica y evalúa y es aquí donde se establecen las condiciones para continuar un siguiente incremento. Es también muy práctico cuando no existe el personal necesario para la conclusión del proyecto en su totalidad.

2.3.2.1 Modelo incremental

El primer modelo de los procesos evolutivos es el *modelo incremental*, es un modelo lineal pero con varios niveles de trabajo. El primer nivel es el que refleja los requisitos básicos y en base a éste el cliente evalúa y da pie para los siguientes bloques del sistema, que pueden ser o no independientes entre sí, pero la mayor diferencia con respecto a las anteriores es que cada secuencia lineal produce un producto completamente operacional, que se complementa con los otros bloques funcionales, por ejemplo, son versiones que se pueden juntar como las piezas de un rompecabezas.

La Fig. 2-5 muestra el esquema del modelo incremental.

Tiene la ventaja de que la comunicación con el cliente es más cercana, ya que continuamente se le entrega un avance del trabajo, pero existe gran dificultad al hacer la evaluación del costo total sobre el proyecto por la gran variabilidad de los objetivos, por lo tanto, se requiere de gente muy experimentada.

2.3.2.2 Modelo en espiral

El *modelo en espiral* es otro modelo de procesos evolutivos, avanza por etapas que giran en espiral hacia afuera empezando por un esbozo inicial; el término de un ciclo, es el comienzo a otro nuevo mejorando al anterior. Su ventaja principal es que se pueden implementar diversos procesos en cada vuelta de la espira, esto se refiere a que puede tomar los métodos de otros modelos para adaptarlos a una necesidad específica en cada una de sus iteraciones.

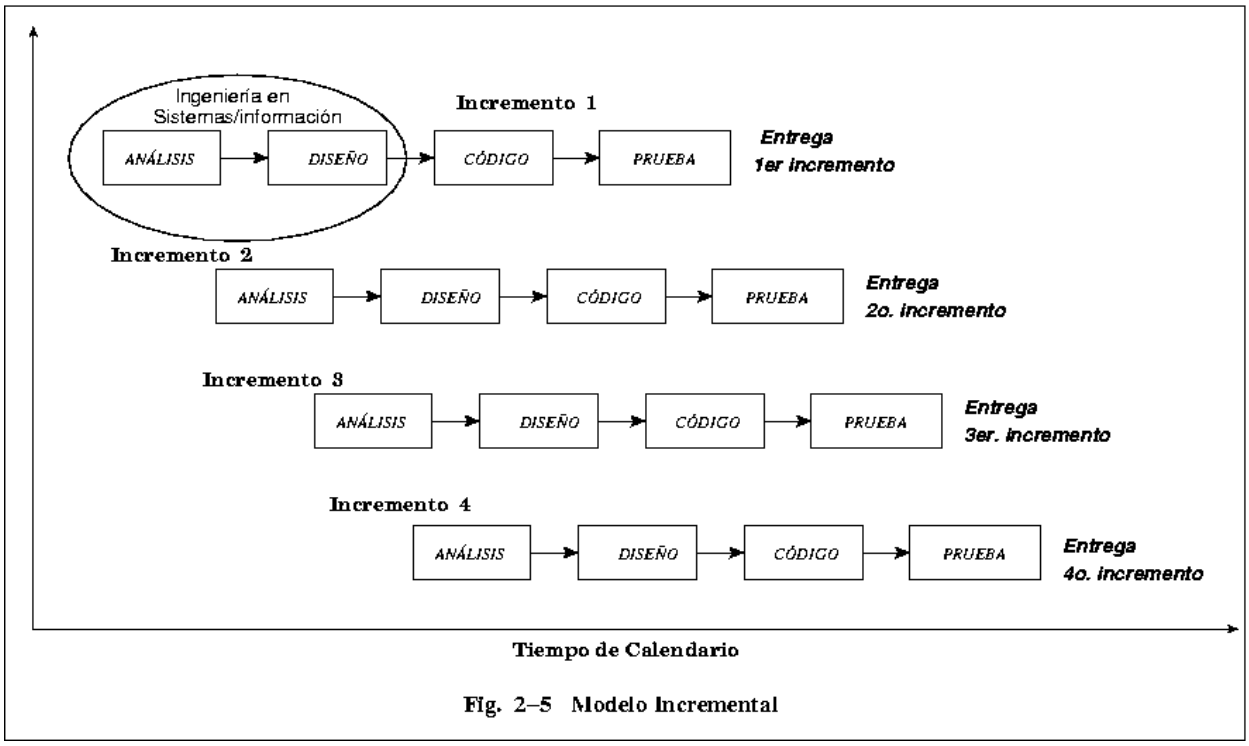


Fig. 2–5 Modelo Incremental

El modelo en espiral es propuesto originalmente por Bohem y una definición es la siguiente: "El modelo es un proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial."[4]

El modelo en espiral se basa en el dibujo de una espira, cada vuelta se desarrolla en base a las anteriores y cada vez que se tiene una nueva vuelta el proyecto estará más detallado y más cerca del proyecto terminado. En las primeras iteraciones el sistema podría comenzar en el papel o en un prototipo muy sencillo.

Generalmente el modelo se constituye de cuatro estrategias:

1.- *Definición de objetivos*, se determinan los objetivos, soluciones, alternativas y restricciones del proyecto. Se estipula un plan detallado de administración y se realizan los estudios de riesgos.

2.- *Evaluación y reducción de riesgos*, se analizan los riesgos, las decisiones del punto anterior y dependiendo de éstos, se planean las estrategias y planes de contingencia. Aquí pueden ser construidos prototipos de simulación del software.

[4]Pressman, Roger S. *Ingeniería del software: un enfoque práctico*. McGraw Hill. España. 1998 . Pág. 26.

3.- *Desarrollo del sistema*, se consideran las actividades del desarrollo, incluyendo el diseño, la codificación y la verificación. Se toman en cuenta las conclusiones anteriores, características y requisitos para las nuevas interacciones. Aquí se establece la interfaz máquina-usuario y sus características principales.

4.- *Revisión*, se revisan las etapas anteriores y se planea (si es que el sistema lo requiere) la siguiente iteración de acuerdo a los resultados obtenidos de las estrategias pasadas, se puede optar por una nueva o continuar trabajando en la misma.

Un modelo más apegado al proyecto real se ilustra en la Fig. 2-6, donde cada vuelta representa un prototipo que se va acumulando a través de la espiral y cada sector corresponde a una estrategia de desenvolvimiento.

El modelo se divide en áreas o secciones, llamadas actividades estructurales o regiones de tareas. Roger S. Pressman propone seis tareas:

1. Comunicación con el cliente, son las tareas para establecer el trato entre el cliente y el desarrollador.
2. Planificación, se establecen los recursos, necesidades y alguna otra prioridad para el proyecto.
3. Análisis de riesgos, se realiza la evaluación de los riesgos técnicos y de gestión.
4. Desarrollo e ingeniería, aquí se construyen las representaciones de la aplicación.
5. Construcción y adaptación, se realizan las tareas necesarias para construir, probar, instalar y proporcionar soporte al usuario.
6. Evaluación con el cliente, el cliente evalúa y, aprueba o rechaza el proyecto, si es rechazado, se establecen nuevas bases de arranque para la siguiente iteración.

Cada ciclo inicia con la determinación de objetivos, alternativas y restricciones, y continúa con las antes descritas: verificación de alternativas y riesgos (si en algún momento los riesgos son mayores, aquí el proyecto se detiene o se toma una nueva estrategia), el siguiente es el desarrollo y el último paso es considerar los cambios anteriores para preparar un nuevo ciclo.

Las ventajas con las que se cuentan, es que tiene una gran flexibilidad para poder intercalar otros métodos y se pueden evitar algunos riesgos en la tercera etapa, ya que la dificultad de éstos, van variando conforme aumenta el costo acumulado.

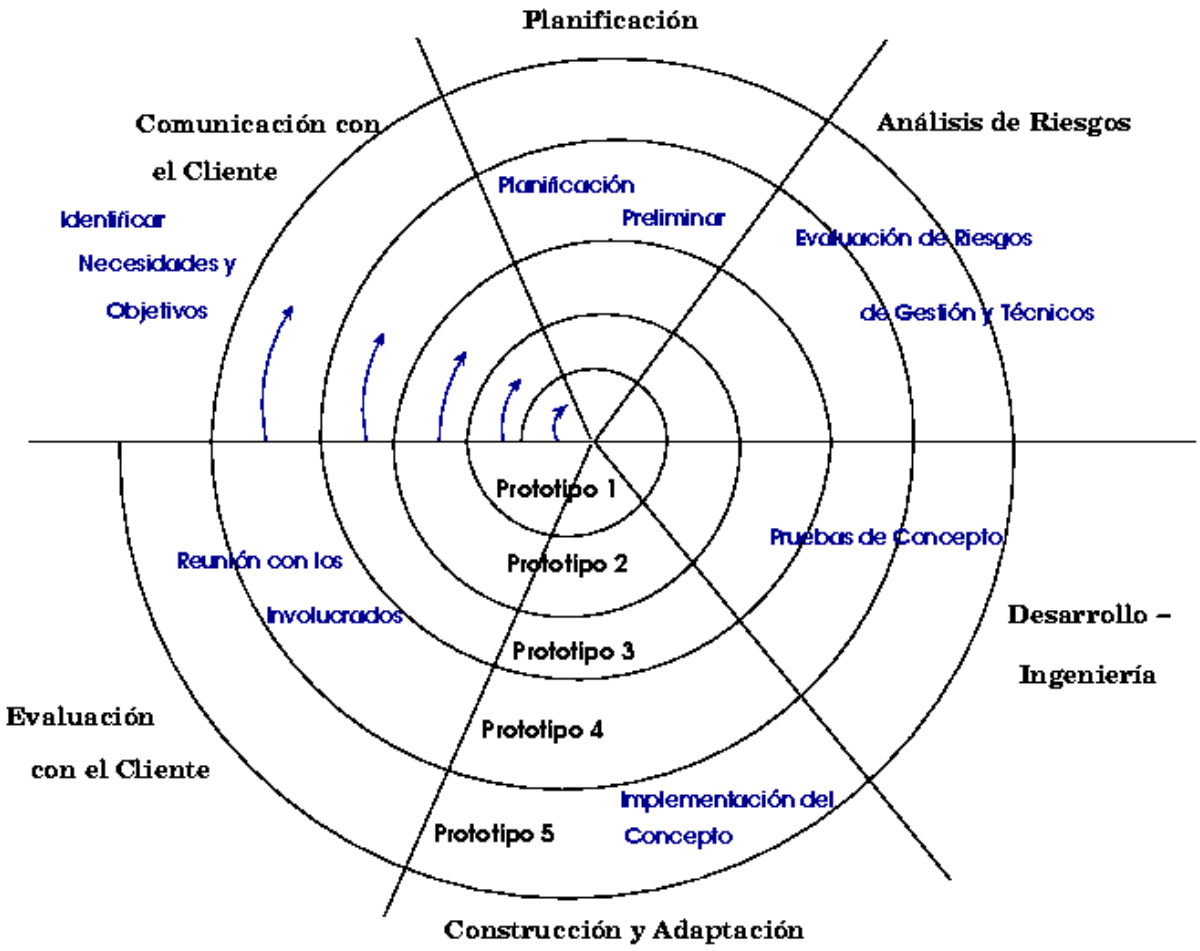


Fig. 2–6 Modelo en Espiral

A diferencia del modelo de proceso clásico que termina cuando se entrega el software, el modelo en espiral puede continuar adaptándose a la vida del software de computadora, puesto que hay veces que algún proceso está inactivo, pero siempre que surja o inicie un cambio, dicho proceso arranca en el punto de entrada adecuado.

En cada vuelta pueden ir intercalándose diversos métodos, por lo tanto, existen muchas posibilidades para alcanzar los objetivos, pero su ventaja también es una de las principales desventajas si no se tiene un criterio bien definido, ya que tiene que haber gente de experiencia que identifique los riesgos y tome decisiones acertadas ante la gran cantidad de caminos que se puedan tomar.

2.3.2.3 Modelo de ensamblaje de componentes

En el modelo de *ensamblaje de componentes* se toma en cuenta el paradigma de la *orientación a objetos.*

Ninguno de los métodos tratados anteriormente se adapta a la *orientación de objetos*, ya que tendrían que reconocer el cambio gradual de los sistemas, por lo que se requiere un modelo evolutivo de proceso que fomente la reutilización de componentes; por lo tanto, el más apegado a estos requerimientos es el modelo en espiral.

Su primera iteración que es similar a los modelos anteriores, comienza con la entrevista con el cliente donde se delimitan los objetivos del problema y la identificación de "clases", tomando como definición que las *clases* son un concepto de la *orientación a objetos* que encapsulan las abstracciones de datos y procedimientos que se requieren para describir el contenido y comportamiento de alguna entidad del mundo real.

Las etapas siguientes son similares al método en espiral, aunque en las etapas de *planificación* y de c*onstrucción y adaptación* se marcan algunas diferencias. Pero el principal punto de diferencia que hay entre éste y el modelo en espiral, es que en cada una de las tareas delimitadas se tiene que llevar el análisis, el diseño, la programación y las pruebas de la programación orientada a objetos.

Lo anterior citado se muestra en la Fig. 2-7 donde se observa que cada región tiene los pasos para clasificarse en una programación orientada a objetos.

En la primera etapa se delimitan las "clases" que pueden ser utilizadas en cualquier región, ya que por definición pueden ser reutilizables en cualquier aplicación, por lo que disminuye el trabajo en un porcentaje muy considerable.

Las clases que se crean se van almacenando en una "biblioteca" donde se encuentran los datos y los algoritmos que alguna vez fueron utilizados, además de los nuevos que se van creando.

Aunque principalmente la importancia del modelo radica en el tamaño de la biblioteca, es notable el ahorro de trabajo que se establece aún si no se cuenta con una muy extensa

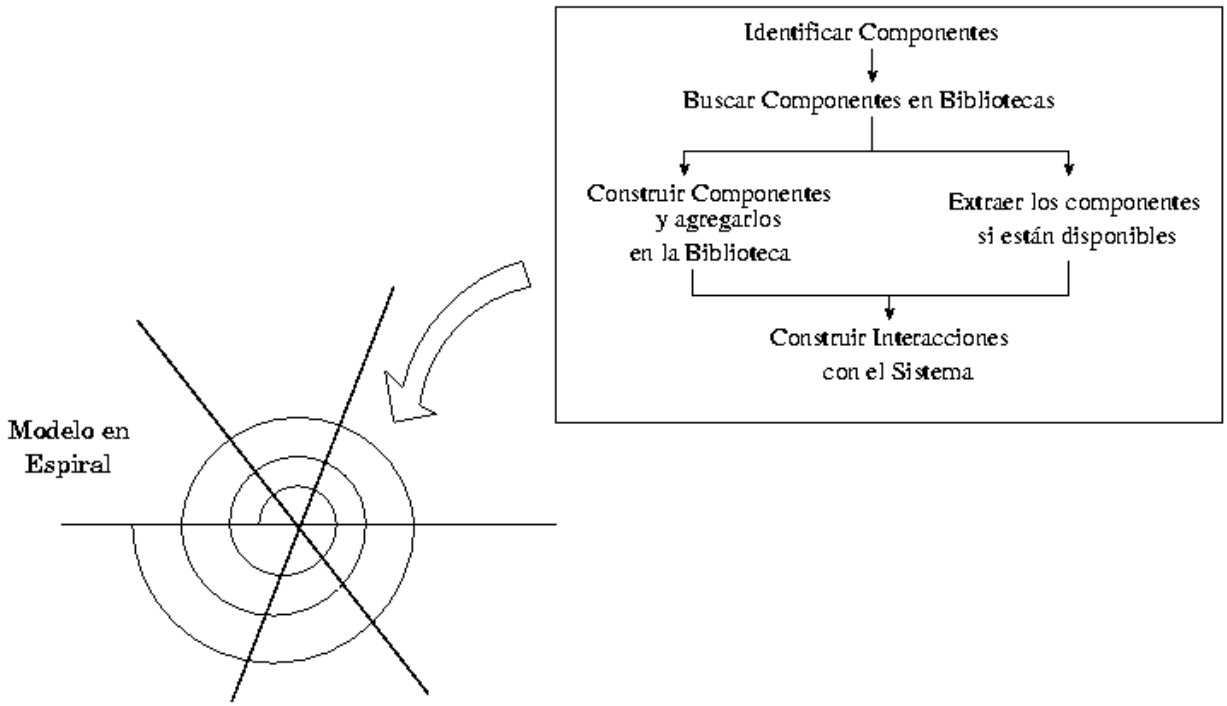


Figura 2–7 Modelo de Ensamblaje de Componentes

.

2.3.2.4 Modelo de desarrollo concurrente

El modelo de desarrollo concurrente, forma parte al igual que el modelo en espiral de los métodos evolutivos que hoy en día están cobrando un gran auge en las grandes empresas.

En la realización del sistema de información existe mucho personal que se ve involucrado en la realización de éste, por lo que el flujo de información suele ser muy grande y muchas veces inadecuado, la solución que se viene estructurando es el poder definir una estructura neutra de datos y, por tanto, especificar la información a intercambiar entre las distintas aplicaciones.

Se puede representar gráficamente en forma de diagrama de bloques, ya que existen diferentes tareas y estados asociados entre sí, con un flujo de datos basados en el modelo en espiral.

Cada una de las entidades de este diagrama son regiones de tareas (principalmente del modelo en espiral). La actividad (análisis) y la comunicación con el usuario (comunicación con el cliente) se pueden establecer en cualquier entidad del diagrama, ya que en todas se tiene un peso específico. Las otras regiones se pueden distribuir de una manera similar, esto es, al principio del proyecto, cuando se finaliza la *comunicación con el cliente* se ha terminado el primer bloque y

comienza el estado de cambios en espera y la actividad de *análisis*, no se encuentra en uno delimitado, por lo tanto, se considera en bajo desarrollo, pero si el cliente hace cambios entonces se traslada de bajo desarrollo a cambios en espera (*ingeniería, construcción y adaptación*).

"En realidad, el modelo de desarrollo concurrente es aplicable a todo tipo de desarrollo de software y proporciona una imagen exacta del estado actual de un proyecto. En vez de confinar las actividades de ingeniería de software a una secuencia de sucesos, define una red de actividades. Todas las actividades de la red existen simultáneamente con otras. Los sucesos generados dentro de una actividad dada o en algún otro lugar en la red de actividad inicia las transiciones entre los estados de una actividad."[5]

La Fig. 2-8 muestra la estructura básica de un elemento del modelo de desarrollo concurrente.

Una de las ventajas es la disminución en el tiempo total transcurrido y la reducción de costos, pero obliga a avanzar en la toma de decisiones en cada etapa, lo que conlleva a que las decisiones son cada vez más rápidas y por tanto se corre el riesgo de decidir a partir de una mala o poca información, tomando en cuenta que el flujo de información es mucho más complejo y exige una gran agilidad, especialmente cuando dicho flujo se produce entre distintas empresas que colaboran en el desarrollo de un mismo proyecto.

2.4 Gestión de riesgo

El análisis sobre el riesgo es un tema sumamente delicado, ya que se tienen que tomar en cuenta muchos puntos predecibles y otros que no los son. Si se toma en cuenta que pueden existir a futuro cambios de opinión del personal, cambios de lugares, acciones, etc. y que también lo puede no haber, es algo que no tiene una seguridad y por lo tanto, muy peligroso para el análisis de la ingeniería del software.

El objetivo de realizar la gestión del riesgo es identificar, estudiar y eliminar las *fuentes de riesgo* antes de que exploten y hagan estragos que generen problemas más grandes.

Las características que por lo regular se enumeran cuando se habla de riesgo son: la *incertidumbre*, ya que nunca se está seguro de que pueda ocurrir un hecho indeseable y la *pérdida*, ya que si ocurren tales hechos habrán diversas consecuencias.

[5]Pressman, Roger S. *Ingeniería del software: un enfoque práctico*. McGraw Hill. España. 1998. Pág. 31.

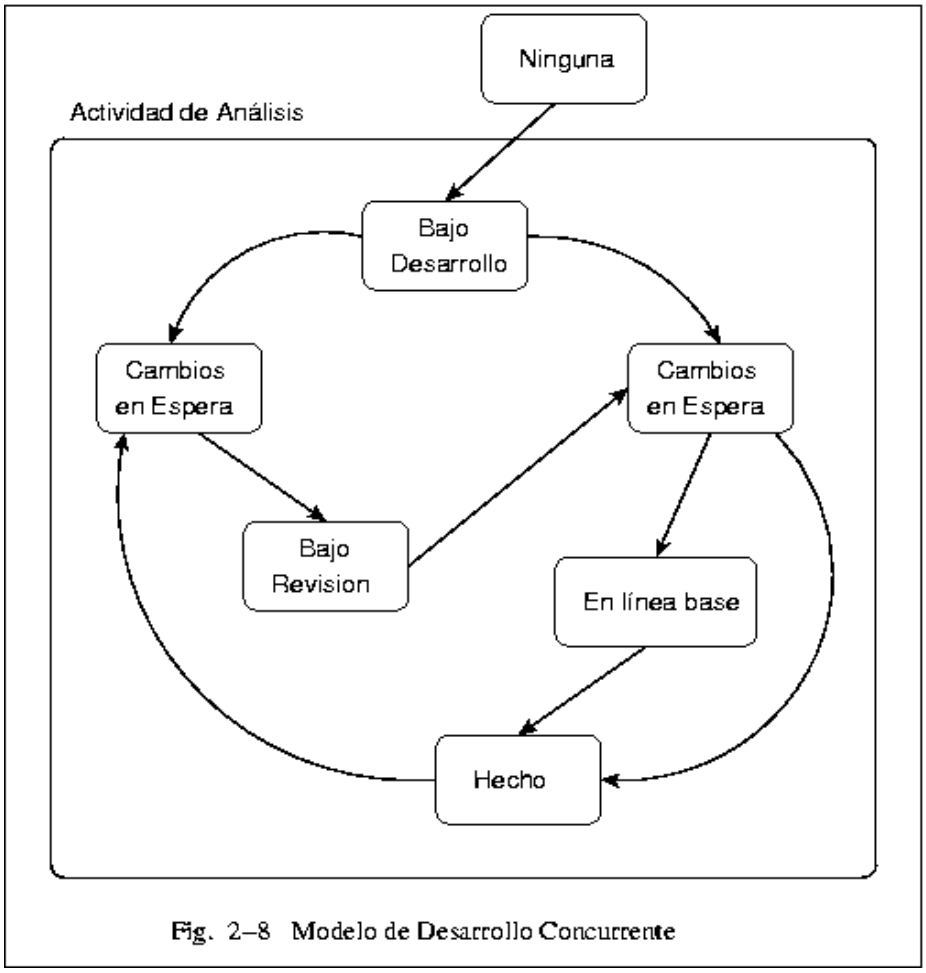


Fig. 2–8 Modelo de Desarrollo Concurrente

El comienzo de este proceso se basa en tres características principales que son:

- Las características del futuro, ¿cuáles son los riesgos que podrían hacer que el proyecto fracasara?
- Las consecuencias, ¿cuáles serían las modificaciones que hay que realizar en el contorno del proyecto para adaptarlo a los nuevos cambios?, ¿se tendrían otras consecuencias?
- Respecto a las elecciones ¿qué métodos y herramientas hay que emplear para los cambios, se utilizará una nueva tecnología o habrá nuevo personal?

Las anteriores son preguntas en las que nos podemos basar para minimizar los riesgos o tomar los "riesgos adecuados". Un riesgo conocido o predecible es un riesgo que puede ser evitado.

El jefe de desarrollo debe supervisar el proyecto de manera que pueda controlar cualquier tipo de circunstancia adversa antes de que se tengan dimensiones mayores y por lo tanto, requiera de un mayor número de modificaciones.

Los tipos de riesgos varían conforme se establece cada proyecto, pero se pueden clasificar en:

a) *Riesgos administrativos*, incluyen cualquier tipo de eventualidad relacionada con la organización, los productos de trabajo, agentes exteriores o con el plan de tareas.
b) *Riesgos técnicos*, incluyen cualquier tipo de incertidumbre relacionados con los modelos del mismo sistema, incluyendo los cambios de funcionalidad del sistema, la arquitectura o la implantación de éste. Aquí se contemplan los riesgos que surgen en el transcurso del proyecto.

Una clasificación más detallada es la siguiente:

- *Los riesgos organizacionales*, identifican los problemas potenciales de presupuestos, recursos, clientes y requisitos generales del proyecto.

- *Los riesgos de personal*, se enfocan a los problemas ocasionados por el personal involucrado, ya sea directa o indirectamente.

- *Los riesgos técnicos*, identifican problemas con el diseño, la interfaz máquina–usuario, la verificación y el mantenimiento del equipo.

- *Los riesgos de tecnología*, se enfocan a los problemas de utilización de nuevas tecnologías de software o hardware, además de la estimación en la tecnología utilizada, si es una tecnología de punta o si los métodos utilizados son eficientes.

- *Los riesgos del negocio*, están relacionados con los problemas en el mercado; por ejemplo el realizar un programa excelente y, por lo tanto, más caro que no tenga comprador, construir algo que nadie lo necesite o simplemente perder el presupuesto a la mitad del proyecto.

- *Los riesgos conocidos*, son aquellos que se pueden visualizar después de una evaluación en el proyecto.

- *Los riesgos de requerimientos*, son los que se identifican con cualquier cambio que llegase a realizar el cliente y el proceso de administrarlo.

- *Los riesgos predecibles*, se encuentran principalmente por la experiencia previa en el desarrollo de sistemas.

- *Los riesgos impredecibles*, los cuales surgen de situaciones muy complicadas, son extremadamente difíciles de predecir.

Pero en particular, se tienen identificados algunos riesgos por la cotidianidad de éstos en los proyectos de software.

1. Los cambios de requisitos.
2. Escatimar la calidad.
3. Diseño inadecuado.
4. Un proyecto orientado a la investigación más que al desarrollo.
5. Personal mediocre.
6. Errores en la contratación.
7. Diferencias con el cliente.
8. Riesgos del tamaño del producto.
9. El impacto en el negocio, entre otros.

El análisis de riesgo se mide con la probabilidad de que éste sea real y las consecuencias asociadas. Según estudios realizados, la exposición de riesgo en el proyecto es la probabilidad de ocurrencia del riesgo, multiplicada por la magnitud de pérdida de éste, por ejemplo, si existe el 20% de probabilidad de que ocurriera un riesgo que retrasaría el proyecto en cinco semanas, entonces la exposición del riesgo sería de una semana.

Lo anterior nos ayuda para darnos una idea de la exposición de nuestro proyecto a un posible riesgo. También se recomienda realizar una tabla en donde se visualice en un eje horizontal cada uno de los posibles riesgos, como personal, tecnología, costos, etc. y en el otro eje una perspectiva del riesgo, por ejemplo, despreciable, importante o marginal y si es posible los efectos que llegasen a ocasionar.

Tampoco hay que olvidar dar una **prioridad** a cada riesgo para poder centrar un poco más los objetivos de la estrategia.

Para *evitar el riesgo* se pueden hacer una serie de sugerencias, que son:

- Reuniones con todos los involucrados en el sistema.
- Tener una excelente organización y equidad en el trabajo.
- Comunicar los riesgos a toda persona involucrada para que pueda estar prevenida.
- Definir estándares de documentación.
- Documentar bien cada uno de los riesgos encontrados
- No realizar actividades arriesgadas..
- Eliminar el origen del riesgo.

Cuando se tiene ya un amplio conocimiento de los posibles riesgos, se formulan los planes de contingencia, los cuales son una serie de pasos ya estructurados que surgen cuando se produce un evento causado por los riesgos antes identificados.

Los tres puntos más importantes para la estrategia final es tener en consideración ante todo el **evitar el riesgo**, al comenzar el proyecto una **supervisión seria** y finalmente si ya es inevitable el problema, poner en marcha un **plan de contingencia** que pueda sacar el proyecto de la mejor manera.

2.5 Administración del proyecto

La administración se define como los métodos para coordinar y organizar un proyecto. El principal actor para la administración de un proceso de software es el gerente, que proporciona, administra y coordina los recursos, asegurando una alta calidad en el sistema, un costo conveniente tanto para la empresa como para el cliente y el tiempo necesario para realizar el sistema.

Las principales herramientas que el gerente de software utiliza son la planeación, la supervisión, la administración del riesgo y el manejo de las contingencias para enfrentar problemas similares a los que enfrentan los desarrolladores técnicos, la complejidad, los cambios, los riesgos, etc., pero a mayores escalas.

2.5.1 Panorama general de la administración de proyectos

En el desarrollo de la administración se delimitan tres puntos importantes: el *inicio del proyecto,* el *estado estable* y la *terminación del producto*.

1. Los inicios del proyecto, los únicos involucrados son el gerente, el cliente y probablemente algunos líderes de proyecto, que delimitan el alcance del sistema desde el punto de vista de funcionalidad, restricciones y productos a entregar.

Se puede dividir en las siguientes etapas:

a) *Establecimiento de las normas para el proyecto,* funcionalidad, fechas, límites, etc. Se realiza por parte del gerente, el cliente y algunos líderes de proyecto.

b) *Descomposición del proyecto,* el gerente y los líderes de proyectos descomponen el sistema en subsistemas, hacen equipos y asignan trabajos a cada uno de ellos. En este momento realizan un análisis detallado de los requerimientos y especificaciones del subsistema asignado y el sublíder de proyecto asigna tareas específicas. Si hay necesidad de cambios o reentrenamientos de personal aquí se notifican.

c) *Establecimiento de medios de comunicación*, se establecen las formas de comunicación entre los equipos desde fechas de reuniones, tiempos de asesoría, juntas con los clientes hasta tableros de noticias, sitios Web, herramientas para la administración de la configuración, etc.

d) *Planeación de tiempos*, es la asignación de un calendario por parte del gerente a todos los trabajos intermedios y los productos a entregar de forma externa y se les asigna a los equipos.

2. Estado estable, es la parte de desarrollo del sistema y el principal responsable es el líder del proyecto, éste y los sublíderes llevan un seguimiento del proyecto reportando fallas, retrasos, ganancias de tiempos, etc. y valoran situaciones para reasignar tareas, mover personal, entre otros. Los siguientes puntos delimitan esta etapa:

I. *Inicio oficial del proyecto,* se establece oficialmente la fecha de inicio y de entrega entre el cliente y la compañía. Se asienta la fecha oficial del inicio del desarrollo del sistema.
II. *Supervisión de estado,* se supervisa el desarrollo del proyecto y que vaya de acuerdo a la calendarización establecida, se recopila información mediante reuniones, reportes de problemas y se preparan planes de contingencia. Si se lleva una revisión adecuada se pueden identificar la mayoría de riesgos y por lo tanto, analizarlos y darles una priorización.
III. *Replaneación del riesgo,* cuando un riesgo no se puede resolver en los tiempos previstos el líder de proyecto tiene que realizar una recalendarización en el subsistema de modo que no afecte a todo el sistema, si no es posible, deben seguir los planes de contingencia, como lo pueden ser reasignar personal o equipo, contratación de nuevo personal (que la mayoría de las veces no es factible), intercalar equipos, etc.

3. Terminación del proyecto, durante esta etapa se entregan y recopilan los detalles o datos del proyecto que pudiesen servir en un futuro. Se establecen las siguientes divisiones:

a) *Aceptación del sistema por parte del cliente;* analizan entre el cliente y el administrador al sistema, de acuerdo a lo establecido en los lineamientos iniciales.

b) *Instalación;* se instala el sistema en el medio donde va a operar y se entregan documentos e información adyacente. Puede haber entrenamientos para los usuarios y tal vez un tiempo mientras se transfieren los datos del sistema anterior al actual.

c) *Punto final*; se termina oficialmente el sistema. Se recopila la historia del proyecto e información de aprendizaje útil para la organización.

2.5.2 Principales conceptos de la administración

El proceso de administración maneja principalmente los siguientes conceptos:

a. Equipos

El equipo es un grupo de personas que trabajan sobre un mismo subsistema, es el resultado de la división de un proyecto más grande y más complejo. La división en subsistemas es básica para el desarrollo del proyecto ya que se tienen mayores ventajas para afrontarlo.

Cada subsistema tiene las mismas divisiones, métodos y estructuras del sistema en general, tiene un sublíder, subalternos, gente de reporte, etc.

Cada equipo es responsable de hacer ingeniería de software en el subsistema a cargo, el análisis, el diseño, la codificación y las pruebas finales y, todos los puntos que conllevan como el análisis de riesgos, planes de contingencia, diseño de datos, etc.

La información es básica entre cada equipo y debe tener un flujo tanto ascendente, como descendente y moverse a ambos lados.

b. Papeles

Es la asignación de responsabilidades y derechos a cada uno de los participantes del equipo. El papel, según establece Bruegge, define los tipos de tareas técnicas y administrativas que se espera de una persona.

La persona tiene que establecer el conjunto de hechos y de pruebas para que su sistema trabaje. Existen diferentes tipos de papeles, como los *administrativos*, que son relacionados con la organización y ejecución del proyecto; de *desarrollo*, establecen la especificación, el diseño y la construcción de subsistemas; de *funcionalidad cruzada*, relacionados con la coordinación de varios equipos; de *consultor*, que da apoyo a los equipos, ya sea por falta de experiencia o por temas que no se dominen, o de *promotor*, que se relacionan con la promoción de cambios en la organización.

c. Productos de trabajo

Son los productos que se van haciendo a lo largo del desarrollo, pueden ser documentos o un proyecto, ya sea para otros equipos o para entrega del cliente. Los productos de trabajo por lo regular alimentan a otras tareas y son sujetos a tiempos bien delimitados.

Por el establecimiento de sus tiempos, sirven como artefactos de administración importantes, ya que se puede valorar su entrega y por lo tanto, el inicio de las tareas que dependen de los productos de trabajo específicos.

d. Calendarización

Todos los sistemas deben llevar un tiempo establecido, por lo tanto se realiza una calendarización, que es la relación que existe entre el tiempo y la realización de una tarea. A cada tarea se le asigna un tiempo de inicio y de término.

Aquí, hay que organizar los tiempos para que lleven una coherencia en la realización del sistema y que sean acordes a la realidad. Para esto, se pueden utilizar los diagramas de PERT o Gantt que permiten ver el orden de las tareas y cuáles de ellas se pueden realizar simultánea o consecutivamente.

El diagrama de Gantt es "una gráfica de barras en donde el eje horizontal representa el tiempo. El eje vertical representa las diferentes tareas a realizar. Las tareas se representan con barras cuya longitud corresponde a la duración planeada de la tarea."[6]

La gráfica PERT, "representa una calendarización como una gráfica acíclica de tareas. El inicio y la duración planeados de la tarea se usan para calcular la ruta crítica, que representa la ruta más corta posible a través de la gráfica. La longitud de la ruta crítica corresponde a la calendarización más corta posible, suponiendo que se tienen los recursos suficientes para realizar en paralelo las tareas que son independientes".[7]

La Fig. 2-9 muestra el esquema de una gráfica PERT. La ruta crítica se muestra con la línea punteada y las tareas llevan una secuencia de tiempos donde las dos últimas se pueden realizar a un mismo tiempo.

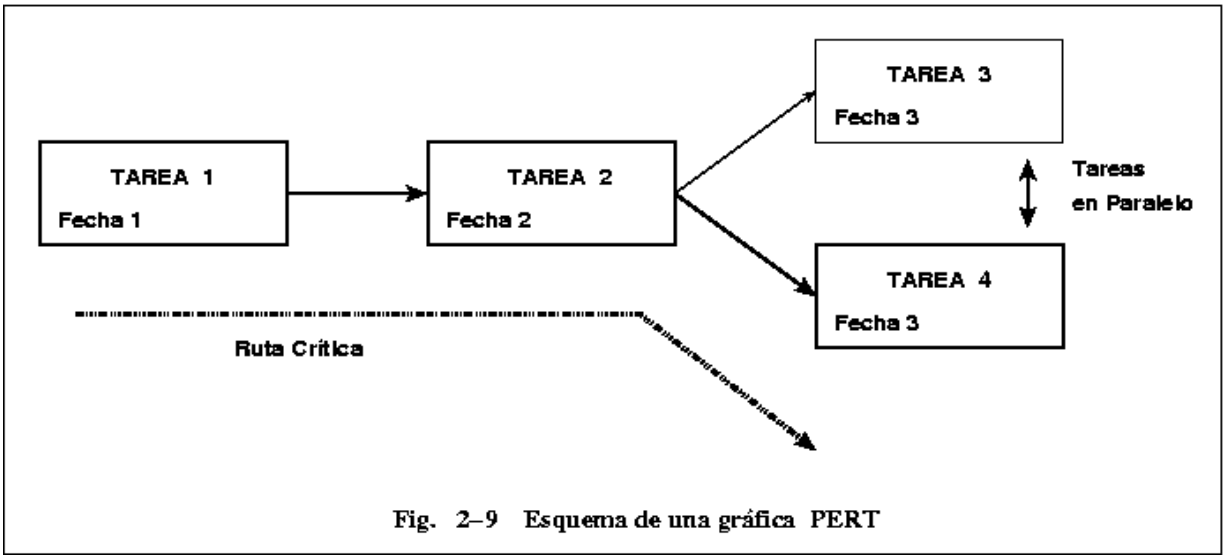


Fig. 2–9 Esquema de una gráfica PERT

Las gráficas Gantt y PERT son una *estimación* de lo que representan y las personas encargadas deben tener datos de proyectos anteriores, experiencias pasadas y visión para ver a futuro, ya que conllevan muchos factores de riesgos que pueden o no ser vistos.

También deben tener la capacidad para poder modificarlos, contar con márgenes de error y adaptarlos a algunos agentes externos.

[6]Bruegge, Bernd. Dutoit, Allen H. *Ingenieria del Software orientado a Objetos.* Pearson Educación. México. 2002. Pág. 426

[7]ídem

2.6 Planificación temporal

La planificación en el proyecto de software juega un papel muy importante, ya que si no se tiene en consideración, es factible salirse del tiempo estimado y ocasionar grandes pérdidas.

El principal objetivo de la planificación es dar al ingeniero de software un marco de trabajo sobre los recursos, riesgos y costos del proyecto, en donde se pueda basar para la toma de decisiones en cualquier nivel del desarrollo del sistema. El marco está sujeto a un tiempo determinado que sea lógico y factible para realizarlo, además que es recomendable que se actualice conforme al desarrollo del sistema.

Entre las razones más comunes por las cuales un proyecto es *atrasado* se identifican: que la fecha de entrega es poco realista, cambios de los requisitos del software por parte del cliente, la subestimación del proyecto, no estudiar los errores predecibles y los no predecibles, algunas dificultades técnicas o humanas que no se toman en cuenta y sobre todo la falta de comunicación.

La Tabla 2-1 es una recopilación de los riesgos más comunes, sus probabilidades y los efectos en el sistema.

Tabla 2-1 Los riesgos más comunes.

Riesgo	***Probabilidad***	***Efectos***
financieros de la organización hacen que se reduzca el Problemas presupuesto del proyecto.	Baja	Catastróficos
Es imposible reclutar personal con las habilidades requeridas para el proyecto.	Alta	Catastróficos
El personal clave está enfermo en un momento crítico para el proyecto.	Moderada	Serios
Componentes de software que se deberían reutilizar tienen defectos que limitan su funcionalidad.	Moderada	Serios
Cambios que requieren trabajo de rediseño significante son propuestos.	Moderada	Serios
La organización es reorganizada de tal manera que una administración diferente es ahora responsable del proyecto.	Alta	Serios
Las bases de datos usadas en el sistema no pueden procesar la cantidad de transacciones por segundo como se esperaba.	Moderada	Serios
El tiempo requerido para desarrollar el software es subestimado.	Alta	Serios
Las herramientas CASE no pueden ser integradas.	Alta	Tolerables
Los clientes no logran entender el impacto de los cambios de requerimientos.	Moderada	Tolerables
Entrenamiento requerido para el personal no está disponible.	Moderada	Tolerables
La velocidad de reparación de defectos es subestimada.	Moderada	Tolerables
El tamaño del software es subestimado.	Alta	Tolerables
El código generado por las herramientas CASE es ineficiente.	Moderada	Insignificantes

El análisis del riesgo juega un papel muy importante, ya que la mayoría de las estimaciones tienen que basarse en las probabilidades de que ocurra un error.

Pressman establece los principios básicos de la planificación temporal en los siguientes puntos:

Compartimentación, es la división del proyecto en secciones de actividades y tareas programables para se tenga un mejor control del proyecto.

Interdependencia, cada una de las actividades debe tener una cierta independencia con respecto a sus similares, deben trabajar como complementarias o paralelas.

Asignación de tiempo, se debe asignar un cierto número de actividades en un tiempo delimitado a un grupo específico, donde ambos deben ser planeados y delimitados.

Validación de esfuerzo, éste lleva una correlación directa con el anterior, ya que las actividades y el tiempo asignado al grupo de trabajo debe ser sin sobre valorar ni minimizar el esfuerzo del equipo en general.

Responsabilidades definidas, en cada equipo debe existir un responsable del proyecto y del personal.

Resultados definidos, cada una de las partes debe tener una visión exacta de lo que realiza su sección, por lo tanto, el resultado debe estar sumamente especificado.

Hitos definidos, todas las tareas deberán estar asociadas a un hito general del proyecto.

2.7 Diseño del software

La importancia del diseño radica en que es aquí donde se fomenta la *calidad*, (opciones útiles y necesarias para tener un software adecuado a las necesidades del grupo o personas dueñas de éste), ya que es el diseño quien proporciona la representación del software en la que se puede valorar la calidad, es una forma de entender veraz y explícitamente cada una de las necesidades del cliente y, es el fundamento para las siguientes fases del desarrollo y mantenimiento del sistema. Por lo tanto, el diseño previene muchos riesgos de calidad en todo el proceso.

La ingeniería del software es relativamente nueva en el desarrollo de teorías y métodos, pero existen técnicas para cualificar un diseño y aplicar sus notaciones. Recordemos que principalmente se estipulan los cuatro pasos siguientes:

Análisis ⟶ Diseño ⟶ Codificación ⟶ Prueba

Y dentro de cualquier método de diseño se producen las siguientes etapas:

a) Diseño de datos. Transforma el modelo de información creado en el análisis en una estructura de datos que se va a requerir para la realización del software, tomando como base principal todas las estructuras de los diagramas entidad–relación y el contenido del diccionario de datos.
b) Diseño arquitectónico. Define la relación entre los principales elementos estructurales del programa. Se obtienen del modelo de análisis y de la interacción de subsistemas definidos en ese modelo, como los diagramas de estado.
c) Diseño procedimental. Transforma los elementos estructurales de la arquitectura del programa en una descripción procedimental de los componentes del software. Se toma como base la especificación del proceso, la especificación del control y los diagramas de transición de datos.
d) Diseño de interfaz. Es la comunicación del sistema con diferentes entidades, pueden ser externas o internas, llámese operadores y usuarios, o un manejo de variables y datos. Como es un flujo de información, entonces basta con la información bien estructurada de los diagramas de flujos de datos.

Cada uno de los procesos de diseño son sistemas interactivos que traducen los requisitos en una representación del software. El primer diseño se representa con un alto nivel de abstracción y conforme avanza se van detallando más las especificaciones, los datos y los requerimientos, por lo tanto, el refinamiento subsiguiente siempre lleva a un menor nivel de abstracción hasta poder llegar a los requisitos específicos del sistema.

Las directrices para la evaluación de un buen sistema son las siguientes:

- Implementar todos los requerimientos explícitos en el modelo de análisis, tomando en cuenta sus requerimientos muy particulares.
- El diseño debe ser claro para cualquier persona que esté involucrada en el proyecto.
- El diseño debe proporcionar una idea contundente de lo que es el software desde la perspectiva de la implementación.

Algunos principios básicos que se marcan para realizar un buen diseño son:

1. El diseñador debe considerar enfoques alternativos, juzgando a cada uno en base a los requisitos del problema, los recursos disponibles y los criterios de calidad alterna.
2. Se debe continuar todos los pasos del modelo de análisis para comprobar que los requisitos han sido tomados en cuenta en este modelo.
3. No se debe gastar tiempo en algo que ya esté inventado. Por el contrario, todos los recursos gastados deberán ser utilizados para crear algo novedoso.
4. Todo el equipo debe trabajar con formatos y estilos preestablecidos para una mejor organización dentro de éste.
5. Todo sistema debe proveer cambios y modificaciones en su ciclo de vida.
6. Un programa no puede fallar ante cualquier circunstancia de uso, esto es, que no debe “caerse” cuando esté en uso.

7. Hay que mantener bien delimitado la diferencia entre diseñar y escribir el código. Un diseño en su fase menos abstracta o en su último nivel no debe ser el código escrito.
8. Hay que revisar los elementos conceptuales del proyecto, tratar de fijar la vista primero en la estructura general y conforme avanza el proceso ir detallándola mejor.

Si se tiene una correcta aplicación de los principios anteriores, se generan lo que se conoce como los factores de calidad internos y externos.

Los *factores de calidad externos* son aquellos que son vistos por el usuario, como lo son la velocidad, la calidad, la utilidad, las interfaces, etc.

Los *factores de calidad internos* permiten un diseño de alta calidad desde la perspectiva técnica, esto es lo que toma importancia para los ingenieros del software.

Los conceptos fundamentales en el diseño de la programación incluyen siete aspectos para su desarrollo, que se explican a continuación:

1. Abstracción

El problema siempre debe dividirse en diferentes niveles de abstracción, el primero establece una solución a un nivel del lenguaje natural y conforme pasan los niveles se toma un lenguaje más procedimental, ya que en el último nivel se establece la solución de manera que pueda implementarse directamente.

2. Refinamiento

El refinamiento sucesivo es la base del diseño descendente. El refinamiento es un proceso de elaboración que comienza en la declaración de la función definida a un nivel superior de abstracción hasta que las instrucciones se expresan en términos simples de un lenguaje de programación. El ingeniero hace una ampliación de su declaración original, dando cada vez más detalles para que puedan ser descompuestos o mejor estructurados.

3. Modularidad

Se basa en dividir el software en componentes bien definidos que pueden ser tratados por separado denominados módulos y que están integrados para satisfacer los requerimientos del programa.

Es más entendible cuando se observan los procesos por partes. La frase “divide y vencerás” es un término que puede ejemplificarlo de una manera muy eficaz. Pero hay que tener cuidado, ya que tan malo es caer en la excesiva modularización como en una modularización muy escasa, ya que a mayor número de módulos, los costos disminuyen pero el esfuerzo de integración es mucho mayor y, a menor número de módulos el costo de integración es casi nulo pero los costos de cada uno se incrementan de manera considerable.

El saber tomar un número adecuado de módulos es una decisión que requiere de muchos procedimientos, pero se han tomado en general los siguientes criterios:

A. Capacidad de descomposición modular. Tomar en cuenta la división total del proyecto, que se refiere a proporcionar mecanismos de descomposición del problema en subproblemas.
B. Capacidad de empleo de componentes modulares. Esto es, tiene que tomar objetos ya existentes o generar objetos reutilizables.
C. Capacidad de comprensión modular. Si se puede manejar un módulo como una unidad por sí sola, será más fácil de construir y de cambiar.
D. Continuidad modular. Es otra ventaja del punto anterior, si existen cambios en algún módulo, el efecto secundario ante los demás, no ocasionará efectos drásticos.

Un sistema puede diseñarse modularmente, pero su implementación debe ser *monolítica.*

4. Jerarquía del software

Aquí se representa la estructura global del software en niveles y la forma en que proporciona integridad conceptual al sistema. La forma común es la estructura jerárquica, ya que se puede visualizar la interacción entre cada uno de los componentes principales del sistema.

Existen propiedades que pueden formar parte del modelado de la arquitectura, por ejemplo,

A) Propiedades estructurales, definen los componentes que están presentes en el sistema, así como las interacciones entre éstos.
B) Propiedades extra–funcionales, se enfocan a observar cómo se consiguen los requisitos de rendimiento, calidad, fiabilidad, etc.
C) Familias de sistemas relacionados, el diseño arquitectónico debe dibujarse similar a otras familias, para que pueda ser reutilizable por algún otro sistema o utilizar partes de otro.

Tomando en cuenta lo anterior, el diseño puede representarse de uno o más *modelos*, pero la mayoría se clasifica en alguno de los puntos siguientes:

- *Modelos estructurales*, representan al sistema como una organización de componentes. El modelo toma un proceso con una mayor abstracción y se observa cada vez más general el sistema, principalmente se realizan para poder identificar patrones similares u objetivos no satisfechos.
- *Modelos dinámicos*, observan cómo se comporta el sistema en función de los acontecimientos externos.
- *Modelos de proceso*, se refieren a los procesos técnicos que debe tener el sistema.
- *Modelos funcionales*, representan la jerarquía del sistema.

5. Jerarquía de control

Representa la organización jerárquica de los módulos del programa, usualmente se utiliza un

diagrama de árbol, pero también son muy usados los diagramas de Warnier y Jackson.

Con la jerarquía de control también se representan dos características importantes para el desarrollo en general: la visibilidad y la conectividad.

La *visibilidad,* indica los componentes que pueden usar datos no propios o los que se pueden invocar, ya sea directa o indirectamente, además que muestra quién los invoca.

La *conectividad,* indica el conjunto de componentes que son invocados directamente o que usan datos de otro componente determinado. Son módulos directos que manejan datos indispensables para uno o dos, por ejemplo el módulo de una ejecución con el módulo de entrada de sus variables.

a) Diagramas de árbol

Una de las formas para representar la jerarquía de control, es una estructura arborescente, un caso particular es un modelo en red pero con restricciones bien marcadas, por ejemplo, ha de tener una estructura ordenada de cada una de las entidades que la conforman y reglas bien establecidas.

El diagrama de árbol está compuesto por nodos que representan entidades, enlazados por arcos que representan asociaciones o relaciones entre dichas entidades. Entre sus características se encuentran las siguientes:

» El nodo *raíz* es el más alto de la jerarquía y corresponde al nivel 0.
» Los arcos representan asociaciones entre los nodos y sólo puede existir uno por cada dos.
» Un padre (nodo con el nivel superior inmediato respecto a otro) puede tener uno o varios hijos; no así los hijos, que sólo pueden tener un padre.
» Todo nodo excepto el raíz tiene un padre.
» Las hojas, son los nodos que no tienen hijos, esto es, las últimas estructuras de la jerarquía.
» Altura es el total del número de niveles.
» Momento es el número de nodos.
» Cada nodo no terminal y sus descendientes forman un subárbol, de modo que la estructura de árbol es una estructura recursiva.

La Fig. 2-10 muestra la estructura y las partes del diagrama de árbol.

b) Diagramas Jackson

El desarrollo de los sistemas Jackson evoluciona fuera del trabajo realizado sobre el análisis de dominio de la información; la relación con el diseño de programas y sistemas se centra en los modelos del dominio de información del mundo real, "...el desarrollador comienza creando un modelo de la realidad con el que se relaciona el sistema, la realidad que proporciona tema, el tema del sistema".[8]

[8] Senn, James, A. *Análisis y diseño de sistemas de información.* McGraw Hill. México. 1992.

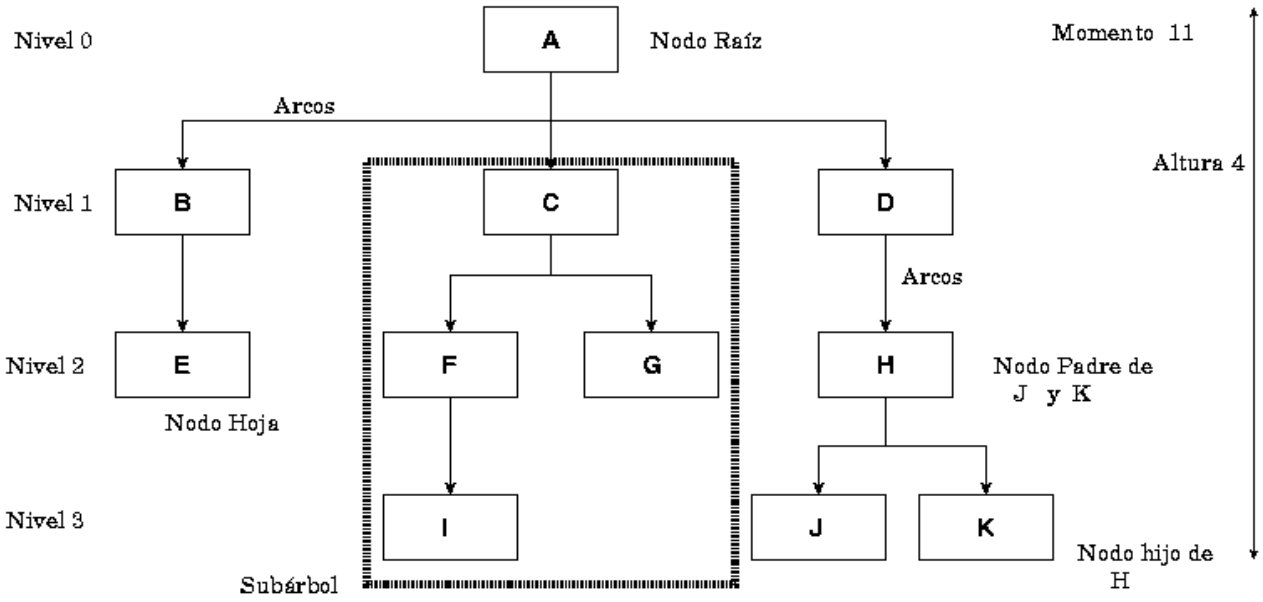


Fig. 2–10 Estructura de un Modelo Árbol

Richard Fairley los define de la siguiente manera: "La programación estructurada fue desarrollada por Michael Jackson como una técnica sistemática para hacer un mapeo de la estructura de un problema en una estructura de programa para resolver el problema"[9], y estructura tres pasos:

i. El problema se modela mediante la especificación de las estructuras de datos de entrada y de salida usando diagramas de árbol estructurados.
ii. El modelo de entrada-salida se convierte en un modelo estructural para el programa, identificando los puntos de correspondencia entre nodos en los árboles de entrada y salida.
iii. El modelo estructural del programa se expande en un modelo de diseño detallado que contiene las operaciones necesarias para resolver el problema.

6. Partición estructural

La estructura del diagrama debe ser analizado por diversos ángulos para poder observar diferentes tipos de problemas, por lo tanto, las particiones se estudian *horizontal* y *verticalmente*, ya que cada una da puntos de vista distintos.

[9]Fairley, Richard E. *Software Enginnering Concepts*. McGraw Hill. México. 1987.

Cuando se observa *horizontalmente* se pueden definir tres particiones: entrada, procesamiento y salida, al ver cada una de las funciones por separado se tiene una mejor idea de las modificaciones que tiene los datos, de las salidas y como son influenciadas por sus antecesores. Pero tiene por desventaja que puede permitir el paso de más variables que las necesarias y no percatarse de esto.

La *partición vertical* toma una estructura en la que se pueden observar claramente las jerarquías del sistema y se pueden ver los módulos para descentralizar los datos y operaciones, con el fin de que los módulos de nivel superior realicen funciones de control y poco procesamiento y los módulos que residen en la baja arquitectura deben realizar las tareas de entrada, salida y procesamiento de datos.

Además, se pueden visualizar las consecuencias de cambios, hechos en diferentes niveles; esto es, un cambio realizado en los últimos nodos no tendrá un peso tan excesivo como un cambio en los primeros módulos y, por lo tanto, se podrá tener un mejor mantenimiento y aumentar el nivel de calidad.

7. Estructura de datos

La estructura de datos es una representación gráfica, lógica y estructurada entre los elementos individuales del sistema. La estructura de datos dicta las alternativas de organización, métodos de acceso, capacidad de asociación y procesamiento de la información y, mayormente se basa en el ingenio del diseñador. Pero existen estructuras básicas bien establecidas que lo conforman, enunciando de los más sencillos a los más estructurados, por ejemplo, el elemento escalar, el vector secuencial, el espacio *n-dimensional* y la matriz *n-dimensiones;* en este nivel y tomando las estructuras básicas descritas anteriormente se cuenta la lista enlazada y la estructura de datos jerárquica que implementa listas multienlazadas que contienen elementos escalares, vectores y muy probablemente espacios *n-dimensionales*.

8. Diseño modular

El diseño modular se ha convertido en un enfoque aceptado por muchos ingenieros de proyectos, ya que otorga al sistema una disminución de la complejidad, facilidad en los cambios, una mejor implementación, formas de estructuración, etc., por lo que forma parte de los conceptos fundamentales del software.

- Independencia funcional

Al tener un sistema estructurado en módulos se requiere que cumplan con ciertas características y una de ellas es la independencia entre los módulos. La *independencia* busca que cada módulo lleve una subrutina específica, esto es, una parte del trabajo, que se encuentre relacionado en lo más mínimo con otras de su misma especie y que además tenga un fácil acoplamiento con éstas; esto es importante ya que si no se cumple, todos los objetivos de la modularidad se vendrían abajo y que el sistema se consideraría como un objeto único.

La independencia se rige por dos criterios cualitativos: la *cohesión* y el *acoplamiento*.

A) Cohesión

La cohesión es una medida para observar qué tan bien un componente encaja junto a los demás, se ejemplifica cuando un módulo realiza una sola tarea dentro de un procedimiento del software y requiere de una nula o poca interacción con otros sistemas que se desarrollan en otras partes de programa, en otras palabras, la cohesión implica que todos los elementos de una clase deben hacer cosas indirectamente relacionadas entre sí.

Se encuentran los siguientes tipos de cohesión de acuerdo a las actividades del sistema:

- *Coincidentalmente cohesivo,* son los módulos que realizan un conjunto de tareas poco relacionadas las unas con las otras.
- *Cohesión lógica,* un módulo en la cual sus elementos manifiestan relaciones en torno a tareas de una categoría general, esto es, realizan tareas que se relacionan sólo por variables, por ejemplo, los módulos que producen una salida independientemente del tipo de dato.
- *Cohesión temporal,* son aquellos módulos que están relacionados sólo en el momento de correr el programa, por un determinado intervalo de tiempo o que deben correrse paralelamente.
- *Cohesión procedimental,* se definen como: "... los niveles moderados de cohesión están relativamente cerca unos de otros en la escala de independencia modular. Cuando los elementos de procesamiento de un módulo están relacionados y deben ejecutarse en un orden específico, existe *cohesión procedimental*... "[10].
- *Cohesión relativa a datos,* es cuando los elementos de procesamiento se concentran en un área de la estructura de datos, también llamada *cohesión de comunicación.*[11]
- *Cohesión secuencial,* se observa como la salida de una actividad y la entrada a la siguiente.
- *Cohesión funcional,* es una actividad simple y específica.

Pero no se puede definir literalmente un nivel de cohesión, ya que se define como un módulo que realiza una sola tarea y simplemente se busca que tenga un nivel alto para que pueda establecer una mayor independencia funcional.

B) Acoplamiento

Es la medida de interconexión entre los módulos de la estructura de un programa y dependen de su complejidad, de los puntos de entradas, referencias y flujos de datos.

[10]Manso Martínez, Ma. Esperanza. *Parte I: Programación Modular* U. de Valladolid. Valladolid. Pág. 5.
[11]Pressman, Roger S. *Ingeniería del Software: un enfoque práctico.* McGraw Hill. España. 1998. Pág. 240.

Los tipos de acoplamiento más comunes son:

- Acoplamiento por contenido, se presenta cuando un módulo modifica los valores y las instrucciones de otro; suele presentarse también cuando un modulo se pasa a otro por puntos distintos a los de la entrada.
- Acoplamiento por zonas compartidas, aquí los módulos comparten zonas llamadas globales, con lo cuál, éstos módulos están disponibles entre ellos.
- Acoplamiento por zonas datos, similar al anterior, solamente que existen ciertas selecciones entre ellos.
- Acoplamiento por datos, se incluyen listas de parámetros para pasar a los elementos entre rutinas.
- Acoplamiento de control, un módulo puede controlar la secuencia de los procesos con la ayuda de banderas.

Mientras menos sea el número de conexiones y más simples (minimizar el acoplamiento), entonces se realiza un mejor diseño.

La conexión simple entre módulos dará como resultado un software fácil de comprender y menos propenso a efectos en cadena que es cuando los errores ocurren en un lugar y se propagan a través del sistema. Un bajo acoplamiento va a ser una indicación de un programa bien particionado.

2.8 Antecedentes de las bases de datos

El procesamiento de la información juega un papel muy importante en cualquier ámbito de la sociedad, como en el área educativa, gubernamental, pública, privada y en toda aquella que maneje datos y variables en sus actividades.

La necesidad de independencia ha llevado al hombre a crear herramientas que le permitan localizar, seleccionar, coleccionar y analizar la información que va requiriendo, para lo cual se empieza a utilizar la tecnología, como las computadoras, las telecomunicaciones y demás instrumentos que han revolucionado la vida del hombre en todos sus aspectos.

A principios de la década de los sesentas, muchas empresas comenzaron a computarizar sus sistemas de información y todos los datos se guardaron en medios electrónicos. Se dejó de utilizar grandes archivos de papel y se empezaron a usar lenguajes de alto nivel para recuperar y manejar los datos en los registros de almacenamiento.

En 1970 el artículo de E. F. Codd sobre el modelo de datos relacional dio un fuerte impulso a la industria, W. Hansen escribe: “El enfoque de Codd proponía el acceso y la manipulación de los

datos únicamente desde el punto de vista de sus características lógicas"[12]. A partir de esta fecha (principalmente en el extranjero), los sistemas para las bases de datos relacionales tuvieron su mayor apogeo.

A partir de la década de los ochenta el manejo de la bases de datos en México deja de ser un trabajo inalcanzable para los grandes centros de investigación, ya que desde entonces se conocían las ventajas de guardar y manejar la información en las bases de datos. Las bibliotecas se hicieron partícipes de las ventajas que ofrece la computación y comienzan a generar infinidad de bases de datos referenciales, que cumplen con un alto grado de homogeneidad por los criterios y normas internacionales que se debe cumplir para la clasificación de cualquier tipo de documento.

La exigencia de los usuarios dieron a los especialistas de computación la visión de que no bastaba el tener computadoras y telecomunicaciones, sino que se necesitaba una interfaz usuario–máquina y programas que permitieran el manejo de la información de una manera más fácil, rápida y confiable. Fue para solucionar el problema que se crearon los manejadores de bases de datos como: Dbase III Plus, Dbase IV y Micro-Isis, este último fue creado por la UNESCO con el propósito de manejar e intercambiar la información a grandes escalas.

Actualmente, la importancia que ha tenido el manejo de las bases de datos a obligado a que se le dedique un tiempo considerable en la estructuración de la empresa, buscando una gestión más racional de la información en todo su conjunto, generar más técnicas de organización, recomendaciones, fidelidad de la información almacenada y hasta leyes para el uso adecuado de los sistemas.

2.9 Bases de datos

En un comienzo se hablaba de archivos y conjuntos de datos, ahora de grandes bases; algunos autores afirman que las bases de datos son una recopilación de los grandes bancos de datos, pero para definirlo correctamente se muestran a continuación algunos párrafos:

"Colección de datos correspondientes a las diferentes perspectivas de un sistema de información (de una empresa o institución), existentes en algún soporte de tipo físico (normalmente de acceso directo), agrupados en una organización integrada y centralizada en la que figuran no sólo los datos en sí, sino también las relaciones existentes entre ellos, y de forma que se minimiza la redundancia y se maximiza la independencia de los datos de las aplicaciones que los requieren."[13]

"Una base de datos es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas y, su definición y

[12] Hansen, Gary W., Hansen , James V. *Diseño y Administración de Bases de Datos.* Prentice Hall. España. 1997.
[13] Guilera Aguera, Llorenc. *Introducción a la informática*. Edunsa. Barcelona. 1988. Pág. 377.

descripción han de ser únicas estando almacenadas junto a los mismos. Por último, los tratamientos que sufran estos datos tendrán que conservar la integridad y seguridad de éstos."[14]

"Una base de datos es una colección de archivos interrelacionados creados con un DBMS. El contenido de una base se obtiene combinando datos de todas las diferentes fuentes en una organización, de tal manera que los datos estén disponibles para todos los usuarios, y los datos redundantes puedan eliminarse, o al menos minimizarse."[15]

"Se entiende como un archivo de datos interrelacionados, recolectados, que satisfacen las necesidades de información de una comunidad determinada de usuarios. Cada unidad de información almacenada en una base de datos está compuesta por datos elementales, cada uno de los cuales representa características particulares de la entidad que se describe. Por ejemplo, una base de datos bibliográficos contendrá información sobre libros, reportes, artículos de revista, etcétera."[16]

"Una base de datos está compuesta de varios archivos, además que provee facilidades para capturar las relaciones que existen entre los registros. Por lo tanto un sistema de base de datos trasciende a un sistema de archivos por proveer el diseño con facilidades para representar las relaciones entre los registros."[17]

Lo que se puede concluir es, que las bases de datos son un conjunto de información relacionada no redundante, que está organizada, sistematizada y debe encauzarse en un propósito específico en beneficio de una comunidad. Tiene que cumplir con los objetivos de independencia entre las aplicaciones y las estructuras de datos, integridad en los datos y la seguridad de éstos ante los múltiples usuarios que la utilicen.

2.10 Características de las bases de datos

En las bases de datos por manejar grandes cantidades de información y porque cada sistema se estructura para un objetivo principal y específico, los objetivos secundarios pueden llegar a tener modificaciones, lo que conlleva a que la base se vea de diferentes puntos de vista.

Las bases de datos tienen algunos puntos importantes para el manejo de la calidad, como la *no redundancia, representación de la información, excelentes respaldos y obtención de los registros, tiempo de respuesta, integridad de los datos, independencia y seguridad de la*

[14]Mota Herranz, Laura. *Bases de datos relacionales: teoría y diseño*. Universidad Politécnica de Valencia. Valencia. 1994. Pág. 9.
[15]Alice Y., H. Tsai. *Sistema de base de datos: Administración y uso.* Prentice Hall Hisponoamericana S.A. 1990. Canada. Pág. 5.
[16]Gil Rivera, Ma. del Carmen. *Las Bases de datos, importancia y aplicación en la educación.*
[17]Marty, Tim. Hartley, Tim. *DB2/SQL A professional Progammer's Guide*. McGraw Hill Book Company. 1989. New York. Pág. XXII.

información y el intercambio de datos, que detallaremos a continuación.

La **redundancia** lógica o física es información que se encuentra en dos o más registros en un mismo sistema de base de datos.

La redundancia lógica debe ser controlada de forma que no exista duplicidades en los datos, no así en las redundancias físicas, que serán necesarias en algunos casos a fin de responder a objetivos básicos del sistema, esto resume según De Miguel: "... en las bases de datos no debe de existir redundancia lógica, aunque sí se admite cierta redundancia física por motivos de eficiencia."

La **independencia** es un factor importante en las bases de datos, tanto física como lógica entre los registros y los tratamientos. Es una de las características básicas en el diseño de las bases de datos y la que marca la diferencia con los ficheros. El manejar datos, modificarlos y acceder a ellos aumenta su complejidad si se habla de una gran cantidad de usuarios, la reestructuración de la base hace que la independencia entre los datos, los programas y las aplicaciones que los manejen tengan una gran importancia. No resulta viable que al realizar cualquier modificación en el programa se cambie el valor de algún registro.

La **representación** de la información debe ser adecuada para el número de usuarios y las características de éstos, ya que si la interacción con el usuario no es bien clara, no se podrán obtener todos los beneficios que requieren y podría causar un conflicto con el ingeniero por un material que realmente no trabaja a su máxima eficacia.

La **obtención** de cualquier registro debe cumplir con los siguientes lineamientos:

a) Conseguir los registros fácilmente y su disponibilidad en cualquier momento.
b) Independencia ante el programa de aplicación en uso.
c) **Tiempo de respuesta** adecuado, sin importar la cantidad de usuarios, esto es, debe tener la capacidad suficiente de manejar un número concreto de máquina cliente y el flujo de transacciones que manejen.
d) **Permisos**, si el sistema lo requiere se adapta a los diferentes niveles entre los usuarios. Entre los mismos datos puede haber diversas restricciones por diferentes usuarios.
e) **Claridad** en los datos, los datos deben estar claros y sin ambigüedades.

La **integridad** de los datos son las medidas tomadas para que la información guardada no sea modificada, por un usuario o por el mismo sistema debido a algún problema en el programa de aplicación. La veracidad de los datos es sumamente importante junto con **la pérdida de la información**, principalmente en este último se necesitan realizar respaldos totales y fraccionados de acuerdo a la importancia de los datos, en un tiempo fijo si es que los registros en la base son modificados continuamente.

La integridad tiene una mayor importancia en una base de datos que en un sistema de archivos privados, "... la integridad de datos es más importante en una base de datos que en un sistema de archivos privados, precisamente porque el primero se comparte y porque sin procedimientos de

validación adecuados, es posible que un programa con errores genere datos incorrectos, que afecten a otros programas al utilizar la información."[18]

Los **datos compartidos** se refiere a que varias aplicaciones pueden compartir los mismos datos de una misma base, sin generar errores de traslape cuidando la integridad de los datos y dentro de un tiempo considerable. Independientemente de que se vayan generando nuevos programas de aplicación.

La **flexibilidad** en una base, es que a los datos se les pueda acceder fácilmente, entrar por cualquier programa de aplicación o por cualquier función de relación entre sus tablas.

2.11 Arquitectura de las bases de datos

La arquitectura de una base de datos se reconoce por el grupo ANSI/SPARC en 1978 donde propone para cualquier tipo de bases de datos una arquitectura de tres niveles definidos: externo, conceptual e interno.

El ***nivel externo*** consiste en la vista que tiene el usuario sobre las bases de datos, es el que tiene mayor abstracción, la primera que tienen los usuarios de los elementos de datos y de las relaciones orientadas al mismo. En cierto modo es la parte del modelo conceptual a la que tienen acceso.

Existen diferentes tipos de usuarios que están dentro del nivel externo y son:

a) *Programadores de aplicación*, son los que interaccionan con el sistema a nivel datos, es el lenguaje principal escrito, puede ser C o Pascal, éstos son llamados Data Definition Language (DDL), donde proveen los medios necesarios para definir los datos con precisión, especificando las distintas estructuras puesto que la sintaxis de estos programas por lo regular son diferentes a la del lenguaje principal, las llamadas por lo regular van precedidas de caracteres especiales, de tal forma que se pueda generar el código apropiado.

b) *Usuarios sofisticados*, son los que interaccionan con el sistema sin escribir programas, esto es, estos usuarios escriben sus preguntas en un lenguaje de consultas. Cada consulta se somete a un procesador, cuya función es tomar una sentencia en DDL y descomponerla en instrucciones que entienda el manejador de las bases de datos.

c) *Usuarios especializados*, éstos pueden desarrollar aplicaciones especializadas de las bases de datos. Algunos usuarios desarrollan aplicaciones que no se contemplan en un marco tradicional para el procesamiento de datos, como puede ser el diseño asistido por computadora o aplicaciones en redes neuronales.

[18]Date, C.J. *Bases de datos: Una guía práctica.* Addison-Wesley Iberoamericana. Argentina. 1987. Pág. 32.

d) *Usuarios comunes*, son todos los usuarios que interaccionan con el sistema haciendo uso sólo de los programas de aplicación.

El ***nivel conceptual***, tiene un nivel medio de abstracción, es una representación de los datos de las opiniones parciales de los usuarios.

Aquí se incluye la definición de datos y las relaciones entre ellos, es el resultado del esquema conceptual, sencillo y lógico de la descripción de todos los elementos y las relaciones de los datos.

El esquema se formula independientemente del almacenamiento físico de los registros, por lo que la base aparece solamente como una colección de registros lógicos, sin descriptores de almacenamiento. La transformación de registros conceptuales en registros físicos para su almacenamiento se lleva a cabo por el sistema y es enteramente transparente para el usuario.

Aquí se encuentran varios esquemas que delimitan esta etapa, como es el diccionario de datos, el lenguaje de manipulación de datos, entre otros.

1. El *diccionario de datos* es un archivo que contiene entidades y atributos de las bases de datos y las características que las definen, o sea, tiene datos sobre los datos.

Los datos van a formar parte de las normas y especificaciones del proyecto. Nos ayudan a tener una mayor información sobre las entidades y los atributos para las bases de datos.

La parte medular de los diccionarios de datos, la constituye la definición y descripción de las entidades, donde se consideran de manera integral aquellas características que permiten conceptualizar los registros en unidades discretas. Cada entidad tiene un nombre, su definición y sus atributos.

Las ventajas al utilizar un diccionario de datos es porque facilita el control de cada una de las entidades y atributos que forman la estructura de las bases de datos, además que se pueden controlar de manera dinámica la estructura de la interfaz de usuario para las diferentes pantallas del sistema.

Es muy importante tener acceso a éste antes de realizar cualquier modificación, ya que proporcionan un mejor manejo en el control de cada una de las entidades y de los atributos que forman parte de la estructura de bases de datos, además de que se tiene un mayor control dinámicamente.

2. El *lenguaje de manipulación de datos* es el lenguaje que capacita a los usuarios para acceder o manipular los datos. La *manipulación de los* datos se define como el encargado de facilitar a los usuarios el acceso y manejo de los datos.

Para manejar los registros se tienen dos tipos de usuarios: *procedimentales* y *no procedimentales*.

Los *procedimentales* son aquellos usuarios que especifican los datos que requieren y cómo se pueden conseguir y los *no procedimentales* son aquellos que sólo especifican los datos que requieren, se encargan de la recuperación de los datos almacenados, de la inserción y supresión de datos en las bases de datos y de la modificación de los existentes.

Se puede observar que estos últimos son los más fáciles de aprender pero pueden ser menos eficaces y más costosos con respecto a los primeros.

3. *Consulta* (buscar documentación o datos sobre algún asunto o materia), son las sentencias con las que se puede recuperar información, características y formas de uso.

Las consultas van de acuerdo al tipo de usuario, pueden sólo llevarse a cabo por medio de las interfaces o directamente al lenguaje de programación. Actualmente, también existen diferencias de acuerdo al tipo de sistema operativo que se esté manejando: Linux, Windows, Solaris, etc.

4. *Precompilador, compilador y procesador de archivos,* convierten los códigos para que puedan ser procesados por su sucesor.

Es una serie de pasos que la mayoría son ejecutados por el sistema y pueden cambiar de acuerdo al tipo de procesamiento que se esté llevando a cabo.

5. *Manejador de archivos,* está enfocado principalmente con el hardware del sistema, por ejemplo, asigna memoria del disco a las bases de datos y a los programas de aplicación. El ingeniero tendrá que adaptarse a las especificaciones del cliente, ya que tal vez, éste requiera algo que técnicamente no es muy factible.

En ocasiones puede encontrarse el nivel conceptual divido en dos niveles, conceptual y lógico. El primero de ellos corresponde a la visión del sistema global desde un punto de vista organizativo independiente, no informático y el segundo correspondería a la visión de las bases de datos expresada en términos del sistema que se va a implantar con medios informáticos.

El ***nivel interno***, es el nivel más bajo de abstracción y define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso. El ingeniero determina la arquitectura física, los dispositivos que contendrán los datos, los métodos de acceso y actualización, la forma de modificar registros, etc.

Los niveles en la arquitectura de las bases de datos, se pueden representar en la Fig. 2-11.

El modelo de arquitectura permite establecer el principio de independencia de los datos. Esta independencia puede ser lógica o física. Por independencia lógica se entiende que los cambios en el esquema lógico no deben afectar a los esquemas externos que no utilicen los datos modificados. Por independencia física se entiende que el esquema lógico no se vea afectado por cambios realizados en el esquema interno, correspondientes a modos de acceso, etc.

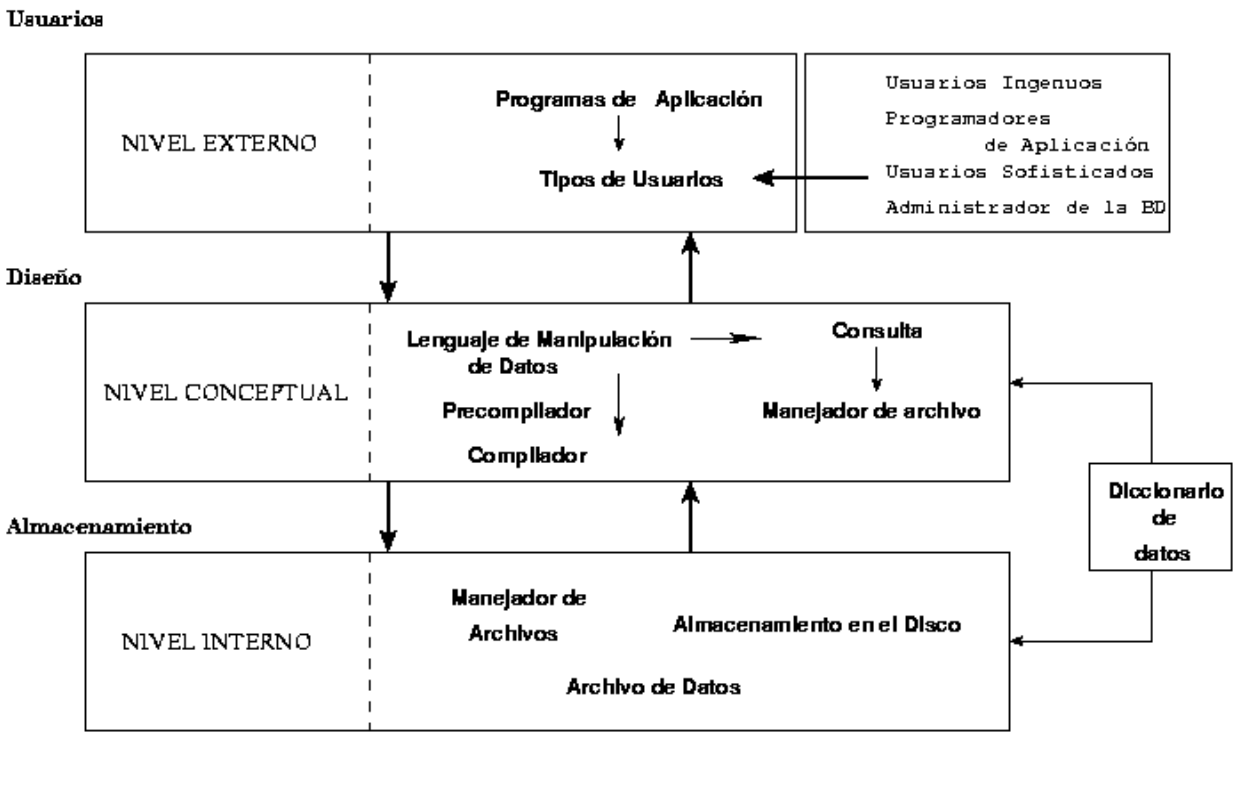


Fig. 2–11 Arquitectura de la Base de Datos

Los archivos múltiples en las bases de datos son muy comunes, ya que no existen o son minoría las empresas en la que sus sistemas de archivos sean pequeños. Al usar gran cantidad de archivos las relaciones entre éstos aumentan considerablemente y para que exista una base de datos el uso de los archivos debe estar integrado y coordinado. El mantenimiento del complejo conjunto de archivos y sus relaciones puede ser un gran problema y es por eso, que se comienzan a tener visiones múltiples de una estructura, definir relaciones, dominios, etc.

El principal método para el diseño de las bases de datos es la construcción de *modelos* que reflejen sólo la estructura de la base, de manera que permitan la manipulación de los bloques conceptuales de construcción, a este tipo de modelos se les denomina *modelos estructurales*.

Los modelos estructurales se basan en los *modelos relacionales* que tiene ya bien definidos sus archivos, relaciones únicas y las operaciones a las que ellas se le asignan.

El modelo relacional se ha ampliado al comenzar a manejar la definición de *conexión (*las relaciones entre los archivos), y donde existen tres tipos de conexiones para clasificar las relaciones en cinco tipos de características estructurales diferentes. Cuando se tiene ya toda la información para la base de datos entonces se tiene el *modelo estructural* o un *modelo de la base de datos*.

El modelo capta parte del significado o *semántica* de los datos (tómese como semántica "el estudio del significado de los signos lingüísticos y de sus combinaciones, desde un punto de vista sincrónico o diacrónico"[19]), en otras palabras, la semántica *afecta a la estructura de la base*, pero aquí se ignoran los detalles de estas distinciones semánticas.

Al ser la base de datos una estructura muy compleja que es utilizada por una gran cantidad de personas, surgen múltiples visiones de la *semántica* de la misma base de datos. Aún cuando es posible implantar las diferentes visiones en la base de datos utilizando múltiples arreglos de archivos, la resultante multiplicidad de datos y archivos duplicados hacen un gran problema cuando se realicen modificaciones o en el mantenimiento mismo del sistema. Para darle la mejor solución Wiederhold, afirma que se deben definir las bases exclusivamente desde un punto de vista y que después la misma base de datos podrá definirse desde otros puntos de vista.

Una sola visión se puede describir mediante un modelo, donde éste será un subconjunto de la realidad con características propias. Para conformar la base de datos se tienen que especificar muchos puntos de vista y por lo tanto, muchos modelos, pero estos modelos se construyen como se mencionó anteriormente de estructuras bien delimitadas, las expresiones básicas son los *elementos datos* y los *vínculos* entre ellos.

Existen los vínculos binarios, que son la mínima relación entre los registros, sus tres elementos son: el objeto, el atributo y el valor, como se muestra en la Fig. 2-12.

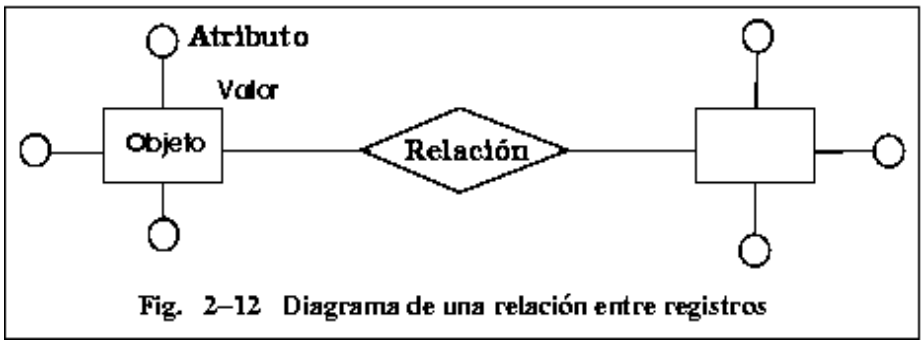


Fig. 2–12 Diagrama de una relación entre registros

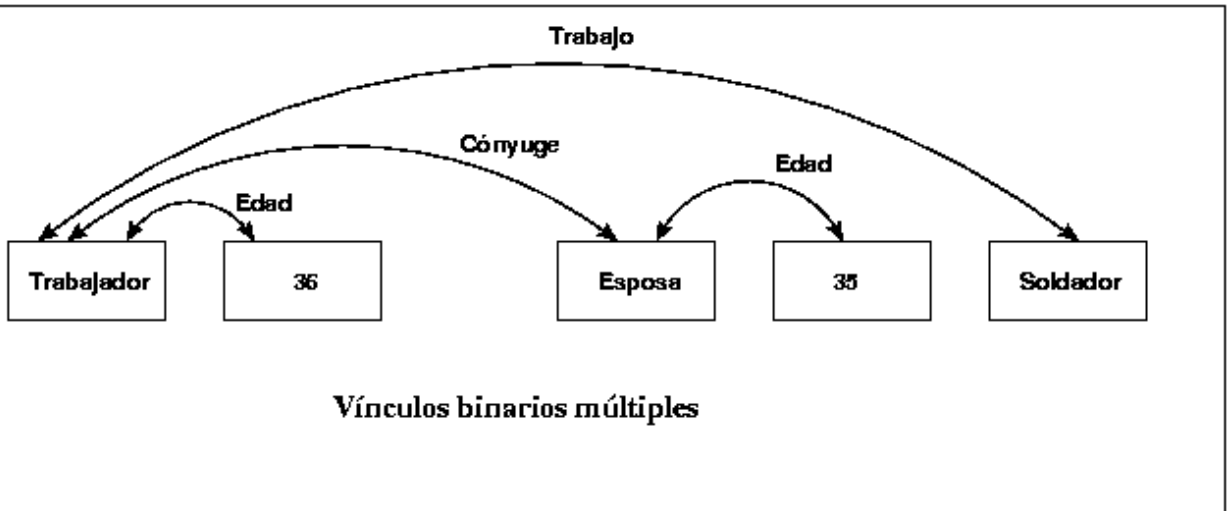
Al recolectar un mayor número de datos y establecer una organización entre éstos, surgen los *vínculos binarios múltiples*, que son una organización de *n* elementos dato relacionados en forma simple en una lista o en una *n-ada*.

La n-ada representa múltiples atributos de algún objeto, puede decirse que delimita un objeto específico y se define como "un conjunto de elementos dato atómicos relacionados y heterogéneos."[20]

Esta agrupación hace un mejor manejo de los datos, pero los datos originales se vuelven en gran

[19] Diccionario de la Real Academia Española. 2001.
[20] Wiederhold, Gio. *Diseño de Bases de Datos*. McGraw Hill. México. 1988. Pág. 415.

medida menos obvios. El esquema de un vínculo binario múltiple y de una n-ada se muestran en la Fig. 2-13.



Vínculos binarios múltiples

HOMBRE	EDAD	CÓNYUGE	EDAD CÓNYUGE	TRABAJO
Trabajador	36	Esposa	35	Soldador

n – ada

Fig. 2–13 N – ADA y Vínculos binarios múltiples

Cuando conjuntos de *n-adas* similares se ensamblan, entonces surgen las *relaciones*, "... una relación es un conjunto de n-adas homogéneas", se menciona también, "... dentro de una relación los elementos dato de una posición específica en cualquier n-ada pertenecen al mismo atributo."[21] Cada n-ada va a representar algún objeto diferente y expresar vínculos similares entre sus atributos.

Sobre las definiciones de atributos y sus relaciones, Wiederhold escribe, "cada vínculo en un registro se define mediante el nombre que sirve de etiqueta a las columnas de la n-ada: *el atributo.* Ahora una columna de una relación contiene elementos datos semejantes. Los vínculos específicos entre los atributos no se representan en la relación, sino que son explicados mediante una línea del encabezado de atributos, el *esquema-relación.* El orden de los atributos no es importante siempre y cuando la disposición de la n-ada coincida con el esquema relación. La estructura de la base de datos se definirá de manera que los vínculos originales puedan

[21] ídem.

recuperarse adecuadamente."[22]

Para definir el nombre de la etiqueta se pueden considerar una o algunas de las columnas de atributo, a estas columnas se les dan el nombre de *parte rectora* y las columnas de los atributos restantes se les denomina *parte dependiente*. Esta diferencia proporciona la esencia de los vínculos originales entre atributos y es vital cuando se manipulen los modelos.

También existen diferentes grados de acuerdo a la abstracción que se llega a manejar, los grados más comunes son hasta el nivel 4, ya que es demasiado tedioso estar haciendo grandes cantidades de diagramas para poder explicar definiciones que bien podrían describirse en niveles más bajos.

La Fig. 2-14 muestra varios tipos de relación y las partes que la componen.

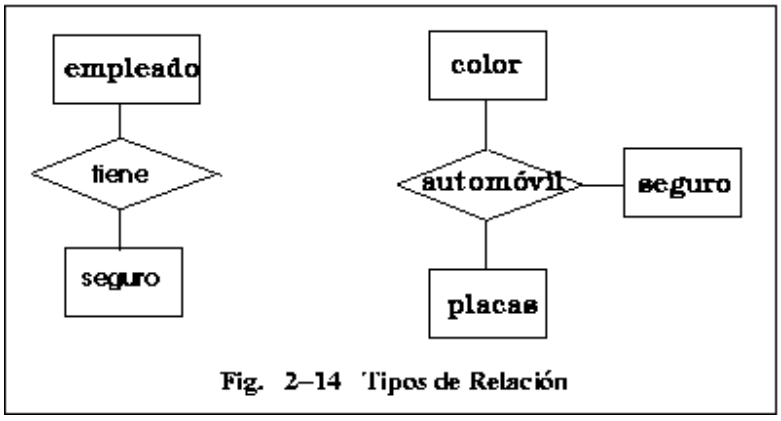


Fig. 2–14 Tipos de Relación

2.12 Sistema manejador de bases de datos (SMBD)

Para poder dar una organización a todos los niveles, es necesario una aplicación que interaccione con el usuario y los sistemas físicos de las bases de datos, esto es, una interfaz usuario–máquina, ésta es la función que desempeña e*l sistema manejador de bases de datos* (SMBD), que se define como una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de una tarea específica.

El SMBD maneja la abstracción de la información (el sistema esconde ciertos detalles de cómo se almacenan y se mantienen los datos), a diferentes niveles dependiendo del tipo de usuarios. Para la mayoría se ocultan los detalles del lugar y la forma del almacenamiento, la forma de recuperar y la actualización de los datos, esto facilita un mejor control sobre las bases.

La abstracción de la información se puede representar de la siguiente manera, los usuarios en un grupo determinado realiza una consulta, por lo cual, cada uno obtiene la interfaz a la base de

[22] Wiederhold, Gio. Diseño de Bases de Datos. McGraw Hill. México. 1988. Pág. 417.

datos conceptual.

La base de datos conceptual es la representación abstracta de la base de datos física y los panoramas son abstracciones de porciones de la base de datos conceptual.

El gestor de archivos se mueve entre los datos conceptuales y físicos, y sólo el administrador de la base de datos tiene acceso la base de datos físicos.

Las principales funciones de un SMBD radican en la creación y mantenimiento de las bases de datos, minimizar las redundancias y manipulación de los datos.

En general, se pueden enumerar los siguientes objetivos para SMBD:

- Crear y organizar las bases de datos, a nivel conceptual y físico.
- Crear independencias entre los datos y las aplicaciones que los utilizan; una independencia física, que se puede modificar el esquema físico sin que afecte a los superiores y una independencia lógica, que se pueden modificar los esquemas conceptuales sin modificar los programas de aplicación.
- Manejar los datos de acuerdo a las peticiones de los usuarios, aquí se pueden establecer *prioridades* y se deben manejar *tiempos de respuestas*, ya que éstos no deben ser muy excesivos para evitar la falta de calidad en el sistema. El tiempo de respuesta es el tiempo que transcurre desde que el usuario termina de realizar una petición al sistema hasta que empieza a recibir respuesta.
- Manejar sólo los datos necesarios, o sea, no tener información repetida y que todo dato sea útil en alguna aplicación.
- Tener una interacción con el manejador de archivos a través de las sentencias en DML. Así el manejador de base es el responsable del verdadero almacenamiento de los datos.
- Establecer y mantener excelentes trayectorias de accesos a las bases de datos, ya que deben ser rápidas y eficaces.
- Asegurar la protección de los datos con *respaldos* continuos, escalonados o alguna otra dependiendo de las características del sistema. Tener una seguridad contra accesos no autorizados o malintencionados de los usuarios.
- Suministrar mecanismos de seguimiento en las operaciones que se efectúan en las bases de datos, mediante procesos “espías” que mantienen archivos en los que se almacenan la fecha y hora de conexión de los usuarios, las operaciones realizadas, los datos modificados en cada sesión, problemas del software, etc.
- Versatilidad en las funciones para el manejo de las bases de datos.
- Control de concurrencia. Consiste en controlar la interacción entre los usuarios concurrentes para no afectar la inconsistencia de los datos.

Cuando ya se realiza el análisis de diseño y se estudia a grandes rasgos la documentación para el sistema, entonces se comienza a formular las propuestas de solución y el diseño del sistema, que en este caso es el siguiente capítulo y el segundo paso en la metodología que utilizaremos.

III

PROPUESTA DE SOLUCIÓN

3. Propuesta de solución

Al terminar de analizar el problema y documentar los recursos con los que se puede disponer, se comienza a estructurar una posible solución, desde la metodología a utilizar hasta el manejo de datos, sistemas de información, estructuras de tiempos, entre otros. Y empezaremos con la metodología a utilizar.

3.1 Metodología utilizada

La metodología que se utiliza para el Banco de Imágenes Astronómicas es la *metodología en espiral*, por las siguientes razones.

3.1.1 Metodología en espiral

La metodología en espiral tiene una naturaleza evolutiva compuesta por ciclos que se van dividiendo en tareas, lo que significa que cada vuelta tiende a mejorar a su antecesor y avanzar en el proyecto (pero puede tener sus excepciones si es que la metodología no se ha adaptado correctamente). Así, el ciclo más interno podría referirse a las formas más básicas, posiblemente a un prototipo en papel. La segunda interacción podría ser la definición de los primeros objetivos y puntos que al cliente lo lleven a delimitar más su proyecto; las vueltas subsecuentes irán cumpliendo cada uno de los requerimientos del sistema.

Como se define en el capítulo anterior, cada una de las vueltas que componen la metodología en *espiral* se dividen en sectores o tareas que dependen del sistema en particular, ya que varían conforme al tamaño del software, a las características del sistema, al personal con que se cuenta, los clientes, los fondos económicos y otros puntos tanto externos, como internos en el sistema.

Las tareas de software deben proporcionar la estrategia para mantener un riguroso nivel de calidad en el sistema, pero sin aumentar con un trabajo innecesario.

En general, se pueden aconsejar las siguientes tareas, ya que la mayoría de los proyectos de software son afectados por los mismos factores,

1. *Conceptos de operación*, es la primera entrevista con el cliente, se definen los detalles más básicos.
2. *Requerimientos del software*, se detallan los objetivos del sistema, tipos de interfaces máquina–usuario, fechas de reuniones, requerimientos de bases de datos, etc.

3. *Diseño de producción de software*, los ingenieros del software estipulan los detalles mismos del proyecto como el sistema operativo, bases de datos, etc., se definen los grupos también se establecen reuniones, asesorias y las formas de comunicación entre el equipo.
4. *Diseño detallado*, aquí se comienzan a generar los modelos de análisis del sistema, que se dividen en cuatro fases:
   a) Diseño de datos, transforma la información en estructuras entendibles para un diseñador, donde las principales fuentes de información son los diagramas entidad–relación y el diccionario de datos.
   b) Diseño arquitectónico, se definen las relaciones entre los módulos de control, esto es, combina la estructura del programa y la estructura de datos definiendo las interfaces que permiten el flujo de datos.
   c) Diseño de interfaz, describe la comunicación del mismo software con los sistemas que operan con él y con la gente que lo va a emplear. La información la obtienen principalmente de los diagramas de estado.
   d) Diseño procedimental, es la definición del programa en un lenguaje que pueda ser entendido por cualquier persona, como las reglas que se establecen para un aprendizaje factible. Las construcciones son: la secuencia, la condición y la repetición, por lo tanto, se basan en gran parte en los diagramas de flujo, porque son muy comunes para todo aquél que se desenvuelva en el desarrollo del software.
5. *Código*, en base al estudio anterior se comienza a estructurar la programación, las funciones, las interfaces, las bases de datos, etc., donde los bloques pueden ir realizándose en paralelo o en serie de acuerdo al mapa de administración de tiempos realizado.
6. *Prueba unitaria*, a cada una de las tareas de cada subgrupo, se le asignan pruebas de acceso y rendimiento para tener un mayor control de la calidad del sistema.
7. *Integración y pruebas*, todas las particiones hechas del proyecto se juntan y se forma una unidad, se realizan pruebas de calidad y se configuran los detalles que puedan surgir en la unión de las particiones.
8. *Pruebas de aceptación*. *Implementación*, el proyecto es presentado al cliente, se presentan las pruebas ya hechas (pruebas de integridad, de unidad, etc.) y se espera la aceptación por parte de éste. Se instala en el lugar definitivo de operación, también pueden incluirse asesorias o cursos al personal que va a operar el sistema en la empresa o institución.

Para poder delimitar las regiones de tareas en un proyecto específico (en este caso el BIA), se observan también otros puntos característicos, ya que como se mencionó anteriormente no se definen para ningún sistema, por ejemplo, las tareas de un proyecto muy grande serían consideradas muy exageradas para un proyecto de menor magnitud y viceversa.

Un punto de los antes mencionados es el *tipo de proyecto* y otro el establecer el *grado de rigor* del proyecto sobre el que se está trabajando.

3.1.2 Grado de rigor y tipo de proyecto

Aunque es difícil determinar a qué **tipo de proyectos** pertenecen los sistemas de ingeniería por su naturaleza, Pressman menciona que en general, la mayoría de las organizaciones de software encuentran proyectos en las siguientes clasificaciones:

A. Proyectos de desarrollo de concepto.
   Entran en la clasificación, los proyectos que investigan un nuevo concepto, ya sean en el ámbito privado, como industrias farmacéuticas, químicas, militares o dentro de las universidades públicas o privadas.
B. Proyectos de desarrollo de una nueva aplicación.
   Aquí se encuentran los proyectos que son delimitados por un cliente específico, son los encargos particulares de alguna empresa.
C. Proyectos de mejoras de aplicaciones.
   Son aquellos que se encargan para que sea actualizado o mejorado el sistema, podrían ser las interfaces, el funcionamiento, los rendimientos, etc.
D. Proyecto de mantenimiento de aplicaciones.
   Son los que corrigen, adaptan o amplían el software, muchas veces sin que los cambios sean vistos por el usuario a corto plazo.
E. Proyectos de reingeniería.
   Son los proyectos que hacen una reconstrucción a un sistema cuando ya está en funcionamiento.

El Banco de Imágenes Astronómicas cae dentro del ***grupo B: Proyectos de desarrollo de una nueva aplicación***, ya que se le considera como un sistema que el Instituto de Astronomía (IAUNAM) necesita para tener un mejor manejo de las imágenes que se trabajan en él. El IAUNAM en este caso, es el cliente.

Los **grados de rigor** se definen a partir de todas las especificaciones y estudios preliminares del sistema y es la forma en *cómo* se debe tratar el proyecto, ya sea con un mayor rigor y exactitud en los lineamientos o no; esto puede traer algunos puntos de discusión, ya que, independientemente del proyecto, ***todos*** los proyectos en sí, tienen una gran importancia y se deben apegar a lo establecido en los objetivos iniciales, así como respetar los tiempos y la entrega del sistema con la más alta calidad de desarrollo; pero la importancia entre los sistemas puede diferir grandemente entre los que se estén trabajando, por ejemplo, no tiene el mismo peso trabajar con un sistema que maneja la información de una base de datos grande, que uno para obtener la raíz cuadrada de algún número.

Los grados de rigor se pueden clasificar en:

1) *Casual,* son proyectos pequeños en donde las actividades de pruebas se minimizan y son

considerados pequeños desarrollos de software. Independientemente de su tamaño llevan todos los procesos de la ingeniería del software. La documentación es mínima.

2) *Estructurado*, es más detallado que el anterior, llevan un proceso de calidad más sofisticado, pruebas específicas y una documentación más amplia. El número de tareas aumentan considerablemente con respecto al anterior, pero no sobrepasa a proyectos que están catalogados en la siguiente fase.
3) *Estricto*, aquí se aplican todas las medidas de seguridad para la obtención de un sistema de calidad, todo el proceso completo con la más alta disciplina, grados de error, análisis de riesgos, planes de contingencia, etc.

Existe una cuarta clasificación, pero siguiendo una estructura lógica de las anteriores propuestas, no tiene mucho que ver directamente con éstas y en algún momento podría clasificarse con un grado de rigor casual; por lo tanto, se menciona la cuarta clasificación dando pie a lo citado.

* *De reacción rápida*, se pueden catalogar como métodos de bomberos porque se realizan en tiempos muy cortos, la documentación es entregada después del proyecto y las pruebas muchas veces se realizan conforme ya trabaja el sistema.

Para definir el grado de rigor se toman en cuenta puntos ya delimitados, llamados criterios de adaptación y un valor que Pressman define como el valor Selector del Conjunto de Tareas (SCT).

Entre los criterios de adaptación se pueden definir once puntos:

1. Tamaño del proyecto.
2. Número potencial de usuarios.
3. Misión crítica.
4. Longevidad de la aplicación.
5. Estabilidad de los requisitos.
6. Facilidad de comunicación con el cliente o el desarrollador.
7. Madurez de la tecnología aplicable.
8. Limitaciones de rendimiento.
9. Características ya establecidas o no.
10. Personal del proyecto.
11. Factores de reingeniería.

A cada uno de los puntos anteriores se le asigna un número, que es el grado que se aplica según se necesite de éstos, por ejemplo, el personal del proyecto, si es mínimo se les da el valor de *uno*, en caso contrario se le asigna un valor de *cinco*.

Las operaciones y pasos para obtener el valor SCT se muestran en la Tabla 3-a y son los siguientes:

1) Después de asignar los valores a los criterios de adaptación (del 1 al 5), se multiplican por los factores de ponderación, que van de 0.8 a 1.2. Los *valores de ponderación* son una

indicación de la relativa importancia de un criterio de adaptación a los tipos de software desarrollados dentro del entorno de la empresa, (los valores que aquí fueron tomados son un ejemplo de varios proyectos, pero basados principalmente en los valores que estipula Pressman).

2) El resultado se vuelve a multiplicar por el *multiplicador de punto de entrada*, que depende del tipo de proyecto a realizar, como se definió que es un proyecto para el desarrollo de una nueva aplicación, entonces se multiplica por los valores delimitados en la estructura del SCT.
3) Se calcula la media de todas las entradas en la columna *Producto* y éste es el valor SCT.
4) De acuerdo con el SCT y la Tabla 3-b, se delimita el grado de rigor del proyecto.

Tabla 3-a Cálculo del Selector de Conjunto de Tareas

Criterio de Adaptación	Grado	Peso	Multiplicador de punto de Entrada Desarrollo de una Nueva Aplicación	Producto
Tamaño del Proyecto	4	1.2	1	4.8
Número de Usuarios	5	1.1	1	5.5
Importancia para el Negocio	4	1.1	1	4.4
Longevidad	4	0.9	1	3.6
Estabilidad de los Requisitos	5	1.2	1	6
Facilidad de Comunicación	4	0.9	1	3.6
Madurez de la Tecnología	3	0.9	1	2.7
Limitaciones de Rendimiento	4	0.8	1	3.2
Empotrado/No empotrado	4	1.2	1	4.8
Personal del Proyecto	2	1	1	2
Interoperatividad	4	1.1	1	4.4
Factores de Reingeniería	2	1.2	0	2.4
Selector de Conjunto de Tareas (SCT)				***3.95***

Tabla 3-b Delimitación de los grados de rigor.

Valor del selector del conjunto de tareas	Grado de Rigor
SCT < 1.2	Casual
1.0 < SCT < 3.0	Estructurado
SCT > 3.0	Estricto

El valor resultante es 3.95 y basándonos en la Tabla 3-b, se deduce que el proyecto tiene un grado de rigor *ESTRICTO*.

3.1.3 Regiones de tareas

Para seleccionar las regiones de tareas de la metodología en espiral para nuestro proyecto, se toma en cuenta los grados de rigor, necesidades, tipo de proyectos y las propuestas descritas anteriormente. Por lo tanto, para el proyecto del BIA, se definen las siguientes tareas:

1. La primera tarea es la *comunicación con el cliente.*
Se establece la comunicación con el jefe del proyecto, investigadores y gente de cómputo; se delimitan alcances, opciones, requisitos y todo lo referente para realizar el análisis preliminar del proyecto. Es el ámbito del concepto.

2. La segunda tarea es *planificación y análisis de riesgos.*
Se analizan los recursos que se establecieron en la etapa anterior y basándose en estos datos, se realiza el estudio del análisis de riesgo, estudio de tiempos y se comienzan los primeros pasos del diseño del software que son la estructura de datos.

3. La tercera tarea es *desarrollo-ingeniería* o *prueba del concepto.*
Se define la propuesta de solución a utilizar en base a los análisis y diagramas de la etapa anterior. Se genera el modelado de datos, diagramas de procesos jerárquicos, flujo de datos, diagramas entidad–relación, diccionario de datos, etc., y se asignan tareas de acuerdo a las fechas ya establecidas.

4. La cuarta tarea es *construcción y adaptación.*
Es la realización tangible de lo que se ha hecho en papel, se incluyen las etapas de pruebas de calidad y rendimiento. Se comienza a realizar la programación de cada una de las estructuras y la unión de éstas si es que fueron diversas etapas. Se prueba e instala el prototipo.

5. La quinta etapa es la *evaluación con el cliente.*
Se busca la opinión final del jefe de proyecto, se le entregan las pruebas realizadas (pruebas de carga, pruebas de punto de foco, prueba beta, checar las condiciones de errores, de unidad y de función) y la documentación, dictamina si el proyecto termina o se establecen los lineamientos para una nueva iteración en la espira.

La Fig. 3-1 muestra el esquema del diagrama en espiral del BIA.

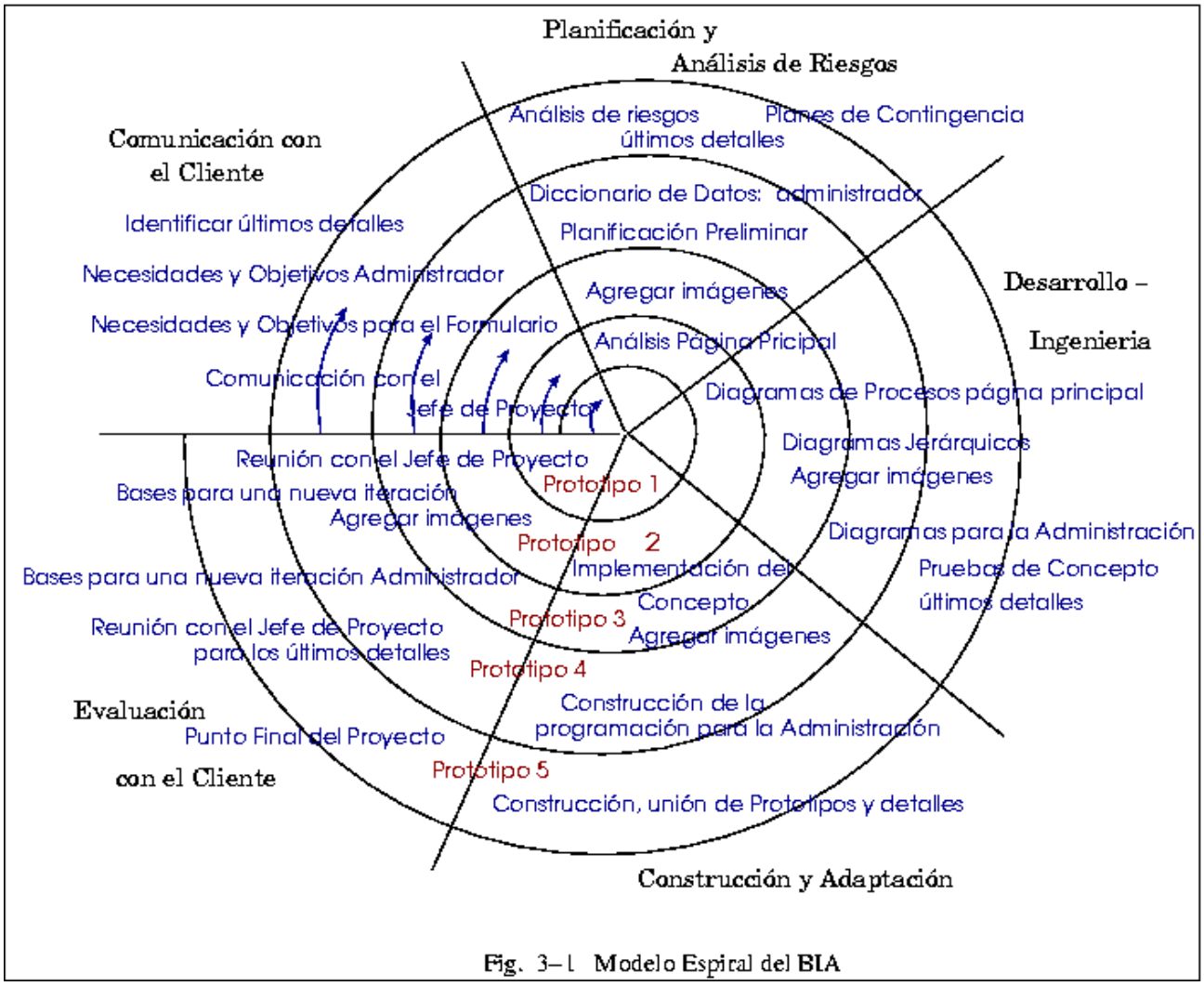


Fig. 3–1 Modelo Espiral del BIA

3.2 Estudio de tiempos

Tiene una gran importancia el establecer los tiempos en cualquier sistema, lo que conlleva a que se maneje una calendarización del inicio y del término del proyecto. No es fácil delimitar el tiempo de realización de un sistema, por lo que el ingeniero que establece los *tiempos* debe basarse en los estudios de la ingeniería del software y ayudarse con la experiencia previa.

Los diagramas de PERT y Gantt proporcionan una manera de organizar los tiempos del proyecto, además de ayudar a establecer las rutas críticas; pero hay que recordar que las gráficas son una estimación del proyecto en general y por lo tanto, pueden tener variaciones que deben ser

previstas por medio de tiempos de respaldo, en el caso del BIA, por el desarrollador del sistema.

El diagrama de Gantt se basa en el conjunto de tareas que son delimitadas por la metodología a seguir, en este caso, la metodología en espiral. La duración y la fecha de inicio son las entradas de cada tarea. Cada región de tarea de cada iteración que se realice debe ocupar un espacio en el esquema general.

En el diagrama de Gantt se observan fácilmente si las actividades pueden ser llevadas en una forma paralela y cuáles de ellas tienen que ser consecutivas. Cuando el proyecto es realizado por varias personas, el paralelismo puede resultar muy práctico además de indispensable, ya que todo el personal no puede realizar la misma actividad a un mismo tiempo. Pero en el caso del BIA, la realización de procesos en paralelo no es muy factible, ya que sólo es una persona la que realiza el proyecto y aunque indirectamente si se puede trabajar en procesos paralelos, el objetivo final (disminución de tiempos) no se puede modificar.

Los diagramas de Gantt utilizan barras con diferentes longitudes para representar la duración de cada una de sus actividades, además que muestran un colchón de tiempo para cada una de las etapas por cualquier inconveniente que pueda surgir en el transcurso de trabajo.

La Tabla 3-c, que se encuentra en el Apéndice delimita el gráfico del tiempo GANT para el BIA.

En el diagrama PERT del BIA (Fig. 3-2) se observan los inconvenientes de la no paralelización, ya que el inicio y la duración planeados en las etapas se muestran casi en una línea y la ruta crítica no se observa de una manera estricta.

3.3 Modelado del sistema

En el análisis del sistema se desarrollan los modelos generales del proyecto, es decir, se desarrolla un modelo del funcionamiento del sistema que se expresa en términos de procesos, relaciones, en el control de flujos y en las transformaciones funcionales.

Para el análisis se realizan tres tipos de modelos, *el modelo de objetos, el modelo dinámico* y *el modelo funcional*.

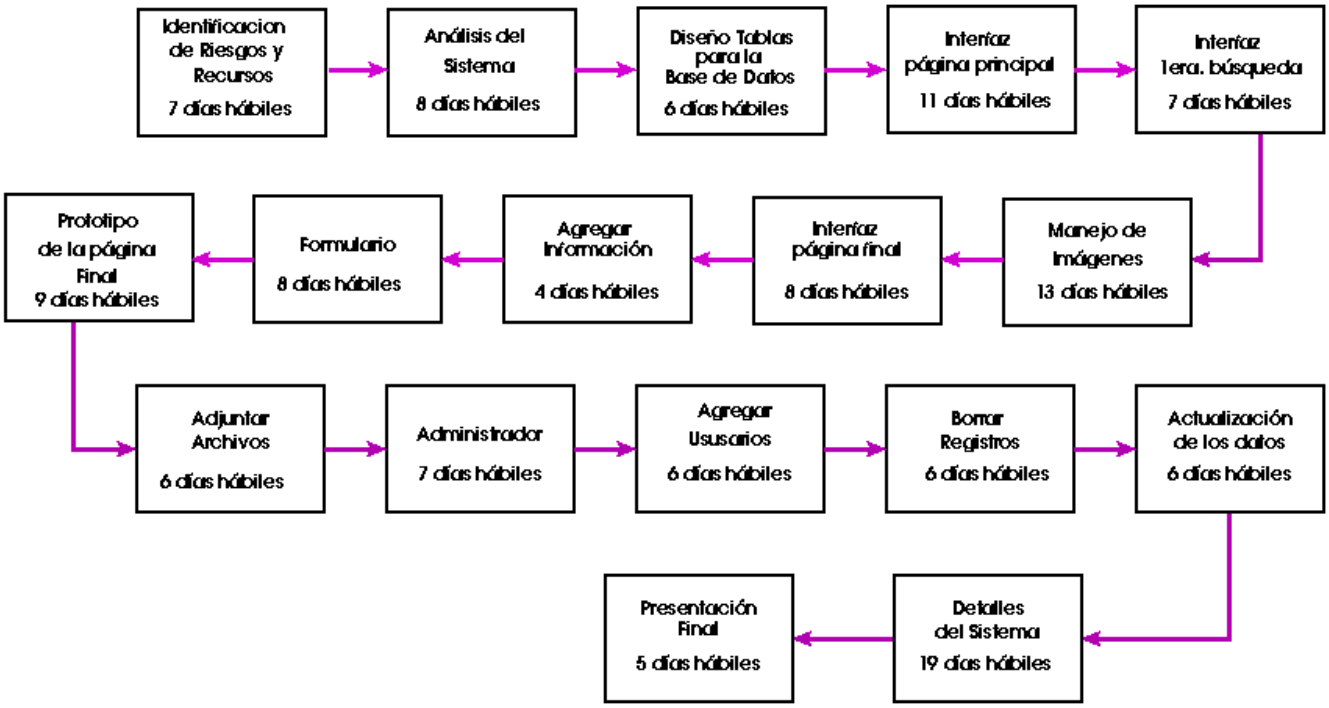


Fig. 3–2 Diagrama PERT del BIA

3.3.1 Modelo de objetos

El primer modelo de objetos va a describir una estructura estática de los objetos del sistema y las relaciones que existen entre éstos, principalmente utiliza el diagrama de objetos, que se identifica por el estudio de clases, sus atributos y las operaciones que existen entre ellos.

Los elementos estructurales para el modelo de objetos son cuatro:

a) **Objeto**, es la representación de cualquier composición de información, la abstracción, un concepto bien delimitado que tiene un significado determinado para el sistema. Todos los objetos tienen una identidad y son distinguibles.

b) **Clases**, es un grupo de objetos, con atributos, características y operaciones propias.

c) **Operaciones**, las operaciones son acciones que pueden realizar las clases.

d) **Atributos**, definen las propiedades de los objetos de una clase.

También se encuentran las *asociaciones*, que establecen las relaciones entre las clases, pero éstas no están catalogadas como elementos estructurales.

La notación empleada para representar gráficamente las estructuras del modelo de objetos se muestra en la Fig. 3-3.

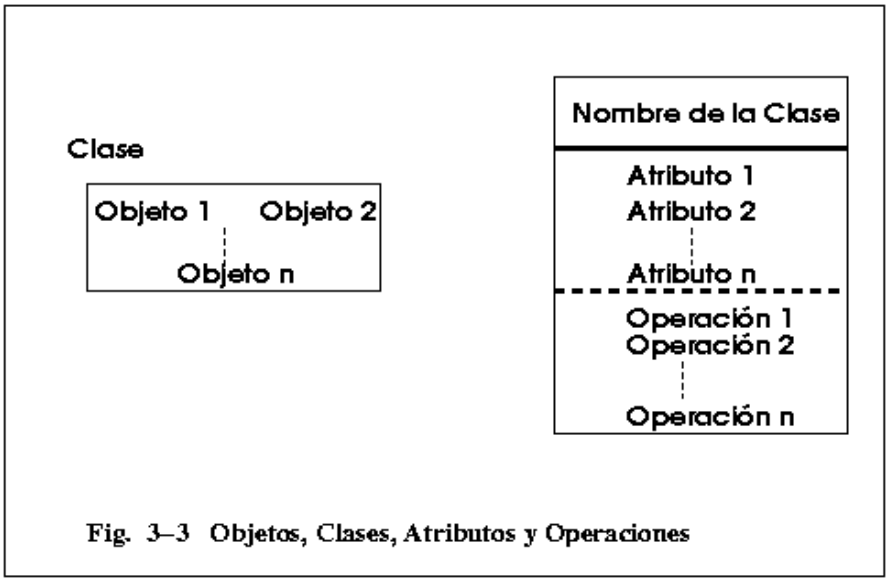


Fig. 3–3 Objetos, Clases, Atributos y Operaciones

Para el diagrama de objetos del BIA, se tienen que identificar las clases, las asociaciones entre las clases y los objetos que la componen, la Fig. 3-4 muestra el modelo de objetos del BIA.

3.3.2 Modelo dinámico

El segundo modelo para analizar es el *modelo dinámico*, donde supone que en cualquier momento el sistema puede encontrarse en un estado diferente y sólo cuando éste recibe una entrada el sistema se transfiere a otro, describe los aspectos conforme pasa el tiempo y específica e implanta los aspectos de control del sistema; su principal herramienta consiste en los diagramas de estado que se conforman por *estados, transiciones* y *eventos*.

En los diagramas de estado no se manejan flujos de datos ni variables de transferencia, sino que se manejan elementos para las transiciones entre las etapas; para realizarlo se tienen que identificar cada uno de los **estados** del proceso y los argumentos para moverse entre sí (**ligas o transiciones**), así como los **eventos** que ocasionan que se disparen estos argumentos.

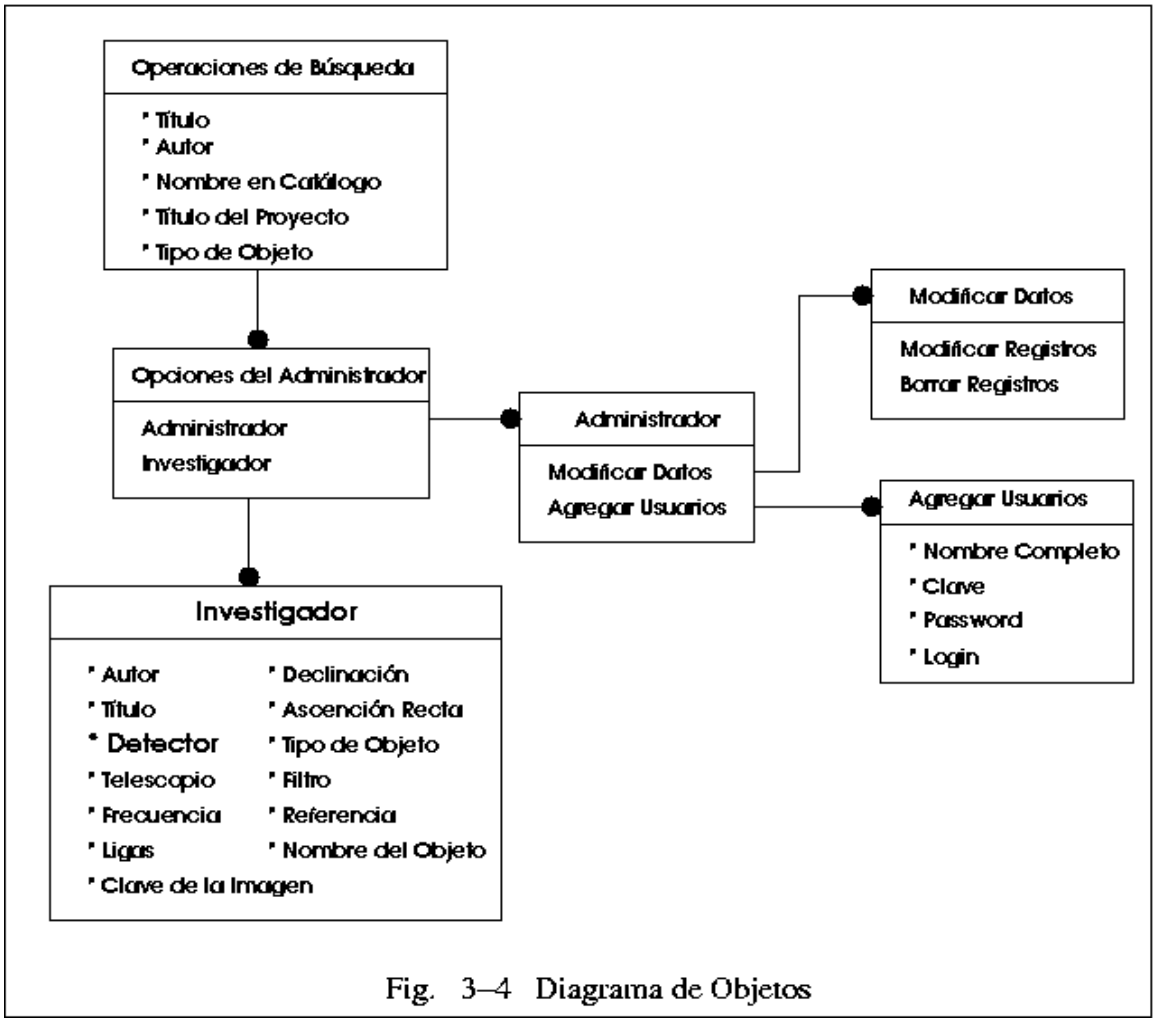


Fig. 3–4 Diagrama de Objetos

En algunos proyectos si se pueden observar los estados iniciales y los estados finales (que se les denominan “de un solo ciclo de vida”), pero en la mayoría de los sistemas no se observan estos estados.

La Fig. 3-5 muestra el diagrama de estado para el BIA, donde se observan los bloques que lo componen y la descripción de cada uno de ellos se encuentran en la Tabla 3-d. Cabe mencionar que estos bloques adquieren una gran importancia cuando se genera el diccionario de datos, en nuestro caso, más adelante.

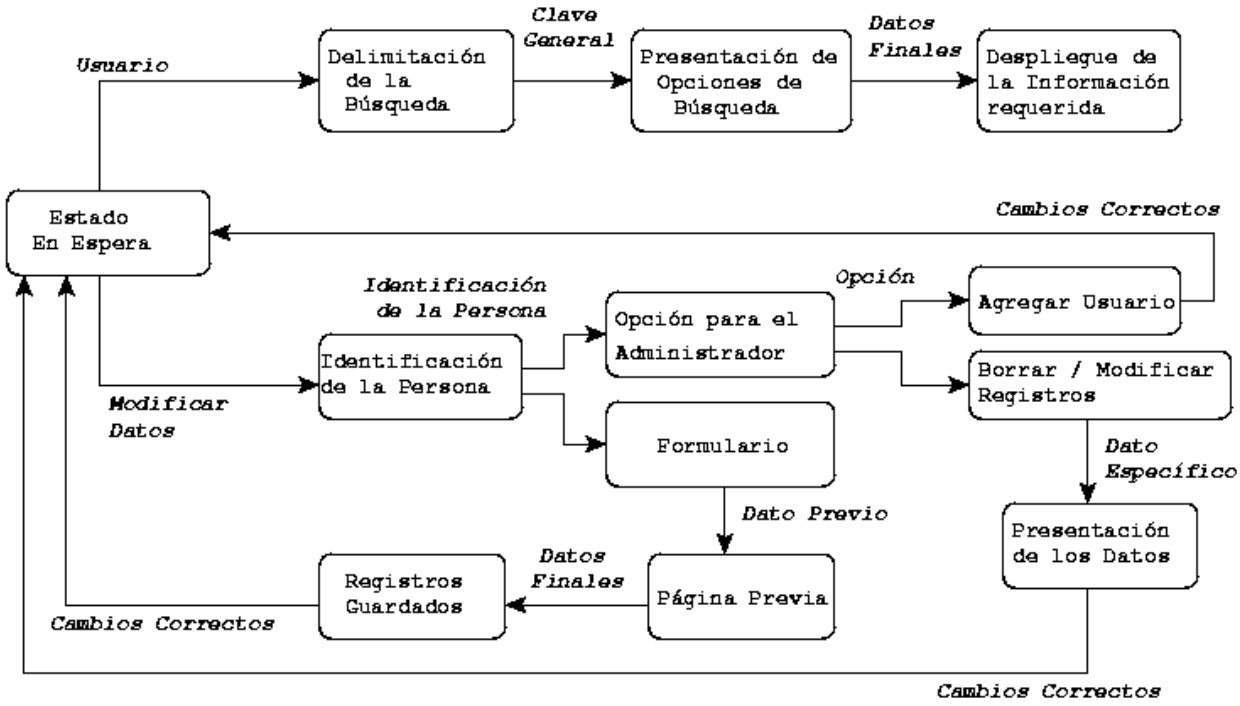


Fig. 3–5 Diagrama de Estado

3.3.3 Modelo funcional

El tercer y último paso es el modelo funcional el cual describe las transformaciones de los valores dentro del sistema. Su principal herramienta son los diagramas de flujo de datos, con los que se pueden representar cálculos, entradas, flujos de datos y archivos para almacenar datos.

Los diagramas de flujos de datos pueden ser representados en diferentes niveles de abstracción que van tomando valores que comienzan del cero, que son los diagramas más generales, hasta los más específicos, donde pueden llegar hasta la descripción o acceso de una sola variable.

Los elementos que los construyen son los procesos, flujos de datos, almacén de datos y los terminadores, que describiremos más adelante dada la estructura del capítulo, además de los diagramas de flujos del sistema.

Tabla 3-d Descripción de los estados.

Estado	***Descripción***
En espera.	Espera la entrada inicial que delimite el siguiente camino.
Delimitación de la búsqueda.	Se elige el tema para la búsqueda general, se genera la clave principal.
Presentación de opciones de búsqueda.	El estado muestra los bloques con la información básica que contengan la clave principal. Se puede generar la clave particular para la siguiente etapa.
Despliegue de la información requerida.	El estado despliega la información sobre la imagen, características particulares, archivos y ligas.
Identificación de la persona.	El estado identifica quién está utilizando el sistema y genera el pase a la sección que le corresponde.
Estado para el administrador.	El estado muestra las opciones para el administrador, ya sea dar de alta a nuevos usuarios o modificar la base de datos.
Agregar usuario.	Se presenta el formulario para agregar nuevos usuarios y se realiza el guardado de la información en la base de datos.
Borrar / Modificar registros.	El estado identifica por medio de un registro (número de identificación) los valores que serán modificados.
Presentación de los datos.	El estado muestra un formulario con los datos solicitados y se determina si éstos van a ser borrados o sólo se modifican. Determina los cambios correctos.
Formulario.	El estado presenta el formulario que tendrá que ser llenado por el investigador, en éste se incluyen todos los datos de la imagen.
Página previa.	El estado muestra cómo se presentará la página final de la imagen. Puede regresar al estado anterior para ser modificado o al estado siguiente para que sean guardados los registros.
Registros guardados.	Se está conforme con la página final, la información es archivada en la base de datos y en los archivos del sistema.

3.4 Diccionario de datos

La documentación del proyecto toma un aspecto muy importante en la estructura general del sistema y para realizarlo existen diferentes tipos, la *documentación interna* y la *documentación externa*. La primera es cuando se escriben los comentarios en el mismo código, con líneas de comentarios con ayuda de diagonales, antidiagonales, signos de más-menos, mayor o menor que, asteriscos, etc. y la documentación externa es cuando se realizan documentos fuera del sistema, como manuales y en este caso, diccionario de datos.

Los diccionarios de datos son documentos que forman parte de las normas y especificaciones del proyecto. Nos ayudan a tener una mayor información sobre las entidades y los atributos, ya sea

para el sistema o para la base de datos.

La parte medular de los diccionarios la constituye la definición y descripción de las entidades, donde se consideran de manera integral aquellas características que permiten conceptualizar los registros en unidades discretas. Cada entidad tiene un nombre, definición y atributos.

Una de las ventajas al utilizar un diccionario de datos (ya que es elemental), es porque facilita el control de cada una de las entidades y atributos que forman la estructura tanto del sistema, como de las bases de datos, además se pueden controlar dinámicamente las estructuras de las interfaces del usuario para las diferentes pantallas del sistema.

Para obtener el diccionario de datos del BIA, se toma en cuenta el modelos de objetos, el modelo dinámico y el modelo funcional, ya que se necesita describir (principalmente de este último) cada uno de los procesos y sus utilidades; para la base de datos el diccionario se rige con las tablas y registros que la conforman.

3.4.1 Diccionario de datos del sistema

Usuarios. Son las personas que sólo buscan información independientemente si son investigadores, estudiantes o público en general. Son agentes pasivos, ya que no pueden modificar dato alguno.

Investigador. Es la persona que trabaja sobre las imágenes y que puede agregar sus investigaciones al banco.

Administrador. Es la persona encargada del BIA y está autorizada para agregar usuarios, modificar y borrar los registros de la base de datos.

Número de identificación Es el número que identifica la información de una sola imagen, es única y es asignada por el sistema de acuerdo a los registros guardados en la base de datos.

Clave general. Es la variable que delimita el primer filtro, es la primera selección por parte de un usuario.

Clave particular. Es la variable única que delimita toda la información de la imagen, contiene el *número de identificación*.

Investigación-Divulgación. El sistema espera la entrada de cualquier usuario.

Opciones. Aquí el usuario elige la opción que le interese, dependiendo del objetivo que tenga. Si es usuario, elegirá el primer filtro de la información requerida (*clave general*), si no lo es, pasa al siguiente bloque para que lo identifique el sistema como *investigador* o *administrador*.

Muestra de opciones. El usuario entra al proceso con la delimitación del primer filtro y se muestran todos los bloques de información con esta característica, si lo requiere puede acceder a la información completa y generar la *clave particular*.

Página final. Se presenta toda la información sobre la imagen guardada en el banco. El proceso del *usuario* relativamente se termina hasta este punto, ya que puede ir a otras imágenes, pero sobre el mismo procedimiento.

Identificación. Aquí se identifica si la persona que accesa a la página es *administrador* o *investigador*, esto se logra con el login y el password escritos. Dependiendo de la persona el sistema conduce a una sección u otra.

Formulario de datos. En este proceso sólo entra el investigador y se presenta un formulario para que sea llenado con las características de la imagen. El formulario tiene los siguientes campos:

- Autor.
- Título del proyecto.
- Ascensión recta.
- Declinación.
- Frecuencia.
- Nombre del objeto.
- Tipo de objeto.
- Telescopio.
- Detector.
- Descripción.
- Filtro.
- Referencias.
- Ligas.
- Email del autor
- La clave de la imagen. Esta clave es asignada por el sistema de acuerdo a los trabajos que el investigador tenga anexados en la base de datos. Puede ser modificada e iniciar una nueva lista de trabajos.
- La imagen. Es un adjuntador de archivos que copia el archivo de la imagen del directorio del autor a los archivos del sistema. Aquí se generan los siguientes pasos para poder guardar la imagen:

1. *Adjuntar archivos*. Se hace la selección del archivo que contiene la imagen.
2. *Archivo guardado.* El archivo se guarda en el espacio asignado al proyecto.
3. *Notificación de la imagen.* En esta etapa se muestra si el archivo fue correctamente guardado, si fue así, se despliegan las características del archivo de la imagen, si no, se da una breve explicación del porqué ocurrió el error.

Página previa. Se muestra la página como finalmente se observa en la búsqueda general. Se puede regresar al formulario para que sea modificada o pasar a la siguiente etapa para que los

datos sean guardados.

Datos guardados en la base. En esta etapa los registros se guardan en la base de datos y se despliega una bandera por parte del sistema, positiva o negativa.

Notificación de registros. Dependiendo de la bandera generada en la etapa anterior (*datos guardados en la base*) se notifica el resultado, si la bandera es positiva se despliegan los datos guardados y se pasa a la etapa de *Opciones*; si la bandera fue negativa significa que los datos no fueron guardados y se regresa a la etapa de *formulario*.

Opciones del administrador. El administrador define entre la opción de *agregar nuevos usuarios* o *modificar datos,* en este último se pide el *número de identificación.*

Agregar nuevos usuarios. Se presenta un formulario con cuatro campos para que sean llenados.

1. Nombre del investigador.
2. Login.
3. Password.
4. Iniciales para su clave.

Guardar los datos. Los datos generados en la etapa *agregar nuevos usuarios* son guardados en la base de datos.

Notificación de nuevo usuario. Se confirma si los registros fueron guardados correctamente. Regresa a la etapa Investigación-Divulgación.

Presentación de datos. Se presentan los datos en un formulario, de acuerdo al número de identificación. Si van a ser modificados los registros, los cambios se registran aquí mismo.

Modificar o borrar datos. Se guardan los cambios en la base de datos, ya sea que se modifiquen o que se borren completamente.

Notificación de nuevos cambios. Se notifica sí al guardar los cambios hubo o no algún problema.

3.4.2 Diccionario de datos de la base de datos

Se tienen tres tablas para almacenar los datos de la imagen y una para identificar a los usuarios, que permite entrar a las secciones de *agregar imágenes* y *administrador*.

a) Tabla GENERAL

La tabla contiene la mayoría de los elementos para identificar la imagen, aunque son

muchas entidades no son muy grandes en la cantidad de información que contienen, ya que pueden ser definidas con uno o dos caracteres. La tabla se constituye por los siguientes registros y la llave primaria en este caso es **ID**.

Field	Type	Null	Key	Default	Extra
ID	int(10) unsigned		PRI	NULL	auto_increment
CLAVE_IMAGEN	varchar (10)				
DECLINACIÓN	varchar (20)	YES		NULL	
NOMBRE_OBJETO	varchar (150)				
TITULO_PROYECTO	varchar (200)				
TIPO_OBJETO	varchar (100)	YES		NULL	
ASCENSION_RECTA	varchar (10)	YES		NULL	
TELESCOPIO	varchar (20)	YES		NULL	
FRECUENCIA	varchar (20)	YES		NULL	
DETECTOR	varchar (20)	YES		NULL	
FILTRO	varchar (5)	YES		NULL	
LIGAS	varchar (100)	YES		NULL	
REFERENCIAS	varchar (150)	YES		NULL	

b) *Tabla AUTORES*

La tabla contiene tres registros, que son específicamente los datos del autor, estos datos se separaron por ser los datos con los que se puede llegar a identificar al autor o autores de la imagen, independientemente del proyecto que registren. La llave primaria en este caso es **id_autor.**

Field	Type	Null	Key	Default	Extra
id_autor	int(10) unsigned		PRI	NULL	auto_increment
AUTOR	varchar (150)	YES		NULL	
email	varchar (100)	YES		NULL	

c) Tabla IM

La tabla contiene el nombre específico de la imagen, como se conoce en los diccionarios astronómicos y su descripción como objeto estelar, esta descripción puede llegar a

ser muy extensa, por lo que se separa junto con su nombre de pila. La llave primaria en este caso es **id_imagen**.

Field	Type	Null	Key	Default	Extra
id_imagenes	int(10) unsigned		PRI	NULL	auto_increment
NombreImagen	varchar (80)	YES		NULL	
DESCRIPCION	varchar (150)	YES		NULL	

d) Tabla AUTHENT

La última tabla que compone el sistema BIA es la identificación de las personas que pueden modificar la base de datos. Define si la persona es algún investigador para presentarle el formulario y poder agregar registros, o si la persona entra como administrador para presentarle el formato borrar–modificar y pueda agregar usuarios.

Field	Type	Null	Key	Default	Extra
login	varchar (20)		PRI		
pswd	varchar (30)				
claveAutor	varchar (10)	YES		NULL	
NombreAutor	varchar (150)				

Al terminar el análisis del proyecto, el siguiente paso de acuerdo a la metodología tomada, es el diseño del sistema, para lo cual, nos enfocaremos más a diagramas, esquemas y casi a la programación física del sistema.

IV

DISEÑO DEL PROYECTO

4. Diseño del proyecto

En el capítulo anterior (propuesta de solución) es donde realmente se comienza a forjar el diseño del proyecto, porque el análisis de los procesos y la estructura principal del sistema ya se han desarrollado; en este capítulo se comienzan a manejar los datos, el flujo de información y las estructuras jerárquicas, esto es, un poco más enfocado a un diseño arquitectónico.

Después del análisis del sistema, en el diseño se manejan principalmente cuatro partes:

a) Diseño de datos.
b) Diseño arquitectónico.
c) Diseño de la interfaz.
d) Diseño procedimental.

4.1 Diseño de datos

La totalidad de la información es difícil para delimitar, ya que en el sistema existen conjuntos de datos que pueden ser manipulados para algún objetivo específico y que pueden relacionarse entre sí y crear nuevos flujos de información, éstos a otros nuevos y así sucesivamente, pero esta larga cadena de relaciones no puede sobrepasar la línea que debe ser bien delimitada por los ingenieros del proyecto.

El diseño de los datos es una parte importante de la estructura de los cuatro pasos para el proceso del software, el mantener una idea que pueda ser entendida por cualquier persona que accede al proyecto es un tanto difícil si no se siguen ciertas reglas ya estructuradas. Para estas reglas, algunos estudios delimitan que para realizar un diseño del sistema se recomiendan seguir los siguientes pasos:

a) Organizar el sistema en subsistemas.
b) Identificar concurrencia inherente al problema.
c) Asignar subsistemas a procesadores y tareas.
d) Escoger una estrategia básica para implementar los almacenamientos en términos de estructura de datos, archivos y bases de datos.
e) Determinar los mecanismos para controlar el acceso a recursos globales.
f) Escoger la implementación del control del software.
g) Considerar las condiciones de frontera.
h) Establecer prioridades de decisión sobre características del producto de software.

El diseño se puede documentar y estructurar como un conjunto de modelos gráficos, entre los cuales se cuentan el *modelo entidad-relación,* los *diagramas de flujo de datos, diagramas de construcción* y los *diagramas jerárquicos*.

4.2 Diseño arquitectónico del proyecto

El diseño arquitectónico del sistema desarrolla un mayor nivel de abstracción en el sistema. Este diseño realiza un modelo por bloques, las conexiones entre éstos y todas las estructuras del sistema, Pressman lo define como "el diseño arquitectónico... desarrolla una estructura de programa modular y representa las relaciones de control entre los módulos. Además, el diseño arquitectónico combina la estructura del programa y la estructura de los datos, definiendo interfaces que permiten el flujo de datos a través del programa."[23]

Para poder encausar la información se van delimitando las estructuras antes dichas, al principio son esquemas muy generales en los que se pueden observar los subsistemas o componentes del proyecto, después se van delimitando los flujos de datos entre las estructuras y el intercambio de información que requieren cada uno de éstos, también se delimitan las interfaces y los datos de entrada y de salida, donde la estructura mínima de la que se componen son de tres pasos: entrada, procesamiento y salida.

Por lo tanto, un esqueleto puede tener varios niveles muy detallados o superfluos de los datos en la estructura general y dada la claridad o complejidad de éstos, la calidad del sistema se mueve incondicionalmente.

Los principales objetivos para cualquier modelo se pueden resumir en dos puntos:

A. Proveer un modelo preciso de las necesidades funcionales de la organización que actuarán como marco referencial para el desarrollo o actualización del sistema.
B. Dar un modelo que sea independiente de cualquier mecanismo o método de procesamiento y que permita la toma de decisiones certeras a cerca de técnicas de implementación y acoplamiento con sistemas existentes.

Los elementos para generar un modelo principalmente son cuatro: el proceso, el flujo de datos, el almacén y los actores; estos elementos se describen a continuación:

a) Procesos.

Un *proceso* representa el bloque donde se efectúa la transformación del valor de los datos, por lo tanto, se habla de una entrada, un proceso y una salida. La notación se muestra en la Fig. 4-1.

[23]Pressman, Roger S. *Ingeniería del Software: un enfoque práctico.* McGraw Hill. España. 1998. Pág. 250.

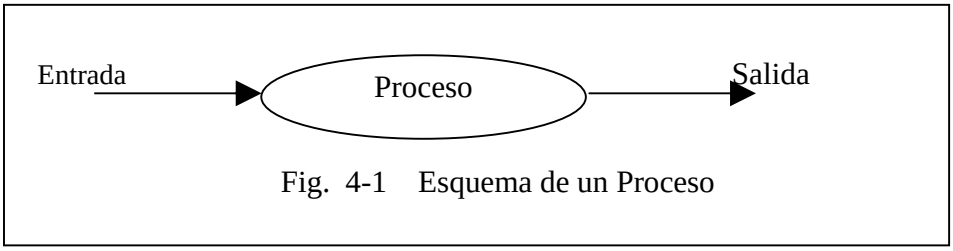


Fig. 4-1 Esquema de un Proceso

b) Flujo de datos.

Un *flujo de datos* conecta la salida de un proceso con la entrada de otro, esto es, agrupa los valores de datos entre objetos dentro de un camino, es la tubería del sistema y la notación se representa en la Fig. 4-2.

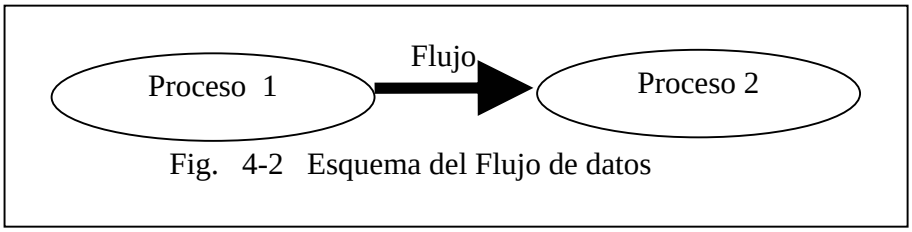


Fig. 4-2 Esquema del Flujo de datos

c) Actores

El *actor* es un objeto activo "... que conduce el grafo de flujo de datos, ya sea consumiendo o produciendo valores. Los actores están asociados con las entradas y salidas de flujo de datos en las fronteras de un diagrama, por ello, algunas veces se llaman terminadores".[24] Por lo regular se le asocia con una persona o un grupo, pero también puede ser otro sistema, como algún otro hardware o software con el que se comunique.

La Fig. 4-3 muestra la representación de los actores empezando y terminando un flujo de datos.

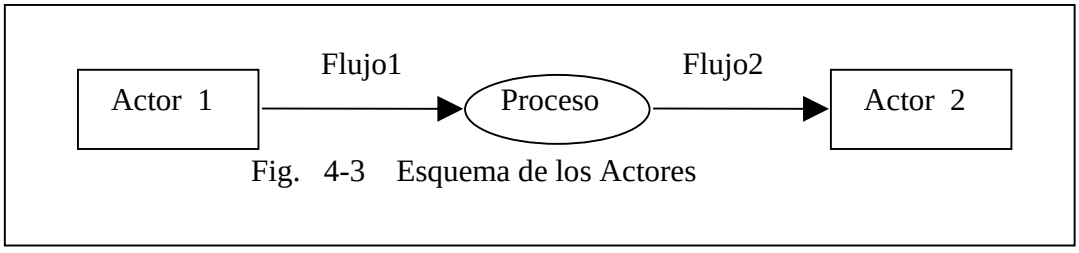


Fig. 4-3 Esquema de los Actores

d) Almacenamiento de datos.

El almacenamiento de datos es un objeto pasivo, ya que no modifica ningún dato en toda la estructura. Como su nombre lo indica sirven para guardar registros, ya sea como datos finales o

[24] Morán y S., Alberto Leopoldo. *Metodología de la Programación II.* España. 2001. Pág. 15

como datos temporales, los primeros son aquellos que no van a tener modificación alguna al ser llamados por el sistema, no así los datos temporales que sí serán modificados.

La notación para la representación son dos líneas horizontales y se muestran en la Fig. 4-4.

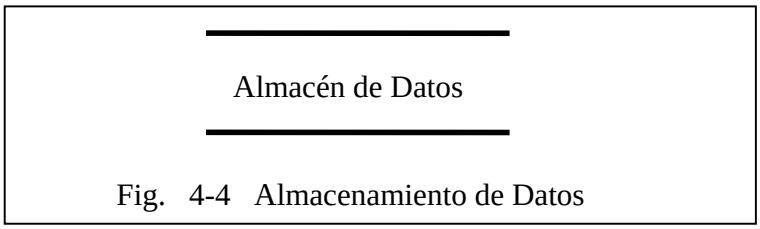
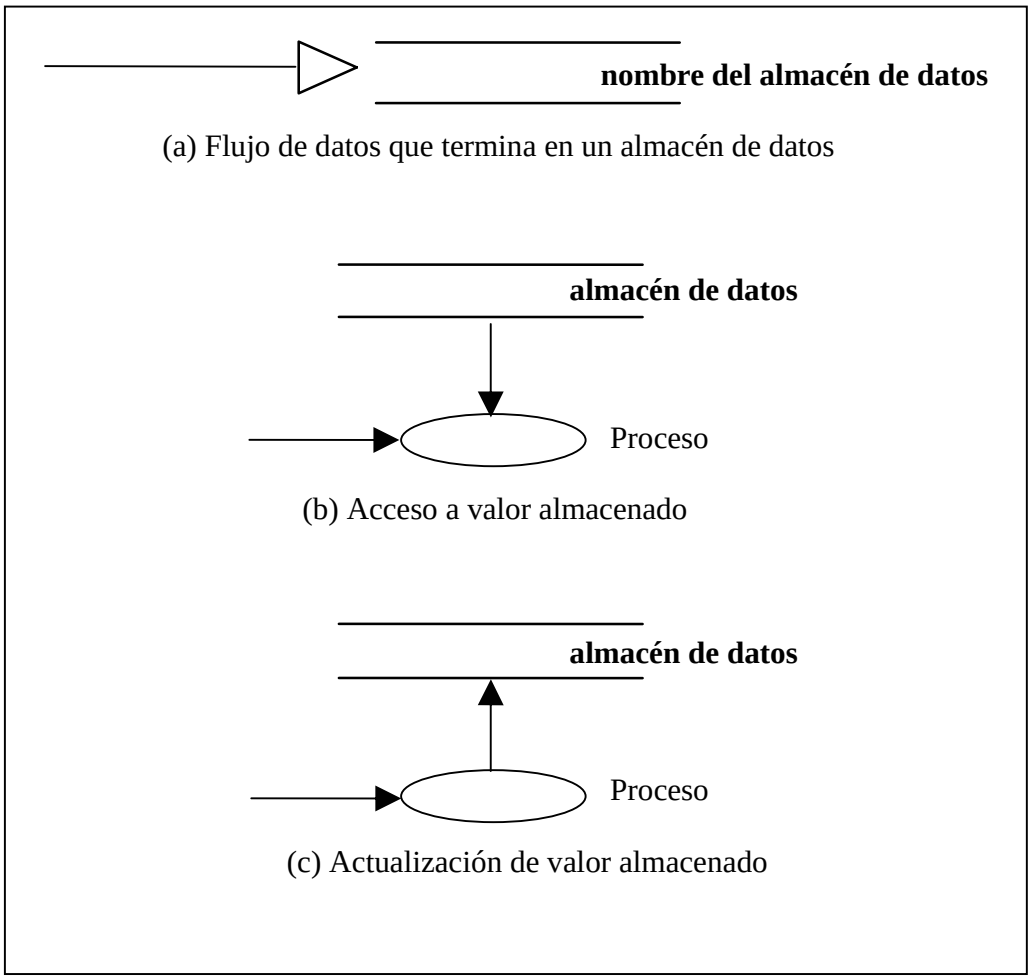


Fig. 4-4 Almacenamiento de Datos

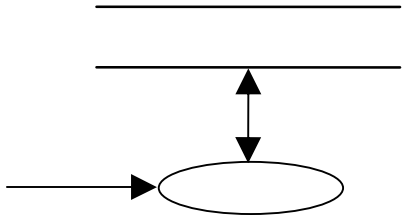
La Fig. 4-5 muestra las notaciones para diferentes tipos de acceso y actualizaciones en los datos.



(a) Flujo de datos que termina en un almacén de datos

(b) Acceso a valor almacenado

(c) Actualización de valor almacenado



(d) Acceso y actualización de dato almacenado

Fig. 4-5 Accesos y actualizaciones de datos

4.2.1 Diagrama de flujo de datos

Un diagrama completo de flujo de datos representa un proceso de alto nivel, esto es, una estructura muy general y abstracta (denominado nivel 0), pero los procesos en el diagrama pueden ser detallados cada vez más, logrando procesos de alto nivel, un diagrama de *nivel n* (donde *n* es mayor que 2) puede llegar sólo a detallar la descripción de una variable.

El diseño a un nivel 0 da un esquema sobre cada uno de los procesos que recorre un dato como un objeto, cuál es su principio y su fin, en dónde son guardados y de dónde se extraen, así como algunas de las modificaciones que tengan en el proceso de transferencia o los nuevos flujos que vayan surgiendo. Además de observar el camino de cada proceso y los cambios que les vayan ocurriendo. Pocos ingenieros piensan que este modelo no es útil para el desarrollo en general, pero la visión global del sistema, el no involucrar demasiados datos y ver una estructura alterna es muy útil para las personas que llegan a involucrarse por primera vez en el sistema.

Para ejemplificar las diferencias entre las estructuras, el nivel 0 del BIA se muestra en la Fig. 4-6 y el nivel 1 en la Fig. 4-7, y se puede observar en la primera un menor detalle en las variables y bloques con respecto a la segunda.

El modelo entidad-relación del sistema es una estructura que delimita de una manera muy sencilla todo el proceso, se compone de estructuras similares a otros diagramas, por ejemplo, *flujo de datos*, *procesos* e identidades como los *terminadores* que muestran las entradas y las salidas de la información, pero además, en el modelo entidad-relación se comienzan a manejar los nombres de variables, lo que hace que el diagrama se observe con una mayor cantidad de información, pero sin perder su esencia de ser un diagrama sencillo. El diagrama entidad–relación se encuentra en la Fig. 4-8 en el Apéndice.

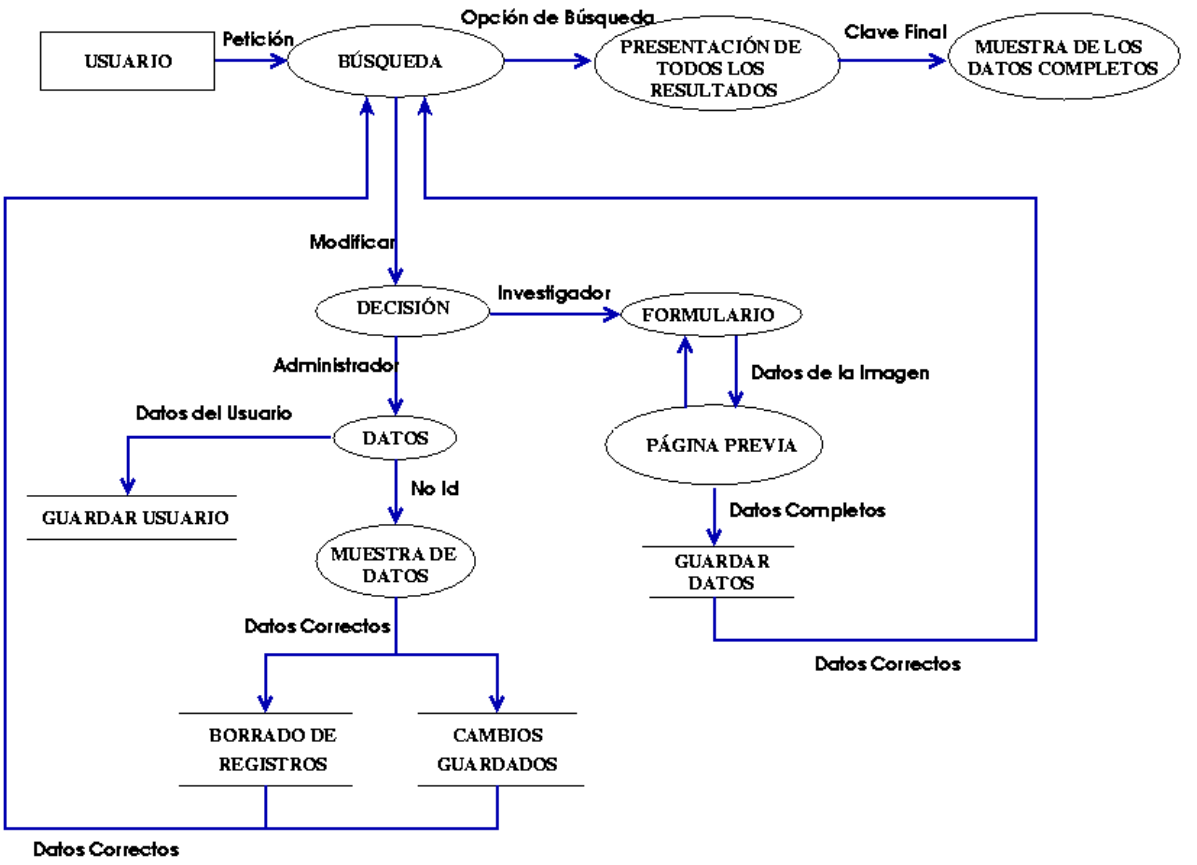


Fig. 4–6 Diagrama de Flujos del BIA – Nivel 0

4.3 Diseño de la interfaz en el sistema

En el desarrollo de los sistemas de software, se ha tenido la necesidad de adaptar su uso para personas ajenas al ambiente de la programación. Así surgen los métodos de interacción máquina–usuario cada vez más fáciles, rápidos y prácticos, basándose principalmente en ventanas. Pero el diseño de las interfaces se condiciona a un ancho de banda, la velocidad de descarga, etc., por lo que se inicia la construcción de las mismas a partir de elementos ligeros y de poco peso, de tal forma que el proceso de descarga y visualización de una página web no se convierta en algo estresante para el usuario.

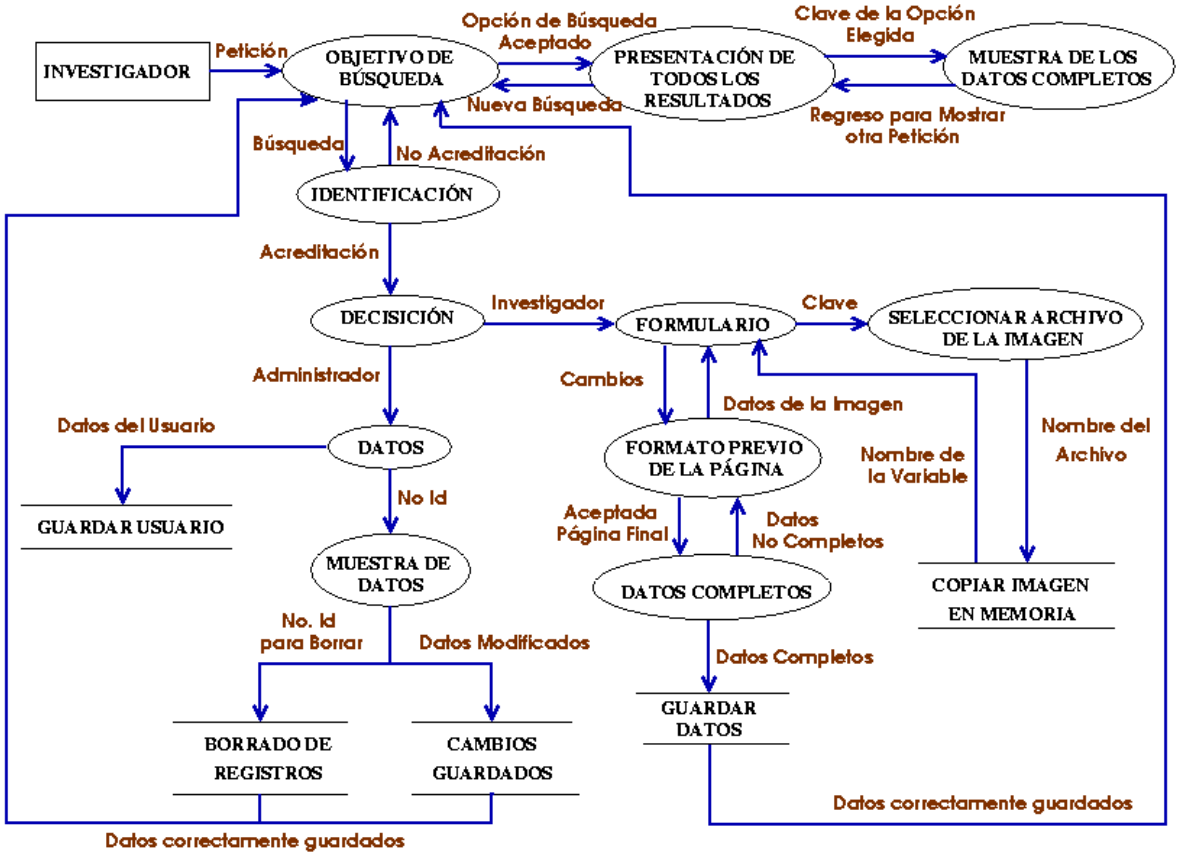


Fig. 4–7 Diagrama de Flujos del BIA – Nivel 1

Existen dos tipos de interfaces de datos: el *diseño de interfaz intermodular* y el *diseño de interfaz externo*. En el primero se trabajan los datos dentro de cada módulo y entre ellos, en el segundo los datos que necesita y que no puede generarlos por sí mismo, como pueden ser pulsos, emisión de luz, alguna conexión a un puerto paralelo o datos que el mismo usuario proporciona.

Para obtener la interfaz *intermodular* es muy útil el "análisis del sistema", ya que se observa el flujo de datos y la transformación de los datos en la transferencia entre los módulos, pero el usuario final directamente no tiene acceso a esta sección, por lo que el diseño es orientado a los ingenieros del sistema, no así en la interfaz externa que principalmente nos enfocaremos a continuación.

Para la interfaz *externa* se necesita cumplir con ciertos requisitos para mantener una excelente calidad, ya que en sí, esto es lo que evalúa el cliente. El usuario no se fija si un dato X se movió de un módulo a otro en vez de tomar otro camino o si el dato se convirtió en flotante u otra cuestión que no necesite, pero sí toma en cuenta lo rápido, fácil, útil y hasta "bonito" que le resulte el sistema.

El estudio debe comenzar analizando el tipo de usuario que utiliza el sistema, cómo maneja la información, qué es lo que espera del sistema, etc., además de investigar sus perfiles, edad, sexo, capacidades físicas, estudios, historial cultural o étnico, motivación, metas y personalidad.

En base al estudio anterior se comienza a estructurar un modelo de diseño que irá *refinando el cliente*, sin olvidar que este proceso se debe apegar a la estructura general del sistema, esto es, tiene que seguir la metodología tomada para todo el sistema (metodolología en cascada, de vida básico, ensamble de componentes, etc., para el BIA es el modelo en espiral).

Principalmente los aspectos que toma el diseño son el tiempo de respuesta, la ayuda y el manejo de la información.

El *tiempo de respuesta*, es aquel intervalo en el que el sistema responde después de que se suministra un dato de entrada, el cual debe tener un tiempo corto, ya que puede producir frustración y enojo en el usuario, aunque hay autores que afirman que el tiempo no debe ser tan corto porque si no, la interfaz podría apresurar al usuario.

La Fig. 4-9 muestra la primera interfaz del BIA para el usuario que sólo busca información, donde delimita el tema a buscar entre cinco opciones que se le presentan en la pantalla.

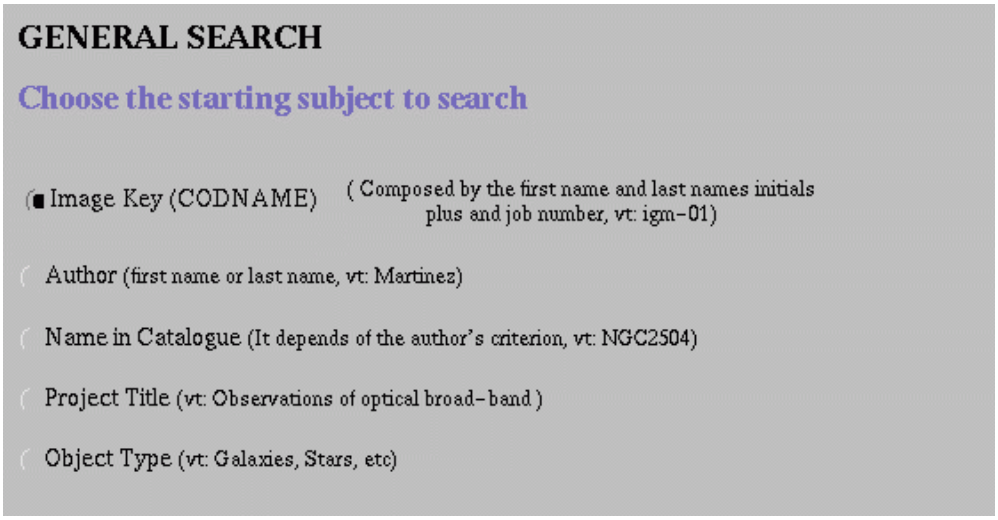


Fig. 4-9 Primera interacción con el usuario: selección del tema para la primera búsqueda.

Una ayuda que contenga al menos la mayoría de los problemas o el porqué de los errores que se generan por parte del sistema, ahorra mucho tiempo al usuario. La ayuda puede ser dentro del

sistema (notaciones que hace el programa al mismo tiempo que se ejecuta) o por medio de un manual (que es menos utilizado por la dificultad que conlleva).

Una gran parte de los problemas con el usuario se adquieren cuando se generan errores y se tiene una mínima o nula información sobre éstos, por lo tanto, para disminuirlos, el sistema debe contener la respuesta para la mayoría de las incógnitas, donde principalmente son los ingenieros de desarrollo que documentan mientras se programa el sistema.

Las respuestas generadas por el sistema deben ser claras, sin ambigüedades, constructivas para poder recuperarse del error y no emitir juicio alguno. La ayuda si es por parte del sistema (recomendado entre otras cosas por el ahorro de tiempo) debe ser visible y accesible.

El sistema también debe prever posibles errores que pueda cometer el usuario, aunque esto suene demasiado hipotético y muy ambiguo.

Para una mejor visualización de la ayuda, algunos puntos que se recomiendan seguir son:

a) ***Mostrar la información que sea relevante en el contexto actual***. El sistema no debe mostrar gráficas, datos o interfaces que no estén directamente relacionadas con la información pedida.

b) ***No abrumar al usuario con datos, utilizar un formato de visualización que le permita una rápida asimilación de la información***, al usuario no se le presentan grandes marcos y cuadros, sólo información puntual.

c) ***Usar etiquetas consistentes, abreviaciones estándar y colores predecibles***, la mayoría de las opciones de trabajo tienen un formato estándar y se recomienda tener un uso similar, por ejemplo, Ctrl+C significa copiar un objeto y Ctrl+V significa pegarlo.

d) ***Estudiar la "geografía" disponible en la pantalla***. Cuando hay múltiples ventanas, mínimo se tiene que ver una porción de éstas conteniendo los datos requeridos.

La **entrada de datos** para el BIA adquiere gran importancia en la sección de *agregar imágenes* y en la sección de *agregar nuevos usuarios*, ya que éstas son principalmente sus funciones en el sistema.

La Fig. 4-10 muestra parte del formulario para guardar imágenes y por lo tanto, las entradas de información al sistema.

Fig. 4-10 Entradas de información del BIA. Parte del formulario.

4.4 Diseño procedimental

Cuando el sistema se diseña se tienen que estructurar todos los diagramas de flujos de todas las secciones y como se mencionó anteriormente, pueden ser detallados hasta lo más mínimo, por ejemplo, llegar a describir una sola variable.

El *diseño procedimental* es la explicación necesaria para definir los detalles de los algoritmos de una manera natural y sin ambigüedades. Las construcciones son las mismas que lleva una estructura de un *diagrama de flujo*, pero se comienzan a manejar las condicionales y las repeticiones.

Las condiciones estructuradas limitan el diseño procedimental del software a un número predecible de operaciones, por lo que un menor número de construcciones hace que se facilite la implementación del código ante el programador.

Las estructuras que llevan los diagramas de construcción son: la *secuencia*, que es el paso entre los procesos; la *condición*, que permite decidir entre una u otra opción y la *repetición*, donde permite que una instrucción se repita las veces que uno haya establecido; la Fig. 4-11 muestra estas estructuras.

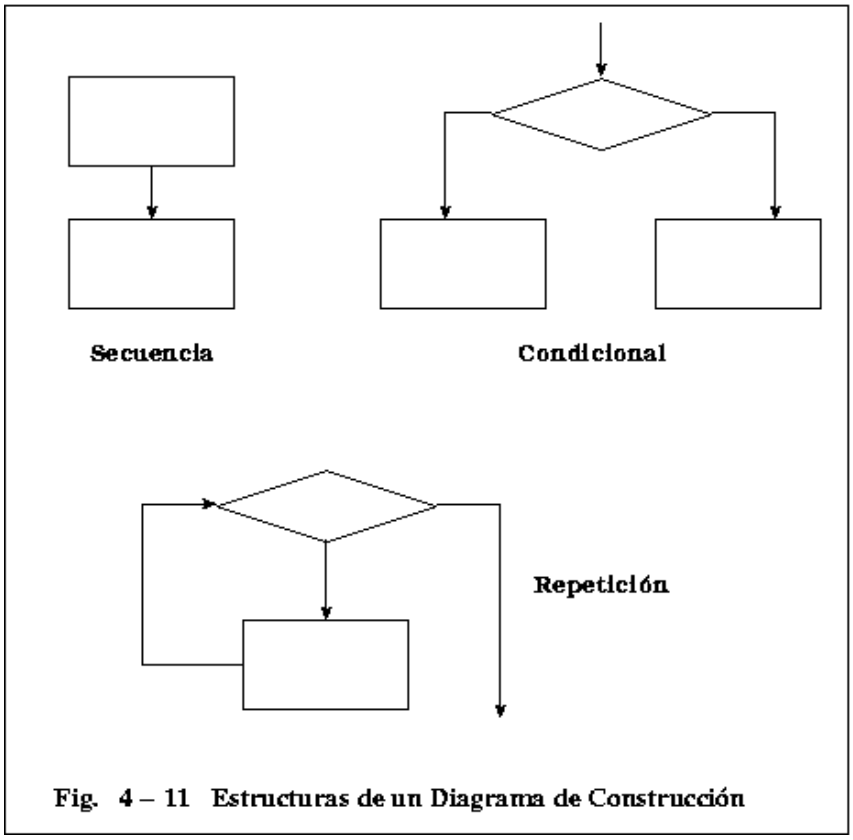


Fig. 4 – 11 Estructuras de un Diagrama de Construcción

También estos diagramas tienen niveles de acuerdo a su complejidad y para ejemplificar los diagramas de construcción de datos, la Fig. 4-12 y la Fig. 4-13 muestran los diagramas de la parte del sistema que sirve para agregar imágenes (tomando en cuenta el proceso de adjuntar archivos de manera general), y se puede observar que el último nivel ya se acerca a la programación en el lenguaje elegido.

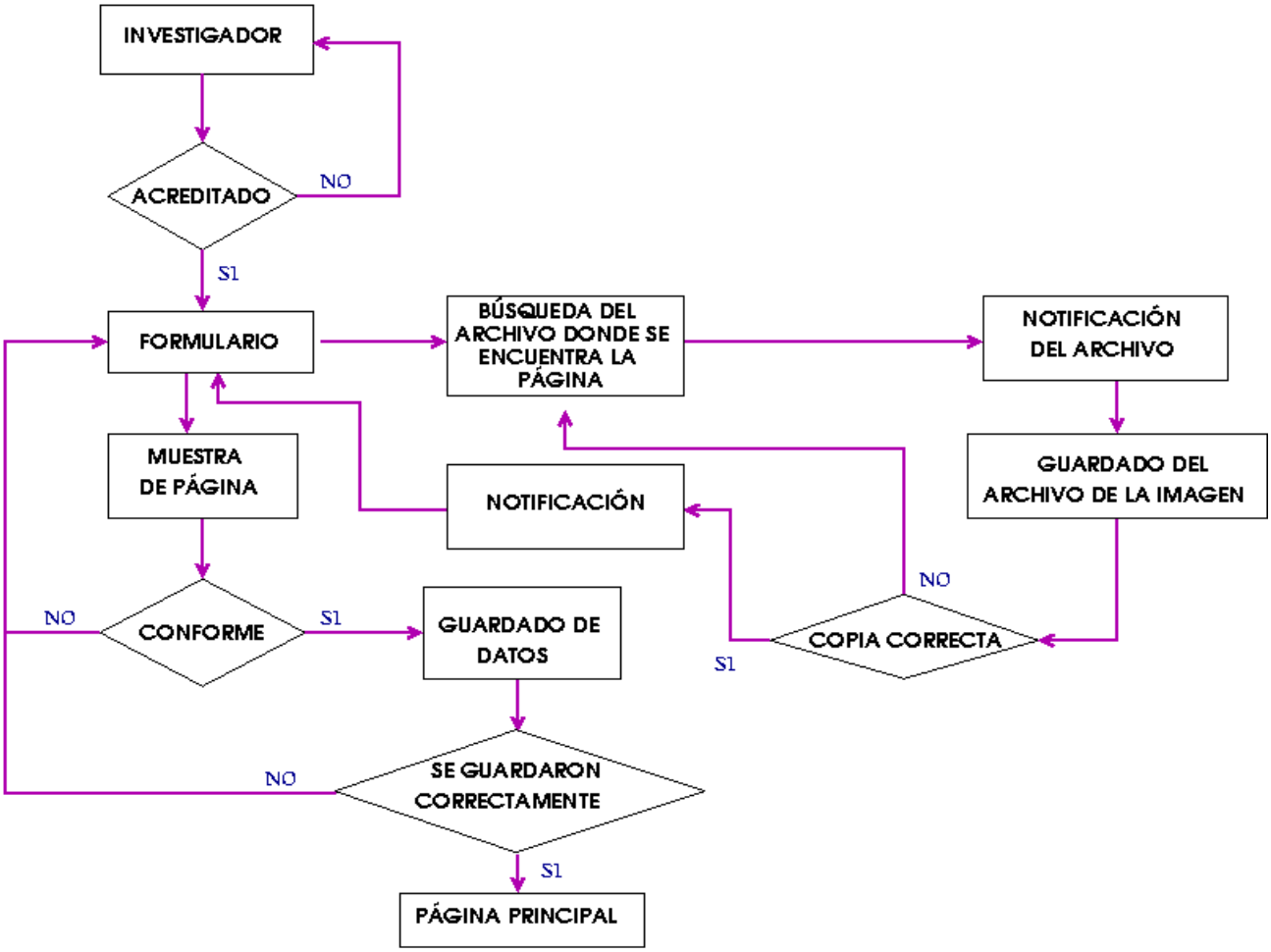


Fig. 4 – 12 Diagrama de Construcción "Agregar Imágenes" – Nivel 1

4.5 Diseño de la base de datos

El diseño de las bases de datos ha tomado una gran importancia a partir de las exigencias de los usuarios respecto a sus sistemas de información para que cada vez sean más flexibles, eficientes y accesibles. En los últimos años gracias al avance tecnológico y a la gran difusión de los sistemas de gestión para las bases de datos cuyos productos ahora ya pueden ser soportados por grandes y pequeños (supercomputadoras y computadoras personales), el camino se ha hecho más fácil, pero aún así, todavía existe un largo camino.

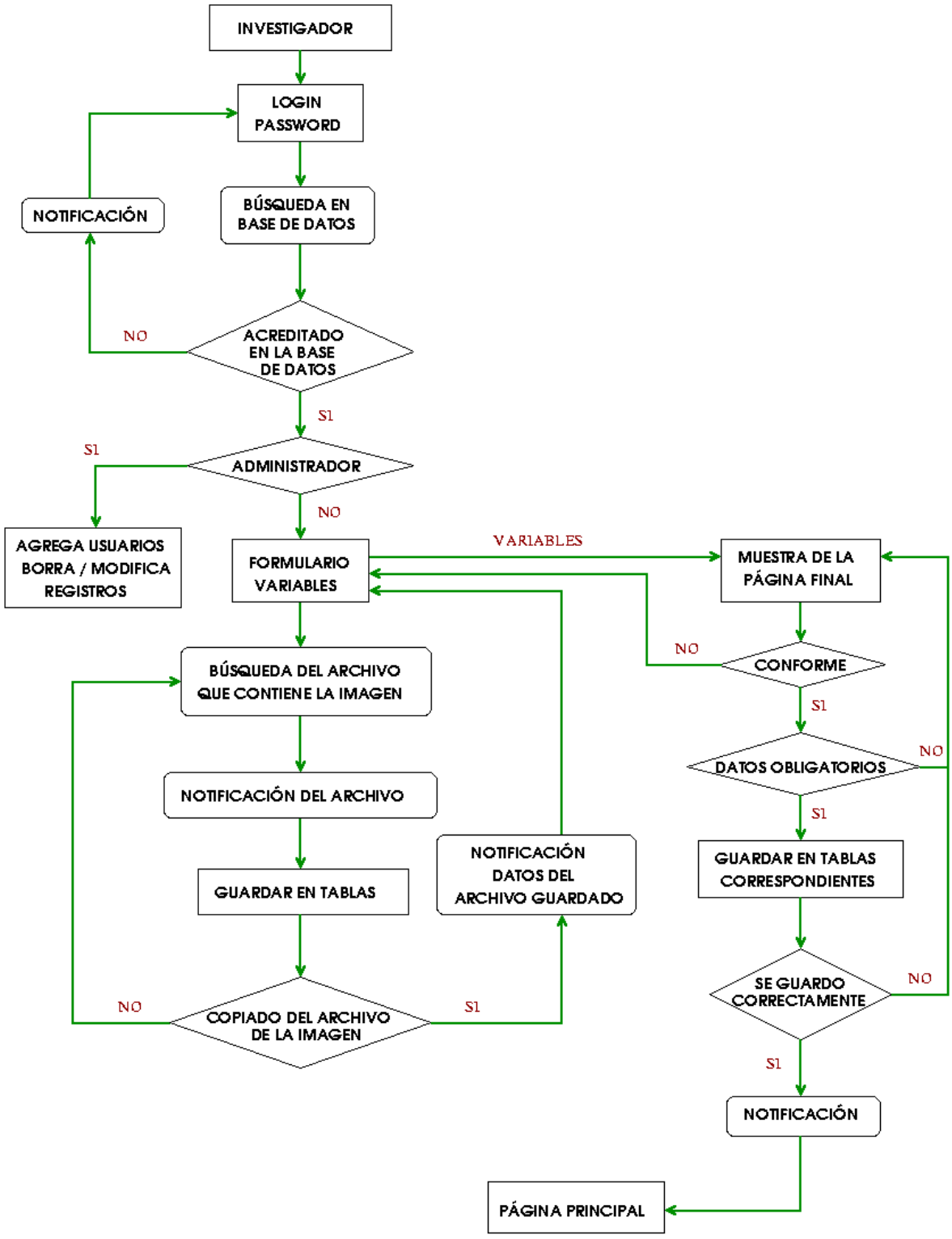


Fig. 4 – 13 Diagrama de Construcción " Guardar imágenes " – Nivel 2

Similar al diseño del proyecto, en las bases de datos también se siguen pasos ordenados y delimitados dentro del marco de una metodología y se manejan tres fases en este proceso:

- ***Diseño de la base de datos conceptual***. Es aquí donde se obtiene una representación de lo que el cliente necesita, objetivos primarios y secundarios, restricciones, manejo de la información, en general todo lo que conlleva al análisis del sistema. No se hace un gran hincapié a los tipos de perfiles de los usuarios ni a la plataforma de trabajo, ya que son sólo las delimitaciones generales.
- ***Diseño de la base de datos lógica***. En este bloque se transforma el esquema conceptual de la etapa anterior al modelo de datos que se apoya en el sistema gestor. Se puede adaptar la etapa (*diseño lógico*) a otros modelos de datos como el relacional, jerárquico o el Codasyl.
- ***Diseño de la base de datos física***. Aquí es donde se delimita el tipo de programación y se consigue la instrumentación necesaria.

4.5.1 Diseño de la base de datos conceptual

La importancia del modelo de datos en el diseño de bases, es la representación de un modelo conceptual que recoja la semántica del mundo real. Un modelo entidad–relacional sencillo pero completo que conlleve la participación tanto de los ingenieros del proyecto como los clientes es lo indispensable para el comienzo del proyecto.

Un *modelo de datos* es una serie de conceptos que pueden utilizarse para describir un conjunto de registros y las operaciones para manipularlos. Son la base conceptual para el diseño de aplicaciones que hacen un uso intensivo de los datos, así como la base para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información.

Estos modelos se dividen en *modelos conceptuales* y *modelos lógicos*. Los primeros manejan una mayor abstracción para presentar la realidad con respecto a los conceptuales que tienen una forma mucho más sencilla en la estructura física de las bases de datos.

Por lo tanto, en el diseño de bases de datos se utilizan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad y después, se transforma el esquema conceptual en un esquema lógico, esto ayuda a disminuir el grado de dificultad con el que se observa a grandes rasgos, todo el proyecto.

Los modelos de datos se clasifican en:

a) Modelos lógicos basados en objetos.
b) Modelos lógicos basados en registros.
c) Modelos físicos de datos.

En los **modelos lógicos basados en objetos** se tiene una representación de datos muy flexible y muy fácil de entender, nos muestra los datos como los captamos del mundo real. El más representativo es el *modelo entidad-relación,* que nos muestra los formatos por medio de entidades, es el más sencillo y más adelante se detalla.

Los **modelos lógicos basados en registros** son principalmente usados para describir los datos en los niveles conceptuales y físicos. Utilizan también registros, instancias y ligas, pero estos modelos basados en registros se usan para especificar una estructura lógica global de la base de datos y para proporcionar una descripción a un nivel más alto de la implementación.

Los tres principales modelos basados en registros de datos son el modelo *relacional*, el modelo de *red* y el modelo *jerárquico*, que también se retoman más adelante.

Los **modelos físicos de datos**, son utilizados para la descripción de los datos más bajos y sólo se manejan el modelo unificador y la memoria de elementos.

4.5.1.1 Modelo entidad-relación

El modelo entidad-relación forma parte del modelo de datos y es la estructura que delimita el enfoque global del sistema; es el más importante para la determinación de la estructura de las bases de datos.

Este modelo es usado como un soporte unificado de los datos que sintetiza el mundo exterior con un lenguaje natural: en relaciones y en entidades. Las *entidades* son objetos existentes que poseen ciertas características que los definen entre los demás, como son los *atributos y* las relaciones e indicadores de tipo de dato. Las *relaciones* son las definiciones que enlazan a las entidades.

El diagrama entidad-relación, está compuesto por entidades, que son representados por un *rectángulo*; los atributos de las entidades se representan por una *elipse*; la *etiqueta* dentro de un *rombo* nos indica la relación que existe entre las entidades destacando con líneas sus uniones y la *llave primaria* de una entidad es aquel atributo que se encuentra subrayado. La notación se muestra en la Fig. 4-14.

Existen diferentes tipos de relación en un diagrama entidad-relación, principalmente se manejan tres, que a continuación se muestran:

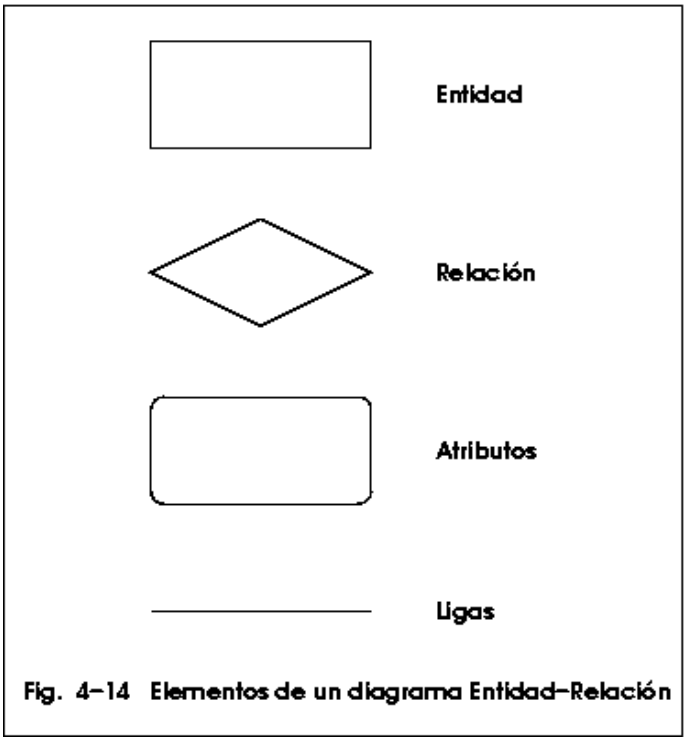
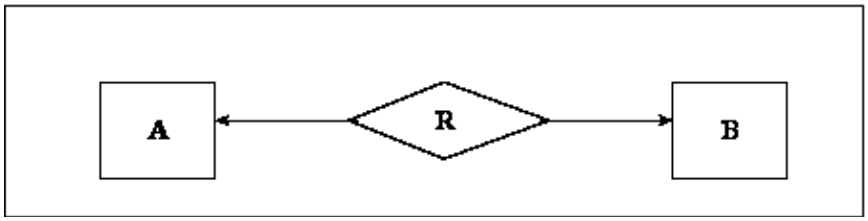


Fig. 4–14 Elementos de un diagrama Entidad–Relación

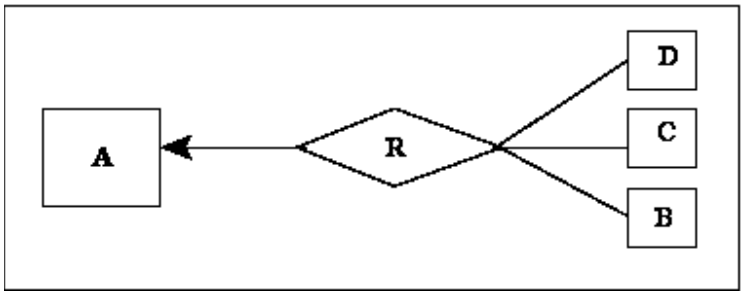
a) Relación uno a uno

Se presenta cuando existe una relación como su nombre lo indica uno a uno, denominado también relación de matrimonio. Una entidad del tipo A sólo se puede relacionar con una entidad del tipo B, y viceversa; lo anterior se muestra a continuación:



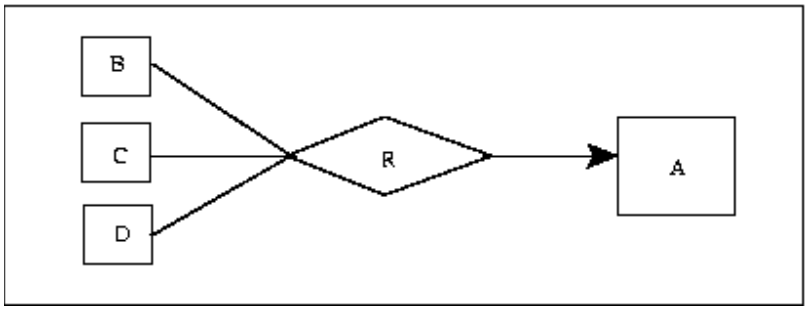
b) Relación uno a muchos

Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B y una entidad del tipo B únicamente puede estar relacionada con una entidad del tipo A. Su representación gráfica es la siguiente:



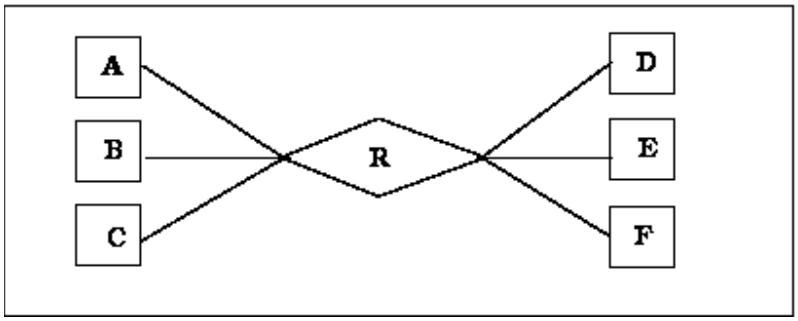
c) Relación muchos a uno.

Indica que una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A puede relacionarse con sólo una entidad del tipo B, gráficamente se ejemplifica así:



d) Relación muchas a muchas.

Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.



En el BIA las relaciones entre las tablas se manejan como relaciones Uno a Uno, ya que éstas sólo tienen comunicación entre sí, por una sola variable: la *llave primaria.*

La *llave primaria* es el atributo considerado clave para la identificación de los demás atributos que describen a la entidad; existen también las *llaves secundarias*, que se obtienen cuando existe más de un atributo que pueda identificarse como llave primaria, en este caso se selecciona la que

se considere más importante para ser la llave primaria.

Los diagramas entidad-relación donde se muestran los atributos, las relaciones, las ligas, llaves primarias y entidades para las tablas *General*, *Autores*, *Imágenes* y *Authent* del BIA se muestran en la Figs. 4-15 a, b, c y d respectivamente.

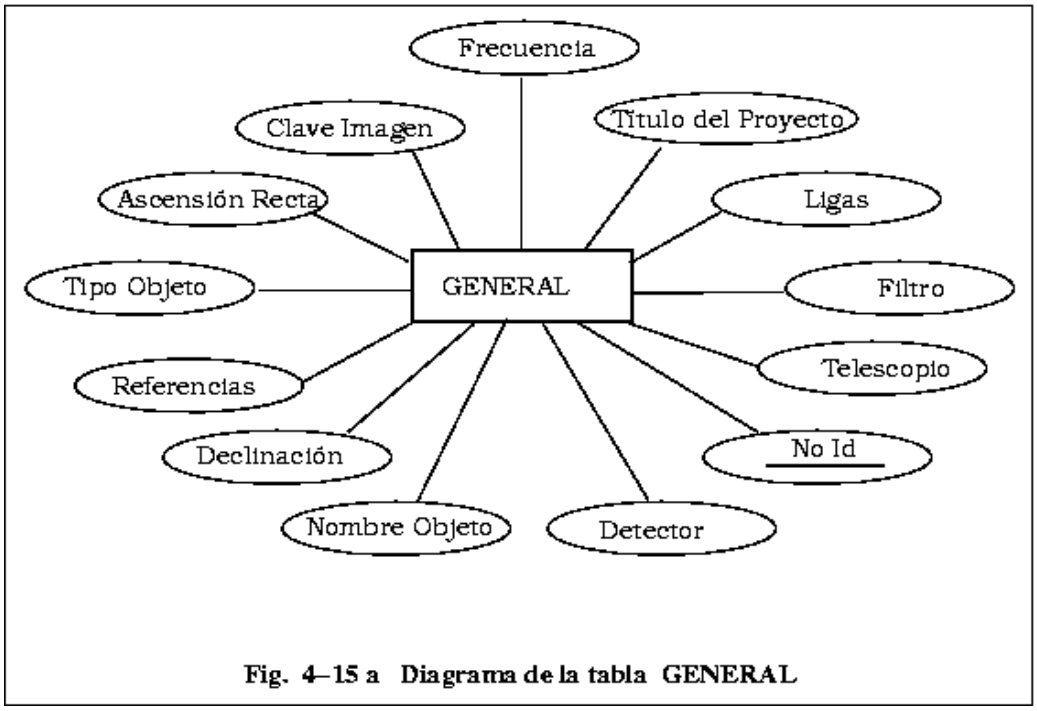


Fig. 4–15 a Diagrama de la tabla GENERAL

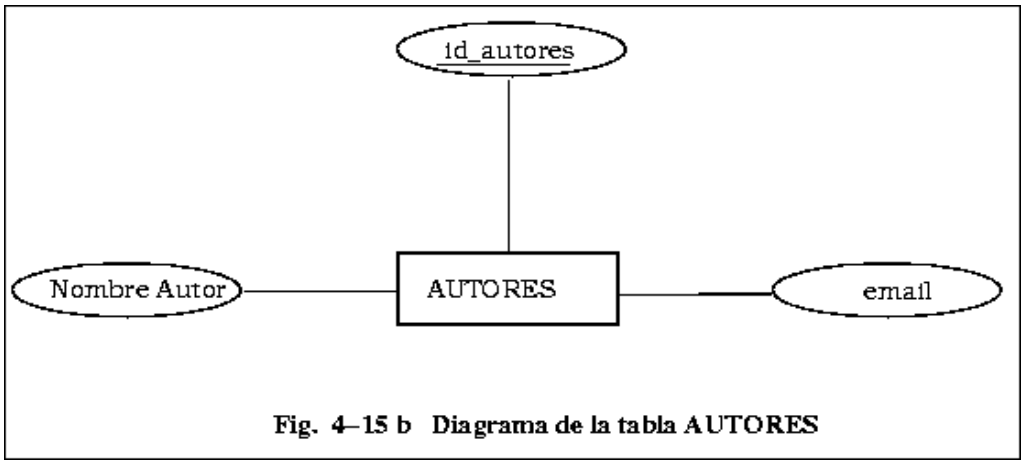


Fig. 4–15 b Diagrama de la tabla AUTORES

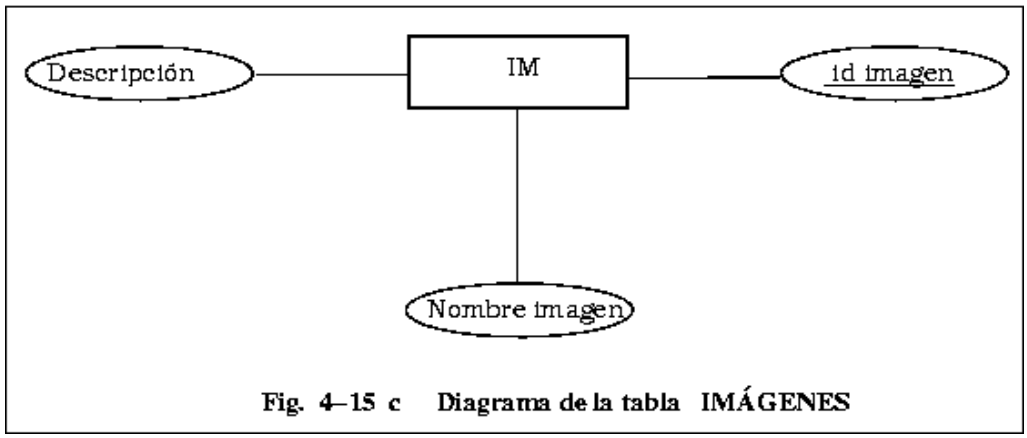


Fig. 4–15 c Diagrama de la tabla IMÁGENES

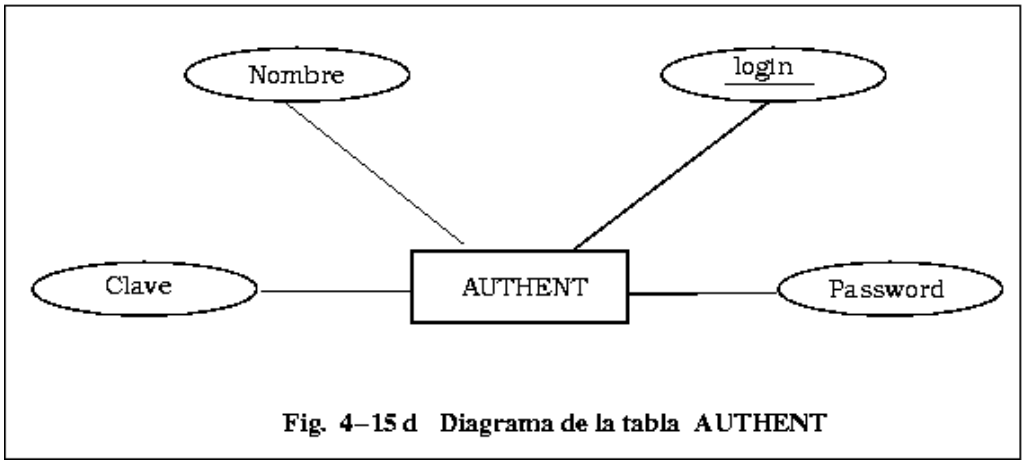


Fig. 4–15 d Diagrama de la tabla AUTHENT

En el proyecto se pueden delimitar tres partes importantes y casi independientes entre sí para la base de datos. La primera cuando se *busca información,* la segunda parte es el *agregar imágenes* y sus datos correspondientes y la tercera parte es la *administración,* en la cual solamente tiene acceso el administrador de la base.

El diagrama entidad-relación para la *búsqueda de imágenes* es una relación Uno a Uno, aunque se muestran tres tablas, solamente se relacionan entre éstas por un registro que identifica los datos de la imagen y su autor o autores, la Fig. 4-16 muestra este diagrama.

En el diagrama entidad-relación para *agregar imágenes* se tiene una relación Uno a Muchos, ya que para poder agregar datos a la base, sólo se necesita registrarse una vez, esto es, cuando un investigador quiera utilizar la base para añadir información, tiene su identificación y puede agregar cuantas imágenes quiera

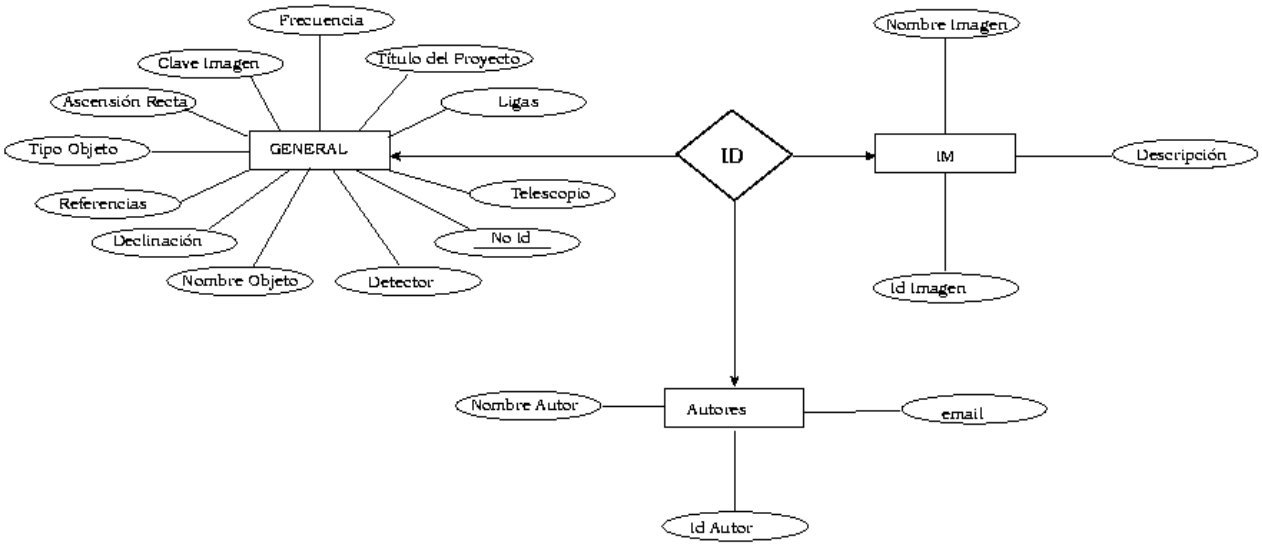


Fig. 4 – 16 Diagrama entidad–relación Búsqueda de imágenes

Aquí entra otro factor importante, ya que para guardar los archivos que contienen las imágenes se tiene un directorio que no pertenece a la estructura de las tablas de la base de datos, con esto nos referimos a que el nombre de la imagen se guarda en una columna de la tabla, pero el archivo que contiene la imagen se guarda en un directorio aledaño. Lo anterior se representa en la Fig. 4-17.

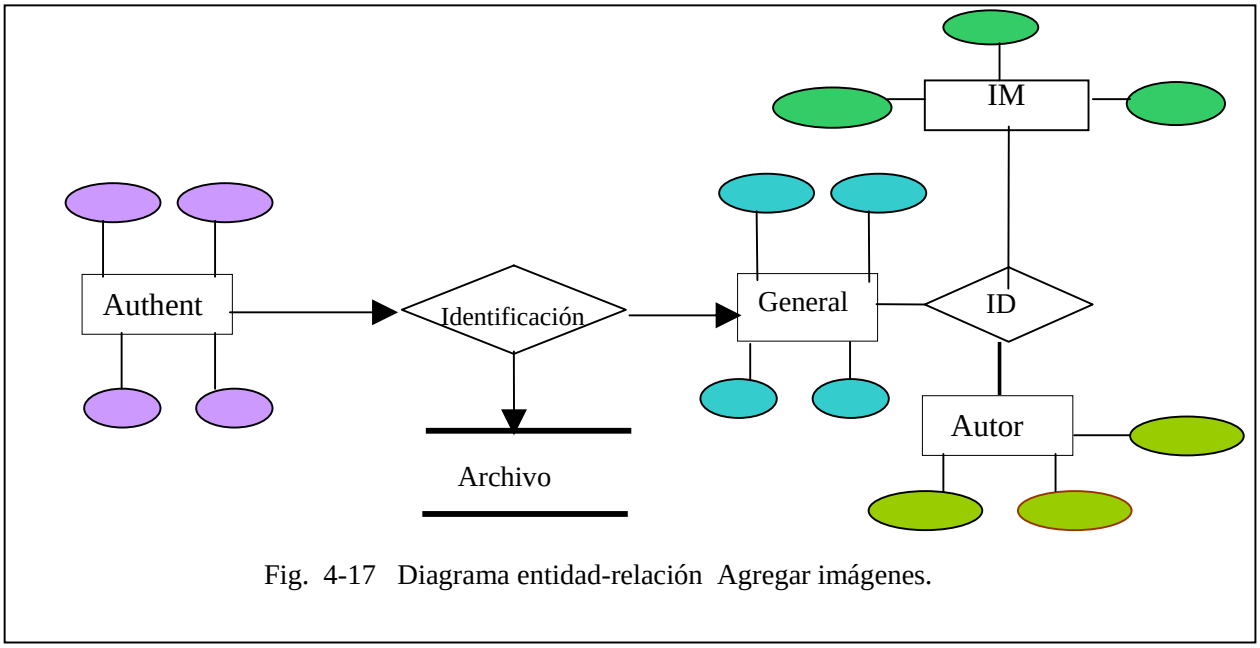


Fig. 4-17 Diagrama entidad-relación Agregar imágenes.

Para el diagrama *modificar datos* también se tiene una relación uno a muchos, ya que el administrador únicamente necesita una clave para entrar a modificar todos los registros en la base o para agregar algún usuario en ésta. Además que se tiene una retroalimentación en la tabla de Authent, ya que el mismo administrador que añade registros en la tabla, tiene que ser alguien que se encuentre dentro de ésta. La Fig. 4-18 representa este bloque.

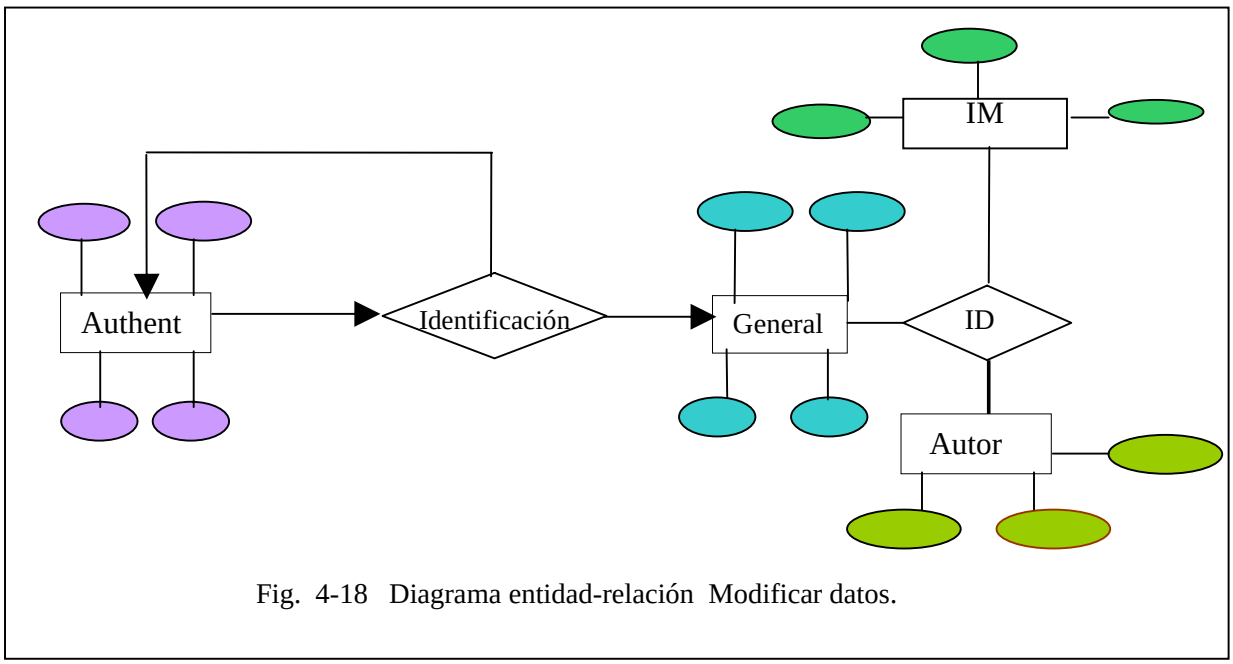


Fig. 4-18 Diagrama entidad-relación Modificar datos.

4.5.2 Diseño lógico de la base de datos

El diseño lógico sigue siendo independiente de los detalles de implementación, así como de la aplicación para programar o del sistema operativo a utilizar. La etapa se basa en un modelo y documentación de datos lógico y global como el diccionario de datos y el esquema relacional, que a su vez, representa la fuente de información para el siguiente paso: el diseño físico.

En resumen, el diseño lógico se interesa en el "qué" y el diseño físico en el "cómo", y se basa principalmente en un modelo formal que es: ***el modelo relacional***.

4.5.2.1 Modelo relacional

El modelo relacional está basado en la teoría de las relaciones donde los documentos se encuentran en forma de relaciones (tablas). De Miguel afirma, "Codd describe... la vista relacional de los datos... parece ser superior al modelo en grafos o en red... Proporciona

un medio para describir datos en su estructura natural únicamente, es decir, sin superponer ninguna estructura adicional con el propósito de su representación en la máquina."[25]

La estructura del modelo relacional se puede representar en la Fig. 4-19, donde se distinguen conjuntos de columnas denominadas *atributos* que representan la propiedad de la misma y tienen un nombre y un conjunto de filas denominadas *tuplas*, que son las ocurrencias de la relación. El número de filas de una relación se le llama *cardinalidad* y el número de columnas se les denomina *dominios*.

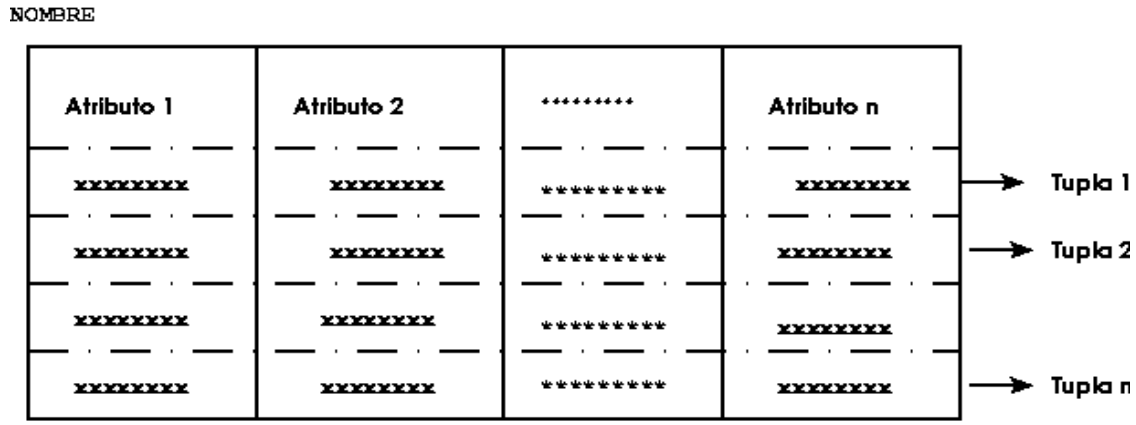


Fig. 4–19 Representación de una Relación

Una relación se puede representar en forma de una tabla, aunque tiene una serie de elementos que la distinguen de éstas:

- No puede haber filas duplicadas, es decir, todas las tuplas deben ser diferentes.
- No importa el orden de las filas.
- La tabla es plana, esto es, en el cruce de una fila y una columna debe tener un valor.

Para convertir el esquema conceptual que ya tenemos (modelos entidad-relación) a un modelo relacional, sin que se pierdan los datos y sus características principales, De Miguel propone tres puntos:

a) Todo tipo de entidad se convierte en una relación.
b) Todo tipo de interrelación N:M se transforma en una relación.
c) Todo tipo de interrelación 1:N se traduce en el fenómeno de propagación de clave o bien se crea una nueva relación.

Aunque a simple vista se observa que se pierden muchas características, no es así si se analiza datalladamente, De Miguel define "la pérdida de la semántica no implica, necesariamente, un

[25]De Miguel, Adoracion. Piattini, Mario. *Fundamentos y modelos de bases de datos.* Alfaomega. España. 1999. Pág. 123.

peligro para la integridad de la base de datos, ya que, si la transformación se ha realizado correctamente, se habrán definido las necesarias restricciones que aseguran la consistencia de los datos."[26]

Y una forma sencilla de representar los modelos relacionales son los diagramas de red.

4.5.2.2 Diagramas de red

Las redes constituyen una manera de representar la naturaleza más fácilmente, las relaciones entre éstas y sus datos. Las entidades se representan por medio de nodos de un *grafo* y sus asociaciones por medio de *arcos*. Esta representación no se basa en principio, a ninguna restricción ni al tipo ni al número de arcos, por lo que permite un modelo de estructuras tan complejas como se requiera.

En la era de los sesenta se desarrolla Integrated Data Store (IDS) de General Electric. Es dirigido por uno de los pioneros de los sistemas de bases de datos el Dr. Charles Bachmann. El proyecto es un nuevo tipo de sistemas de bases de datos llamado sistemas de red y su principal objetivo era satisfacer la necesidad de representación de relaciones entre los datos más complejos en comparación con que las que se podían modelar con los sistemas jerárquicos y en parte, para imponer un estándar de bases de datos.

Más adelante CODASYL, representado por gente del mundo empresarial y del gobierno de los Estados Unidos formaron un grupo denominado Data Base Task Group (DBTG), cuyo objetivo era definir las especificaciones estándar que permitieran la creación de bases de datos y el manejo de los datos. El DBTG presentó su informe final en 1971 y aunque éste no fue formalmente aceptado por American National Standards Institute (ANSI), muchos sistemas se desarrollaron siguiendo la propuesta del DBTG; estos sistemas son los que se conocen como sistemas de red, sistemas CODASYL o sistemas DBTG.

En general, una base de datos CODASYL es sinónimo de base de datos de red. El modelo de red intenta superar las deficiencias del enfoque jerárquico, permitiendo el tipo de relaciones de muchos a muchos.

La estructura de datos en red, es parecida a la estructura jerárquica, cada nodo puede tener varios hijos pero a diferencia de ésta, también puede tener varios padres.

En la Fig. 4-20 se muestra un diagrama en red, donde el nodo E tienen dos padres y los nodos C, F y G tienen sólo uno, cabe mencionar que lo anterior no se podría representar en un modelo jerárquico.

[26]De Miguel, Adoracion. Piattini, Mario. *Fundamentos y modelos de bases de datos.* Alfaomega. España. 1999. Pág. 268.

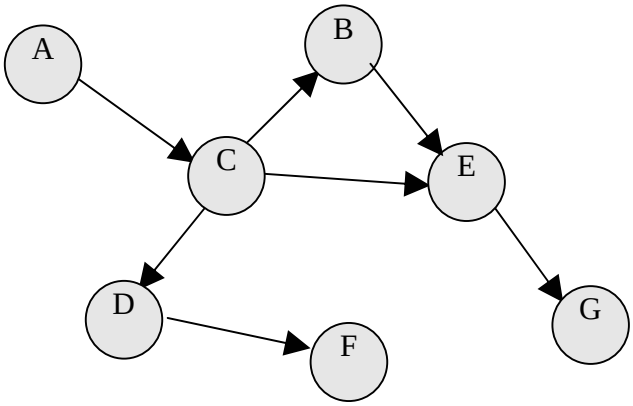


Fig. 4-20 Representación de grafos en un diagrama de red.

Un conjunto se constituye por dos tipos de registros que mantienen una relación de muchos a muchos. Para conseguir representar este tipo de relación es necesario que los dos tipos de registros estén interconectados por medio de un registro conector llamado *conjunto conector*. Los conjuntos poseen las siguientes características:

a) El registro padre se denominará propietario del conjunto, mientras que el registro hijo se denominará miembro.
b) Un conjunto estará formado en un registro propietario y uno o más registros miembros.
c) Una ocurrencia de conjuntos será una colección de registros, uno de ellos será el propietario y los otros sus miembros.
d) Todos los registros propietarios de ocurrencias del mismo tipo de conjunto deberán ser del mismo tipo de registro.
e) El tipo de registro propietario de un tipo de conjunto deberá ser distinto de los tipos del registro miembro.
f) Sólo se permitirá que un registro miembro aparezca una vez en las ocurrencias de conjuntos del mismo tipo.
g) Un registro miembro podrá asociarse con más de un propietario, es decir, puede pertenecer al mismo tiempo a dos o más tipos de conjuntos distintos. Esta situación se puede representar por medio de una estructura multianillo.
h) Se pueden definir niveles múltiples de jerarquías donde un tipo de registro puede ser miembro en un conjunto y al mismo tiempo propietario en otro conjunto diferente.

A continuación se representan los diagramas para los tres bloques que se definen en el proyecto, además de que se muestra la estructura final, llaves primarias, atributos, nombres de las tablas y las relaciones entre éstas.

La Fig. 4-21 muestra la parte *buscar imágenes,* aquí sólo se utilizan las tres tablas que contienen los datos de la imagen.

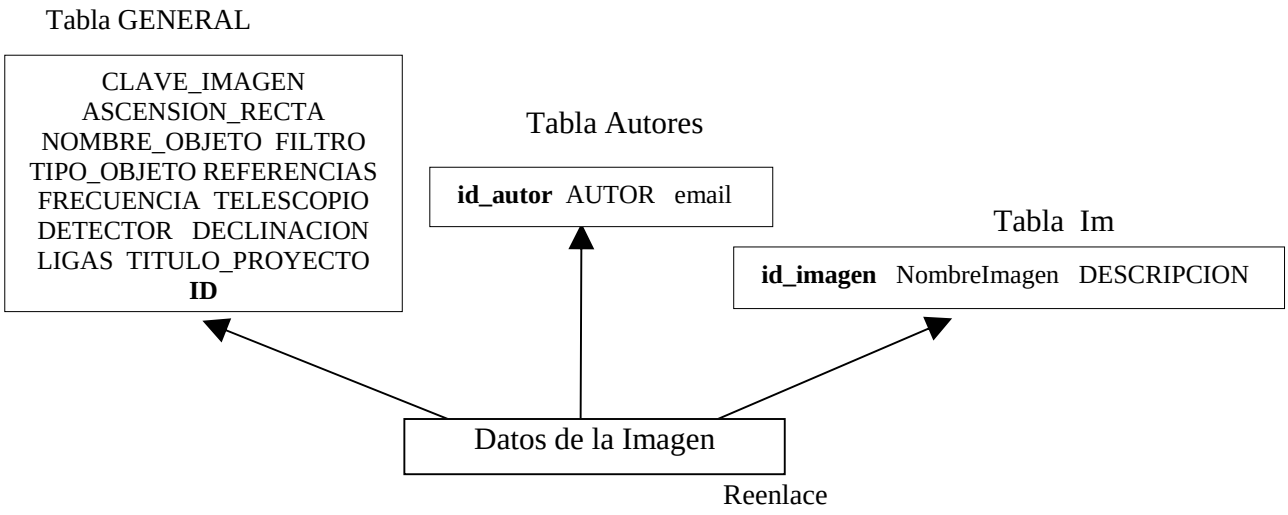


Fig. 4-21 Diagrama Búsqueda de imágenes.

Cabe destacar que las *tuplas* en la tabla de *autores* pueden o no ser diferentes, ya que dependen de la información de la imagen, no pueden reducirse los datos similares (principalmente en los nombres de los autores) ya que la variedad de datos que se registran varían considerablemente; por ejemplo, un investigador A puede trabajar una imagen solo o con un investigador B y su siguiente artículo realizarlo con otro investigador C, etc.

La Fig. 4-22 es para la parte *agregar datos* en la base, la cual ya utiliza un formato un poco más complejo por la relación que existe entre las cuatro tablas y el archivo anexo.

Y la última parte, la *administración de datos,* es donde se pueden modificar los registros y agregar a los nuevos usuarios, esta parte da un sistema de retroalimentación en la tabla de Authent, ya que la persona que puede agregar registros en esta tabla, tendrá que ser parte de la misma. La Fig. 4-23 nos muestra lo antes descrito.

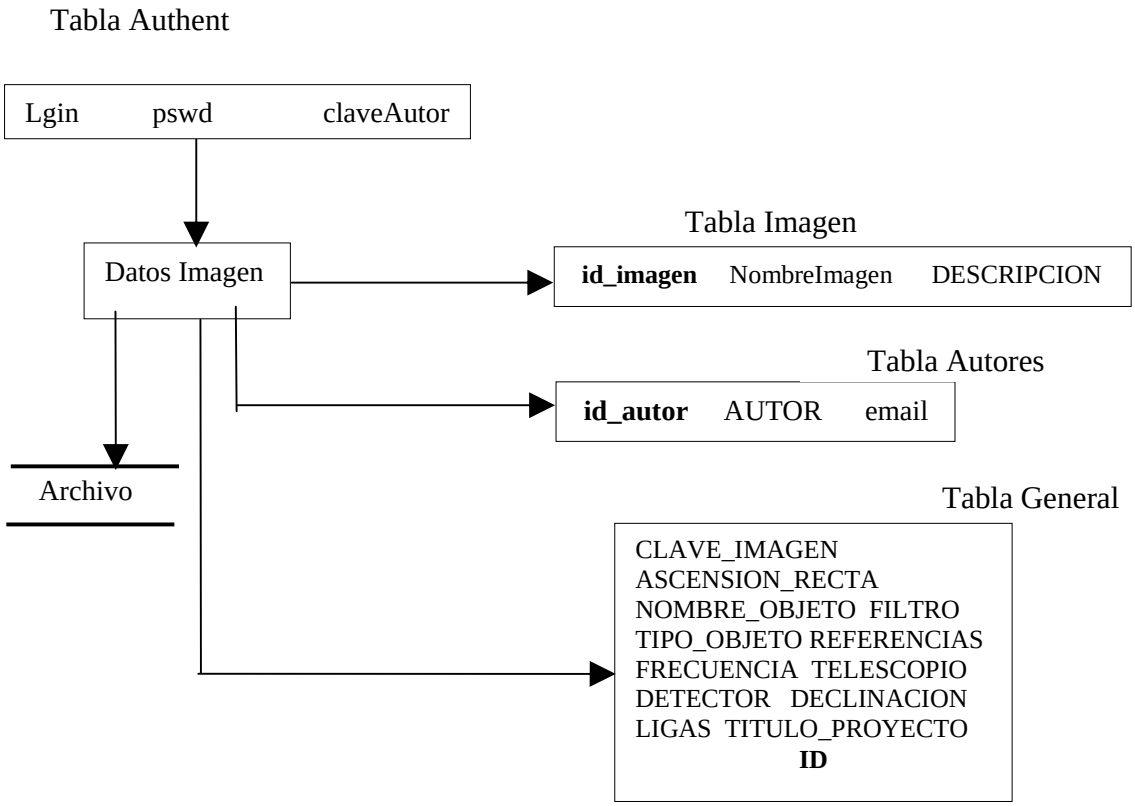


Fig. 4-22 Representación de la sección Agregar Datos.

4.5.2.3. Diagrama jerárquico

Como se mencionó, otra forma de representar los datos es de la forma jerárquica, que muestra una perspectiva del sistema en forma de niveles; donde se observa perfectamente la subordinación y el mando entre los módulos.

La estructura jerárquica o arborescente se compone por nodos que representan a las *entidades* y las *asociaciones* o relaciones entre dichas entidades, que se representan por arcos.

La estructura arborescente es una particularidad del modelo de red y tienen restricciones muy marcadas, esto es, siguen ciertas reglas para que se puedan definir como una estructura jerárquica. Las reglas se detallaron en el capítulo 2 y por lo tanto, sólo mostraremos el esquema jerárquico del proyecto BIA en la Fig. 4-24.

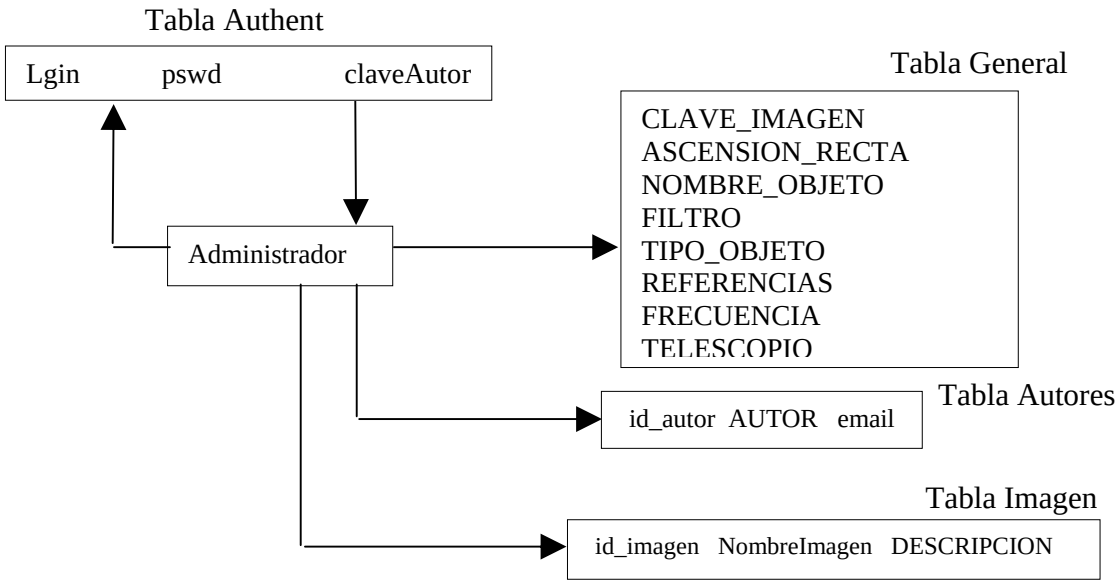


Fig. 4-23 Gráfica de la sección Administración de datos

4.5.3 Diseño físico de la base de datos

Todos los datos como caracteres, no tendrían importancia si no se disponen de medios necesarios para pasar a un "mundo real", por lo que se estructuran los lenguajes de manipulación, que nos van a permitir del mundo de caracteres recuperar los mismos datos para las estructuras reales, en otras palabras es la estructuración del *código* en el programa elegido.

El *diseño físico* se define como el proceso de producir una descripción de la implementación de las bases de datos, describe las relaciones base, el almacén de las estructuras y los métodos de acceso que son usados para acceder a los datos de una manera efectiva. Entre los objetivos principales se cuenta la disminución de los tiempos de respuesta, el minimizar el espacio de rendimiento, el evitar las reorganizaciones, proporcionar la máxima seguridad y optimizar el consumo de recursos.

Para comenzar el proceso físico se necesitan asimilar todas las relaciones que surgieron en el desarrollo lógico y principalmente la información se obteniene del *diccionario de datos*. Los datos que se identifican para cada relación son los siguientes:

- El nombre de la relación.
- Los atributos que soporta.
- La clave primaria, así como llaves alternativas y secundarias.

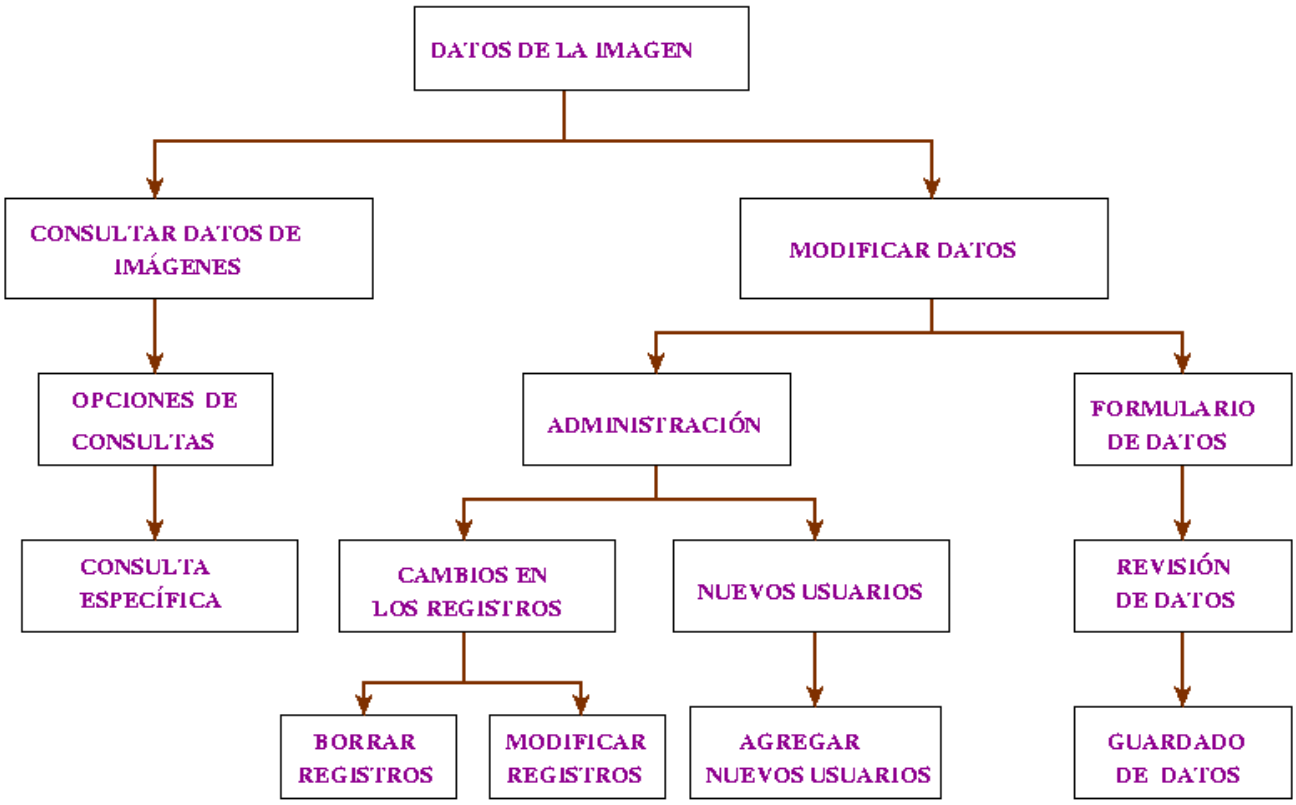


Fig. 4–24 Diagrama Jerárquico del BIA

Y para cada atributo:

- Los dominios, la consistencia de un tipo de dato, longitudes y otra característica en el dominio.
- Una opción para dejar de evaluar los atributos.
- Si el atributo puede ser nulo.
- Si el atributo es derivado y cómo sería computado.

Ya establecido el análisis, el diseño conceptual y el diseño lógico de las tablas para la base de datos, el siguiente paso es decidir cómo implementarlas y la decisión depende del sistema en el que se haya elegido para trabajar, ya que no existe un prototipo establecido, unos ofrecen más facilidades que otros para definir las bases de datos relacionales, la integración entre sus tablas, etc. Por lo tanto, lo que continúa es la programación de las tablas de la base de datos y de las interfaces para el sistema.

V

PROGRAMACIÓN DEL SISTEMA

5. Programación del sistema

Para la quinta etapa del proyecto en cualquier iteración (de acuerdo a la estructura en espiral), se comienza la programación en el lenguaje del sistema, la unión de prototipos y por lo tanto, se definen los lenguajes a utilizar, tanto para la programación de las interfaces como el lenguaje de la base de datos.

5.1 Delimitación de los lenguajes de programación

PHP es el acrónimo de Hipertext Preprocesor, fue creado en el año de 1994 por Rasmus Lerdorf y ha tenido grandes pasos desde entonces, su aplicación a páginas web al hacerlas más dinámicas y su acoplamiento a las bases de datos (en particular MySQL) ha contribuido a que más gente lo ocupe para implementar sus trabajos.

PHP es un lenguaje de programación designado específicamente para el web que puede satisfacer el manejar varias aplicaciones en un sitio, se le denomina como un *lenguaje del lado del servidor* porque se ejecuta en el servidor de web, justo antes de que se envíe la página a través de internet, las personas interesadas solamente reciben una página con el código HTML resultante de la ejecución en PHP, y como la página resultante contiene únicamente código en HTML, es compatible con cualquier navegador.

La ventaja de los códigos entre PHP y HTML es que podemos escribir las funciones HTML con cierto código PHP introducido en el mismo o viceversa, que se incluyen entre etiquetas especiales de comienzo y final que permiten entrar y salir de cualquier código.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, variantes Unix (HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS, entre otros. Al igual que soporta la mayoría de servidores web de hoy en día, como Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, etc.

En resumen, las principales ventajas que tiene PHP se listan a continuación:

- *Libre*, lo que conduce a que cualquier persona pueda tener acceso sin preocuparse de licencias ni de permisos.
- *Abierto*, cualquier persona puede ver la estructura y sus archivos.
- *Código fuente disponible*, puede ser modificado y adaptarse a las características de cada usuario y mejorarlo para la comunidad.
- Diseñado para la web: *compatible* con HTML.

- *Multiplataforma* hardware.
- *Multisistema* operativo, no tiene un sistema operativo predeterminado, por lo tanto, puede trabajar sobre Linux, Windows u otros.
- *Soporte* para varios servidores web.
- Soporte para la *mayoría* de las bases de datos.
- *Buena documentación* con muchos ejemplos.
- *Perfecta integración* de Apache-PHP-MySQL.
- Sintaxis *clara* y bien *definida*.
- Bastante *sencillo* de aprender y utilizar.
- *Modulable*.
- *Seguro.*
- *Amplia base de usuarios.*
- *No dependes* de un único proveedor de servicios.
- La mayoría de su sintaxis es *similar* a C, Java y Perl.
- Grandes *ventajas* para programadores de una interfaz.

La Fig. 5-1 muestra una perspectiva muy precisa de lo que hace PHP y porqué se le denomina lenguaje de servidor.

Son varios los lenguajes de programación que también podrían utilizarse en vez de PHP, el caso más factible es Java, el cual ofrece características similares, como un código abierto, software libre, manejo para bases de datos, facilidad de aprendizaje, etc., en la Tabla 5-a se realiza la comparación entre algunas de las funciones de PHP y Java.

Se elige PHP porque tiene un carácter más delimitado y más específico, con esto nos referimos a que es más orientado a las *personas como programadores* (pero no por esto deja de ser utilizado en grandes proyectos), lo anterior lo concluimos por su manejo de variables, cadenas en los códigos, etc. y por lo tanto, tiene algunas ventajas en comparación con la programación en Java.

Java se utiliza mayormente en aplicaciones más grandes de web porque tiene un mejor manejo en código orientado a objetos, encriptamiento de datos, etc., probablemente si hubiera más gente en el proyecto del BIA, hubiera sido conveniente tomar este lenguaje.

MySQL es un manejador de bases de datos muy potente. Se ha caracterizado por ser un poderoso sistema de gestión de bases de datos relacionales y su principal objetivo de diseño fue la velocidad, además que consume muy pocos recursos tanto de CPU como de memoria. Entre sus ventajas se enumeran:

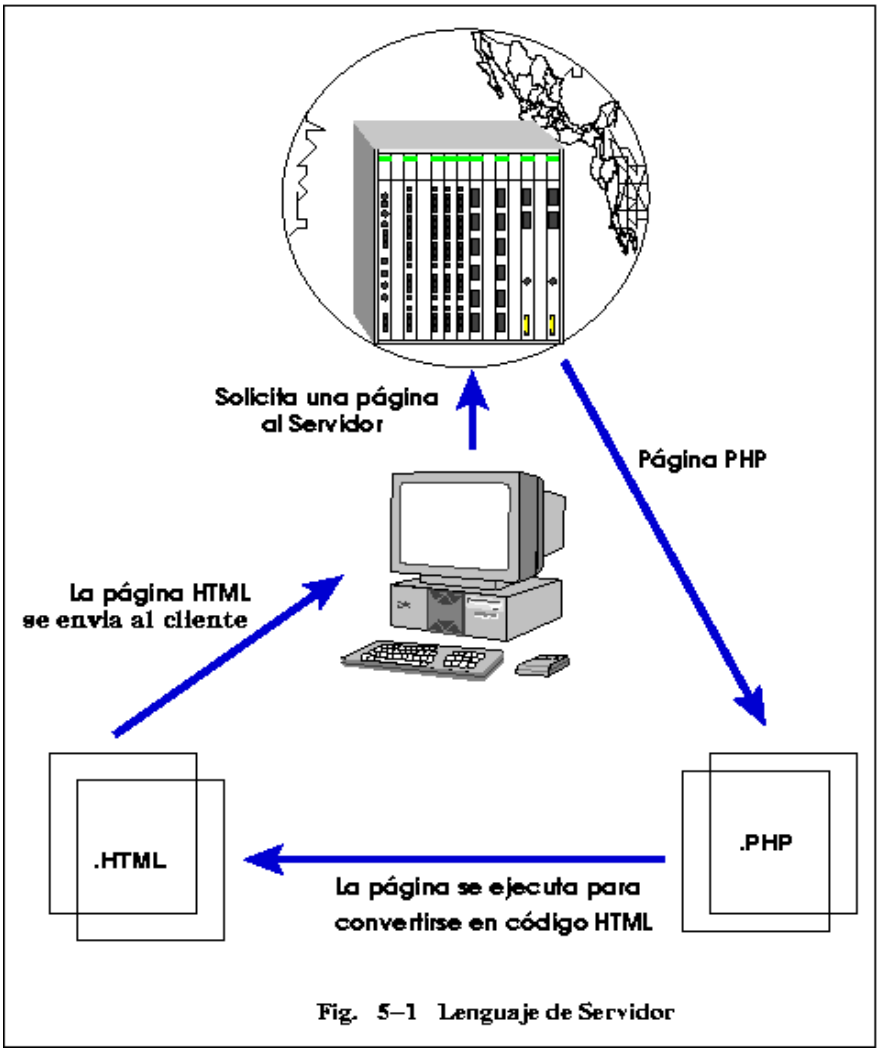


Fig. 5–1 Lenguaje de Servidor

1. *Mayor* rendimiento.
2. *Mayor velocidad* tanto al conectar con el servidor como al servir selects y demás funciones.
3. *Mejores utilidades* de administración (backup, recuperación de errores, etc.).
4. *Registros seguros*, ya que no pierde información ni corrompe los datos.
5. *Mejor integración con PHP*.
6. *No hay límites en el tamaño de los registros.*
7. *Facilidad* de uso.
8. *Capacidad de gestión de lenguajes de consulta*, MySQL está principalmente hecho para desarrollarse en plataformas libres, pero se puede acceder empleando aplicaciones que admitan Open Database Connectivity, un protocolo de comunicación de bases de datos desarrollada por Microsoft.
9. Mayor *control* de acceso.
10. Permite *modificar* o *añadir* campos a una tabla muy rápido con sólo algunos comandos.

Tabla 5-a Comparación de algunos formatos de código entre PHP y Java.

1	php	<?php echo "Hola Mundo<p>"; ?>
	java	<% out.print("Hola Mundo<p>"); %>
		Los dos son similares
2	php	echo "Son multiples lineas, las siguientes lineas tendran una salida correcta";
	java	Java no lo soporta. No soporta las cadenas de múltiples líneas.
		PHP tiene ventaja
3	php	<?php echo $_SERVER["HTTP_USER_AGENT"];?>
	java	<%=request.getHeader("User-Agent"); %>
		Java recupera las clase abstractas con una mejor organización
4	php	<?php phpinfo(); ?>
	java	Java no maneja variables globales.
		No teniendo variables globales ayuda en el mantenimiento y mejoramiento del software.
5	php	if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) { echo "You are using Internet Explorer "; } ?>
	java	<% if (request.getHeader("User-Agent").indexOf("MSIE")>-1) { out.print("You are using Internet Explorer "); } %>
		Al parecer tienen iguales condiciones
6	php	<?php if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) { ?> strstr must have returned true You are using Internet Explorer <?php } else { ?> strstr must have returned false You are not using Internet Explorer <?php } ?>
	java	<% if (request.getHeader("User-Agent").indexOf("MSIE")>=0) { %> indexOf must have returned >=0 You are using Internet Explorer <% } else { %> indexOf must have returned -1 You are not using Internet Explorer <% } %>
		Los dos tienen características similares.
7	php	<?php echo $_POST["name"]; ?>
	java	<% out.print(request.getParameter("name")); %>
		Tienen similares características para este fin.

Apache es uno de los más importantes servidores de web comparable a los comerciales pero gratuito, por lo que se puede tener acceso a su código fuente. Es gestionado por un grupo denominado Apache Group, además de los voluntarios en todo el mundo que aportan ideas para planearlo y desarrollar mayor documentación.

Originalmente se realizó para sistemas operativos Unix, pero actualmente puede correr en una gran cantidad de plataformas: aix, aux, beos, bs2000-osd, bsdi, cygwin, darwin, dgux, digitalunix, freebsd, hpux, irix, linux, macosx, macosxserver, netbsd, netware, openbsd, os2, os390, osf1, qnx, reliantunix, rhapsody, sinix, solaris, sunos, unixware, win32, windows XP, windows server, etc.

El acoplamiento **Apache-PHP-MySQL** ha resultado muy confiable, por lo que la mayoría de los sistemas tienen esta configuración. La gran adaptabilidad, la integridad, la documentación, el rendimiento, entre otras cosas, han hecho que el sistema tenga un soporte cada vez más fuerte, además que los códigos fuentes se tienen a la mano y son gratuitos, por lo que se pueden instalar en cualquier máquina sin tener ningún problema de licencias.

El Software Libr**e** nace en la década de los 80's y comienza por la necesidad de tomar otra opción al llamado software “comercial” o “propietario”, este tipo de software permite al usuario tener los códigos fuentes que le permiten usar, modificar y redistribuir el sistema con las restricciones mínimas para garantizar esa misma libertad a otras personas, es decir, prohíbe que un usuario le restrinja la libertad a otro.

Al tener cualquier persona acceso al código, se evita el problema de copyright y de legalidad del uso del software, ya que no existe un dueño, ni monopolios que puedan pelear ventajas o recursos de lo que se realiza con el sistema.

También se pueden solucionar problemas como la piratería del software, que causa grandes estragos a empresas privadas como Microsoft.

Uno de los principales miembros del software libre es Linux, creado inicialmente por Linus Torvalds, pero que actualmente tiene soporte de millones de personas repartidas en todo el mundo.

Linux destaca por su estabilidad (sistema que reporta menos fallas), lo que conlleva a tener una mejor seguridad, un mejor manejo de datos, menor tiempo y por lo tanto, una reducción considerable en los costos del sistema.

Actualmente muchos usuarios a nivel mundial están cambiando el sistema con el que trabajan por software libre, por ejemplo, universidades, redes gubernamentales y militares, empresas públicas y privadas, ya que este sistema es capaz de cumplir desde tareas muy complejas y de gran escala, hasta tareas de escritorio.

Tiene también grandes ventajas para el manejo de internet y en redes locales tiene muy buenos sistemas de seguridad. Además que se pueden poner estaciones Linux conectadas con estaciones de otros sistemas operativos (UNIXes, Windows NT, etc.).

La programación también es una de las tareas en las que se destaca Linux. La popularidad que tiene entre los programadores e ingenieros se debe a las características avanzadas de este sistema, el cual permite desarrollar un software más complejo.

Por todo lo anterior, delimitamos que la plataforma con la que se trabaja el Banco de Imágenes Astronómicas es ***Linux***, se escoge el servidor ***Apache***, el lenguaje de programación ***PHP*** y para el gestor de la base de datos ***MySQL***.

5.2 Programación de la interfaz

El guardar información en el banco por una persona específica podría requerir mucho tiempo y no sería tan didáctico, si se toma en cuenta que cada investigador realiza sus proyectos en diferentes circunstancias y principalmente en diversos lugares del país, por ejemplo, Ciudad Universitaria DF; San Pedro Mártir, BCN; Tonanzintla, Puebla; Morelia, Mich., y otras universidades e instituciones relacionadas con el IAUNAM, por lo que se requirió establecer una sección en la que cualquier persona acreditada pueda agregar información al banco de una manera accesible y muy sencilla para manejar: a*gregar imágenes.*

Para la realización de las interfaces del BIA, el intercalar PHP con HTML y viceversa de una manera muy sencilla, conlleva grandes ventajas en la realización de páginas dinámicas ya que facilita en gran medida la programación, se observa principalmente porque todas las funciones explícitas de un lenguaje se pueden adaptar al otro y las variables que no se pueden manejar con éste, se pueden manejar con las características del otro lenguaje, etc.

La sección de *agregar imágenes* principalmente se constituye por un formulario, previo un login y un password (que son dados por el administrador y constituye una parte de otra de las secciones del sistema); el formulario se divide en un adjuntador de archivos (donde su función es copiar el archivo de la imagen al espacio asignado al sistema), el llenado de los datos requeridos y de una vista previa de la página final.

En base al análisis realizado en los capítulos anteriores comenzamos a *estructurar* la programación del sistema, ya que los diagramas de construcción son sólo un paso antes para la programación en el código elegido. En el capítulo 4 se mostraron los diagramas de construcción para la sección g*uardar imágenes*, por lo que a continuación se detallan las partes de su código. La Fig. 5-2 es el diagrama de construcción de *guardar imágenes* nivel ***3***, *y* muestra las tres divisiones que se hace para describir a continuación, la estructura del código.

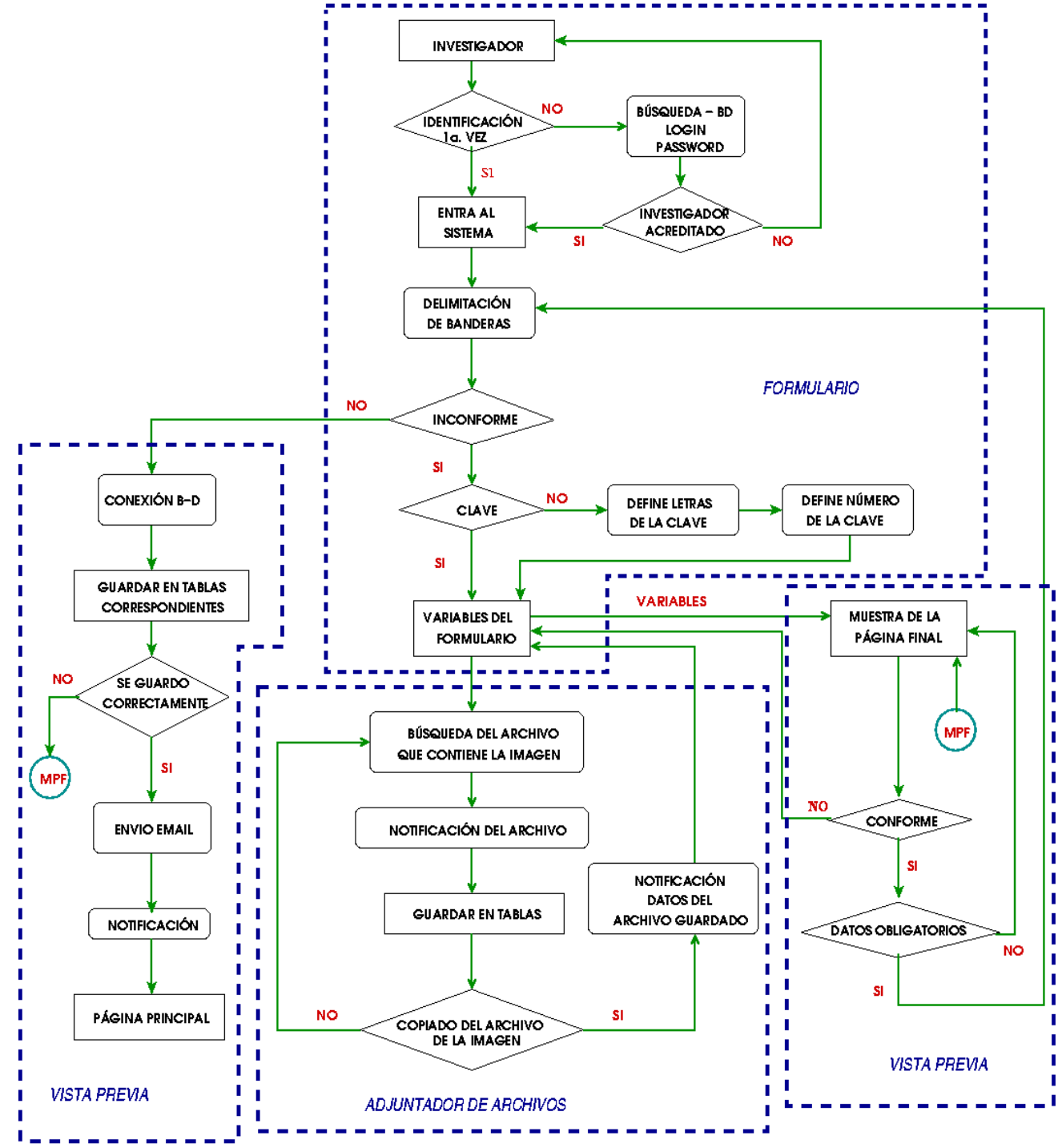


Fig. 5 – 2 Diagrama de Construcción " Guardar Imágenes " – Nivel 3

5.2.1 Formulario

Como se mencionó anteriormente existen subsecciones en la parte de guardar la información y la primera que detallaremos es el formulario.

El formulario es la presentación de los campos para la entrada de la información, se compone en total de 15 registros y una liga para el adjuntador de archivos. La página del formulario se muestra en la Fig. A-6, que se encuentra en el Apéndice.

Cuando se cargan diferentes páginas en red, las variables que se manejan dentro de ellas se pierden, a menos que se pasen por la URL, se guarden en *cookies* o en un formulario, pero principalmente tienen las desventajas de que algunos servidores no las acepten o que no pueden manejarse entre varios scripts, por lo que PHP maneja las llamadas *sesiones*, que nos permiten declarar ciertas variables que pueden ser reutilizadas tantas veces se requieran dentro de una misma sesión compuesta por uno o varios scripts.

Cuando se manejan sesiones, lo *primero* que se debe declarar es la sesión, ya sea para iniciarlas o para hacer referencia de que el script pertenece a ésta. Cuando se menciona *primero*, es que la declaración debe estar obligatoriamente en la línea 1 del script, ya que de lo contrario marcará error en memoria cache:

" Cannot send session cache limiter - headers already sent (output started at /export/local/httpd/htdocs/php/sesiones/formularioSess3b.php:15) in /export/local/httpd/htdocs/php/sesiones/formularioSess3b.php on line16".

El registro de sesión se realiza de la siguiente manera:

```
 1  <?php
 2  // Registro sesion
 3  session_register("variables_formulario");
 4  ?>
 5
 6 <HTML>
 7
 8      <HEAD>
 9        <TITLE> Sesion3  formulario </TITLE>
10      </HEAD>
11
12   <BODY>
```

Para intercalar PHP y HTML sólo hace falta delimitar entre los símbolos **<*?php*** cuando inicia este lenguaje y ***?***> cuando se termina.

En las partes iniciales de cualquier script se deben realizar los llamados a páginas que conlleven funciones o subrutinas ocupadas en la ejecución del programa, en nuestro caso *funcion.php* tiene estas subrutinas, por ejemplo, la rutina de conexión a la base de datos.

```
<?php
include ("../funcion.php");

$link=conexion_bd();
```

Para entrar al formulario se debe tener un login y password como *investigador* (ya que también puede ser administrador), si ambos son correctos, se manda llamar a la función que identifica al usuario (esto sólo ocurre si es la primera vez que entra a la página, ya que en caso contrario, el ciclo entra directo).

```
if(!$variables_formulario[0])
{
 certificacion($password,$login);
 $variables_formulario[15];
}
```

La mayoría de las variables del formulario se manejan por medio de sesiones, pero existen variables como *$inconforme* o *$conforme* que se manejan por medio de tablas en HTML. La variable $inconforme significa que ***no*** se van a guardar los datos y que, independientemente si la página anterior es *identificación* o *página previa* (los términos se definen en el diccionario de datos) se presenta el formulario, la diferencia entre estas páginas es que llevan variables de sesión vacías o no. Por lo tanto, $conforme es exclusivamente para guardar los registros.

```
if($inconforme)
   {
  echo "<br><br><center> Welcome, your  login is  <b> ";
```

Como se marca en el diagrama de construcción al entrar al formulario se identifica si existe la clave del proyecto, si no existe hay que generarla, y aquí la clave se define como $variables_formulario[0].

```
if(!$variables_formulario[0])
  {
```

Al no identificar la clave, se definen primero las letras que la conforman, que se realiza con un llamado a la tabla *Autores* (ya que las letras son guardadas en un registro de ésta); *$login*, es la variable que se transfiere cuando identifica si es investigador o administrador y sirve como llave para encontrarlas.

```
    $llamadoLogin=mysql_query("Select claveAutor from Authent where
lgin='$login'");
```

```
$rows = mysql_fetch_array($llamadoLogin);
$clave= $rows["claveAutor"];
$variables_formulario[16]=$clave;
```

Ahora se define el número que le corresponde de acuerdo a las investigaciones que se han guardado en la base de datos y la *clave* es ahora la que sirve como llave. Al tener los dos datos sólo se realiza una unión de cadenas y el resultado es la *clave particular* del proyecto.

```
    $ClaveAutor= mysql_query("Select lgin, clave_imagen from
Authent,GENERAL where (claveAutor like '$variables_formulario[16]%')
and (clave_imagen like '$variables_formulario[16]%') ");
    $rowsAut = mysql_fetch_array($ClaveAutor);

      // Defino mis dos variables, la Clave siguiente:
      // $variables_formulario[0]
      // el login: variables_formulario[17].
         $variables_formulario[17]= $rowsAut['lgin'];
         $numProy= mysql_num_rows($ClaveAutor);


         $ProyAumentado=$numProy+1;
         $proyfin=$variables_formulario[16]."-0".$ProyAumentado;
         $variables_formulario[0]=$proyfin;
     }
```

Teniendo la clave, se presenta el formulario que se maneja como formato en HTML, por lo que se tiene que dar de alta el tipo de tabla, bordes, alineación, etc., además de manejar colores y tamaños de letras.

Para los formularios en HTML existen varios tipos de entradas que se observan en las variables del sistema, como *text area* que demarca una entrada de texto pero con un largo y un ancho; *Password* que indica que el campo a introducir será una palabra de acceso restringido por lo que muestra asteriscos en lugar de letras escritas; *Checkbox* donde el campo se elige marcando una entre varias opciones de casillas cuadradas y opciones como las siguientes:

- *type*, indica el tipo de variable a introducir.
- *text*, indica que el campo a introducir será un texto.
- *maxlenght* seguido de un valor, limita el número máximo de caracteres a introducir en ese campo.
- *size* seguido de un valor, limita el número de caracteres a mostrar en pantalla.
- *value*, indica que no hay valor inicial del campo, entre comillas se indica el valor de la casilla.
- *checked*, la casilla aparece marcada por omisión.
- *radio*, el campo se elige marcando una casilla circular entre varias opciones.
- *image*, el campo contendrá el valor en coordenadas del punto de la imagen que haya delimitado. Pero debe llevar obligatoriamente el atributo:
  src, y entre comillas el nombre del archivo de imagen.
- *hidden*, el visitante no puede modificar su valor ya que no está visible. Se manda siempre junto al atributo *value*, seguido de su valor entre comillas.

Las variables de entrada del sistema tienen varias características de las escritas anteriormente y se identifican cuando se *definen*. Se observa también que existe una liga en una de las variables que delimita el *adjuntador de archivos*, pero este procedimiento se delimita más adelante.

```
echo "<font color=#2356DD><br><br><i><b> [ * ]  Required
fields</b></i></font>";
?>
<br>

<table align="center" border="0" cellspacing="2">
</table>

<font color=#aa3456>
  <h4> Choose and load your image <br><br></h4></font>

 <form action="VerGuarSess4b.php" method="post">

<table align="center" border="3" cellspacing="2">
  <tr>
    <td align="center" width="160">Imag  <b>*</b><br></td>
    <td width='290' align='center'>
<?php
         echo "<A href =
GuardaArchivo.php?clave=".htmlspecialchars(urlencode($clave))."">Choose
your file </A>";?></td></tr>

 <tr> <td></td>
    <td><b><center>
   <?php
        echo $variables_formulario[15];
   ?>       
        </center></b>
   </td></tr>
</table><br>

<font color=#aa3456>
<h4> Fill the form </h4></font>
<?php
   echo "<h4>Your project has the CODNAME: <br><b>" ;
   echo $variables_formulario[0]."</h4>";
?>

<table align="center" border="2" cellspacing="2">
  <tr> <td align="center">  Codname    <b>*</b><br> </td>
      <td><input type="Text" name="clave_imagen" size="30"
maxlength="10" value="<?php  echo $variables_formulario[0]; ?>">
   </td></tr>   <tr></tr>

   <tr><td align="center"> Researcher Name  <b>*</b><br></td>
    <td><input type="Text" name="autor" size="30" maxlength="50"
value="<? echo $variables_formulario[1]; ?>"></td></tr><tr></tr>

  <tr><td align="center"> Object Name  <b>*</b><br></td>
    <td><input type="Text" name="nomb_objeto" size="30" maxlength="150"
```

```
value="<?php echo $variables_formulario[2]; ?>"></td></tr><tr></tr>

  <tr><td align="center"> Project Title   <b>*</b><br></td>
    <td><input type="Text" name="titulo_proy" size="30" maxlength="250"
value="<? echo $variables_formulario[3]; ?>"></td>
   </tr><tr></tr>

  <tr><td align="center"> Objet Type  <b>*</b><br></td>
    <td><input type="Text" name="tipo_objeto" size="30" maxlength="100"
value="<? echo $variables_formulario[4]; ?>"></td>
   </tr><tr></tr>

  <tr><td align="center" >Description  <b>*</b><br></td>
   <td><textarea name="descripcion" rows="5" cols="30"><?php echo
$variables_formulario[5] ?> </textarea></td>
  </tr><tr></tr>

  <tr><td align="center">Right_Ascencion<br></td>
    <td><input type="Text" name="ascension_recta" size="30"
maxlength="150" value="<?php echo $variables_formulario[6];
?>"></td></tr>

  <tr><td align="center">Declination<br></td>
    <td><input type="Text" name="declinacion" size="30" maxlength="150"
value="<? echo $variables_formulario[7]; ?>"></td></tr>

 <tr><td align="center">Telescope<br></td>
    <td><input type="Text" name="telescopio" size="30" maxlength="150"
value="<? echo $variables_formulario[9]; ?>"></td></tr>

  <tr><td align="center">Detector<br></td>
    <td><input type="Text" name="detector" size="30" maxlength="150"
value="<? echo $variables_formulario[10]; ?>"></td>
  </tr> <tr>
    <td align="center">Filter<br></td>
    <td><input type="Text" name="filtro" size="30" maxlength="150"
value="<? echo $variables_formulario[11] ?>"></td></tr>

  <tr><td align="center">Frecuency<br></td>
    <td><input type="Text" name="frecuencia" size="30" maxlength="150"
value="<? echo $variables_formulario[8]; ?>"></td></tr>

  <tr><td align="center">Reference<br></td>
    <td><input type="Text" name="referencias" size="30" maxlength="150"
value="<? echo $variables_formulario[12]; ?>"></td></tr>

  <tr><td align="right"> Links
      <b>http://</b> <br></td>
    <td><input type="Text" name="ligas" size="30" maxlength="150"
value="<? echo $variables_formulario[13]; ?>"></td>

  </tr><tr>
    <td align="center"> Email  <b>*</b><br></td>
    <td><input type="Text" name="email" size="30" maxlength="150"
value="<? echo $variables_formulario[14]; ?>"></td></tr>
</table>
```

Al terminar el formulario se muestra la página previa, que es el término para mandar por URL las variables (input tipe=submit), cabe destacar que la primera vez que se llenan los datos se van a manejar por medio de HTML (ya que $variables_formulario[ ]=' ', están vacías) y al pasar por la *página previa,* es cuando se comienzan a manejar como variables de sesión, si en este script se comienza a manejar las sesiones es porque ha regresado de la vista final de la página o del adjuntador de archivos.

```
<font color=#aa3456>
<h4> See the Page Prevew  <br></h4></font>
<table align="center" border="3">
    <td>
     <b><center>
   <INPUT type="Submit" value="  Page Prevew  " name="enviar">
     </center></b>
    </td>
</table>
```

La información del formulario se manda a otro script (*vista previa*), donde muestra las variables y la imagen como finalmente las observa el *usuario*.

5.2.2 Vista previa y guardado de datos

La vista previa (Fig. A-7 en el Apéndice) lleva a la persona que agrega la información para que observe la página en la red con la imagen y los registros finales; aquí se presentan dos opciones, si quiere tener una modificación y si se quiere guardar; en ambas opciones se regresa a *formulario*, pero la variable de identificación ('name' para HTML) es la que da la diferencia; para modificar los registros se lleva la variable *$inconforme* y, para guardarlos se lleva la variable $*conforme.* A continuación se muestra el script para ver la página final.

Como el script anterior, se comienza declarando la sesión y las funciones requeridas, así como la cabecera de HTML.

```
<?php
  //Iniciamos la sesion en la pagina
  session_start();
?>
<HTML>
        <HEAD><TITLE> SESION 4 Ver y Guardar </TITLE></HEAD>
   <BODY>
<?php
include ("../funcion.php");
```

Al comenzar a manejar las sesiones, se tienen que delimitar las *variables de sesión*, que se transfieren por primera vez por medio del formulario en HTML, por lo tanto, para manejarlas en

varios scripts las variables que se transfieren en el formulario se tienen que definir como variables en la sesión.

```
$variables_formulario[0]=$clave_imagen;
$variables_formulario[1]=$autor;
$variables_formulario[2]=$nomb_objeto;
$variables_formulario[3]=$titulo_proy;
$variables_formulario[4]=$tipo_objeto;
$variables_formulario[5]=$descripcion;
$variables_formulario[6]=$ascension_recta;
$variables_formulario[7]=$declinacion;
$variables_formulario[8]=$frecuencia;
$variables_formulario[9]=$telescopio;
$variables_formulario[10]=$detector;
$variables_formulario[11]=$filtro;
$variables_formulario[12]=$referencias;
$variables_formulario[13]=$ligas;
$variables_formulario[14]=$email;
```

Se comienzan a manejar colores y tamaños de letras, además que se presentan las variables en el 'cascarón' de la página final (formato, colores, distribución de los datos, etc., que se establecieron para el sistema); cabe mencionar que para el módulo de *página final* en la búsqueda del usuario, se colocan las variables en el 'cascarón', pero con las variables de la clave particular para el llamado a la base de datos.

```
<body text="#000000" bgcolor="#FFCC99" link="#0000EF"
vlink="#51188E" alink="#FF0000" background="../../fondo.jpg">

<br>
<h3><center> Astronomical Images Bank </center> </h3>
<p>
<p><b><?php echo $variables_formulario[2]; ?> </b><br>
<br>
```

Un punto importante para observar la imagen, es que el archivo donde se guarden las imágenes tengan permisos de usuario y de grupo, por ejemplo, se debe observar de la siguiente manera:

drwxrwx--- 2 apache apache 1274 Aug 10 17:29 Archivo_guarda_imagenes

```
<?php
   echo                              "<A                              href=\"".
htmlspecialchars("../../files/$variables_formulario[15]")."\">\n";
   echo  '<img  src=  "../../files/'  .  $variables_formulario[15]."\".
Height=200 width=200>\n","</A>";
 ?>
```

Y se presenta el formato como una tabla; también se incluyen las ligas y el poder enviar un email para el investigador.

```
      <br><br><p><b>Image description</b><br>
      <?php   echo $variables_formulario[5]  ?>
      <br> <p><b>Keyword Values</b><br>

         <table BORDER COLS=2 WIDTH="100%" NOSAVE >
           <tr NOSAVE>
              <td WIDTH="25%" NOSAVE>AUTHOR  </td>
              <td> <?php  echo $variables_formulario[1] ?>  </td></tr>

          <tr><td>TITLE</td>
              <td> <?php  echo $variables_formulario[3] ?>  </td></tr>

           <tr><td> RIGHT_ASCENSION </td>
           <td> <? echo $variables_formulario[6]; ?></td></tr>

           <tr><td>  DECLINATION  </td>
           <td> <? echo $variables_formulario[7]; ?></td></tr>

           <tr><td> OBJECT NAME</td>
           <td> <? echo $variables_formulario[2]; ?></td></tr>

           <tr><td> OBJECT TYPE</td>
           <td> <? echo $variables_formulario[4]?> </td></tr>

           <tr><td>TELESCOPE</td>
           <td> <? echo $variables_formulario[9]; ?></td></tr>

           <tr><td>DETECTOR</td>
           <td><? echo $variables_formulario[10]; ?></td></tr>

           <tr><td>FILTER</td>
           <td><? echo $variables_formulario[11]; ?></td></tr>

          <tr><td> FRECUENCY </td>
           <td> <? echo $variables_formulario[8]; ?></td></tr>

           <tr><td>CODNAME</td>
           <td> <? echo $variables_formulario[0]; ?></td></tr>

           <tr><td>REFERENCES</td>
           <td><? echo $variables_formulario[12]; ?></td></tr>

           <tr><td>LINKS</td>
             <td><?php
                  echo "<A
href='http://".$variables_formulario[13]."'>";
                  echo $variables_formulario[13];
                 ?>
             </td></tr>
         </table>
<center>
<p>Contact: <b>
   <?php echo $variables_formulario[1],"</b> ";
         echo " Email ";
         echo "<a href='mailto:".$variables_formulario[14]."'>";
         echo "$variables_formulario[14]";
```

```
        echo "</a></center>";
   ?>
```

Aquí es donde el sistema hace la diferencia entre las variables que manda a *formulario:* (*$conforme* o *$inconforme*)

```
<form action="formularioSess3b.php" method="post">
  <i><b>
   <INPUT type="Submit" value="   Save   " name="conforme">
      
   <INPUT type="Submit" value="  Modify  " name="inconforme">
  </i></b>
</center>
```

Cuando se envía la variable $conforme al módulo de *formulario*, entonces se realiza el guardado de los registros en las tablas de la base de datos.

```
if($conforme)
 // quiere decir que estas conforme y que estas listo
// para insertar en la base de datos
 {
```

Se verifica que los datos mínimos hayan sido llenados, si no es así, entonces se tendrá que regresar a *página previa*. Lo anterior se realiza con la finalidad de que las imágenes tengan por lo menos las características mínimas para ser buscadas por el usuario, además de registrar el archivo de la imagen y el nombre del autor.

```
 if (!$variables_formulario[1] or !$variables_formulario[2] or
!$variables_formulario[3] or !$variables_formulario[4] or
!$variables_formulario[5] or !$variables_formulario[15])
    {
      echo "<br><br><br>";
      echo "<div align=\"center\">Te faltan datos relevantes <br><h3>
REGRESA </h3></div>\n";
      exit();
    }
```

Como anteriormente se realizó la conexión a la base de datos, sólo se comprueba si la función es correcta.

```
    if($link)
      {
        echo "<br>Estas dentro de la base de datos<br>";
      }
```

Se comienza la inserción de los datos en las diferentes tablas y se va generando el número de identificación que ayuda a verificar si las tablas han sido llenadas correctamente.

```
  //  Comienzo a relizar insercion de datos

  // Tabla "Autores"
$InsertarAutor= mysql_query ("Insert into autores (AUTOR,email) values
('".$variables_formulario[1]."','".$variables_formulario[14]."')");

// id de la tabla Autores
$idAutor=mysql_insert_id();

// Base de Datos "im"
$InsertarIm= mysql_query("Insert into im (NombreImagen,DESCRIPCION)
values('".$variables_formulario[15]."','".$variables_formulario[5]."')"
);

//  id de la tabla Im
$idImagen=mysql_insert_id();

//Base de Datos "GENERAL"
$InsertarGENERAL= mysql_query("Insert into GENERAL (CLAVE_IMAGEN,
DECLINACION,NOMBRE_OBJETO,TITULO_PROYECTO,TIPO_OBJETO,ASCENSION_RECTA,F
RECUENCIA,TELESCOPIO,DETECTOR,FILTRO,LIGAS,REFERENCIAS) values
('".$variables_formulario[0]."','".$variables_formulario[7]."','".
$variables_formulario[2]."','".$variables_formulario[3]."','".$variable
s_formulario[4]."','".$variables_formulario[6]."','".$variables_formula
rio[8]."','".$variables_formulario[9]."','".$variables_formulario[10]."
','".$variables_formulario[11]."','".$variables_formulario[13]."','".$v
ariables_formulario[12]."' )");

// Id de la tabla GENERAL
$idGENERAL=mysql_insert_id();
```

Para comprobar si los datos han sido guardados correctamente, se pueden realizar varias cosas, por ejemplo, revisar la bandera de la variable $link, las variables $insertar o hacer un llamado general a la base de datos con los registros que acaban de ser guardados, esto último es lo que se realiza para el sistema y se observa para cada una de las tablas. Los registros que fueron llamados son los que se presentan al final como *notificación* al investigador junto con la aceptación de que las tablas fueron llenadas correctamente.

```
echo "<center><b> Your were add the next data <br></b></center>";

// Resultado de la BD "im"
$ultimoRes= ("select * from im where id_imagen=".$idImagen);
$resultado= mysql_query($ultimoRes);
$row= mysql_fetch_array($resultado);

if(!$row)
 {
  echo "<br>The data were not added to BD im <br>";
  exit();
 }

echo "<br>".$idImagen;
echo "<br>".$row["NombreImagen"];
```

```
echo "<br>".$row["DESCRIPCION"];

//Resultado de ls BD "GENERAL"
$ultimoResGRAL= mysql_query("select * from GENERAL where
ID=".$idGENERAL);
$rowGENERAL= mysql_fetch_array($ultimoResGRAL);

if(!$rowGENERAL)
   {
    echo "<br> The data were not added to  BD  GENERAL <br>";
    exit();
   }

echo "<br>".$rowGENERAL["ID"];
echo "<br>".$rowGENERAL["CLAVE_IMAGEN"];
echo "<br>".$rowGENERAL["DECLINACION"];
echo "<br>".$rowGENERAL["NOMBRE_OBJETO"];
echo "<br>".$rowGENERAL["TITULO_PROYECTO"];
echo "<br>".$rowGENERAL["TIPO_OBJETO"];
echo "<br>".$rowGENERAL["ASCENCION_RECTA"];
echo "<br>".$rowGENERAL["FRECUENCIA"];
echo "<br>".$rowGENERAL["TELESCOPIO"];
echo "<br>".$rowGENERAL["DETECTOR"];
echo "<br>".$rowGENERAL["FILTRO"];
echo "<br>".$rowGENERAL["LIGAS"];
echo "<br>".$rowGENERAL["REFERENCIAS"];

//Resultado de ls BD "autores"
$ultimoResAutor = mysql_query("select * from autores where
id_autor=".$idAutor);
$rowAutor= mysql_fetch_array($ultimoResAutor);

if(!$rowAutor)
   {
    echo "<br> The data were not added to  DB  AUTORES <br>";
    exit();
   }

echo "<br>".$idAutor;
echo "<br>".$rowAutor["AUTOR"];
echo "<br>".$rowAutor["email"];
```

En la *notificación* también se registra si el envío de email es correcto y utiliza las variables que fueron creadas en la llamada anterior a la base de datos, además que se coloca la liga para regresar a la etapa *Investigación–Divulgación.*

```
  echo "<center><br><br><A href= '../Principal.php'>  Main Page
</A></center>";

//Enviamos el e-mail al destinatario

   //Variables de configuracion del correo
   $asunto = "Base de Datos";
   $cuerpo_mensaje = "Se agregaron datos a la Base de Datos con los
```

```
siguientes datos:<br>la ID general:<br>".$idAutor."<br>El Autor:"
      .$rowAutor["AUTOR"]."<br>Su email:". $rowAutor["email"];

     $headers_mensaje = "Base de datos de Imagenes Astronomicas\n";

    //Funcion para enviar el correo
$mailenviado = mail("bia@astroscu.unam.mx", $asunto, $cuerpo_mensaje,
$headers_mensaje);

       if($mailenviado)
               echo " Mail was sent ";
```

Al terminar de guardar los registros, se tiene que destruir la sesión y dar por finalizadas sus variables; si no se realiza, los datos de las variables pueden confundirse con las nuevas o ser recuperadas por alguien externo.

```
session_unregister();
session_destroy();
exit();
```

5.2.3 Adjuntador de archivos

Para copiar el archivo de la imagen a los archivos del sistema se realiza un adjuntador de archivos, el cual comienza con una liga dentro del *formulario*. Se realiza con una liga, porque la variable al principio no está delimitada como variable de sesión (la primera vez que se entra), lo que provocaría que al pasar entre los scripts (si primero el investigador adjunta la imagen antes de ir a la página previa) las variables se pierden y no son guardadas.

También existe otro problema al pasar las variables, ya que se necesita otra entrada con un formato "submit", pero ya existe uno para pasar a la *página previa*, por lo tanto, si se agrega uno más, ambos se confundirían y no delimitarían su siguiente script. Para guardar la clave entonces se opta por pasarla por medio de una liga y se transfiere la variable *$clave*.

```
<?php
 echo "<A href =
GuardaArchivo.php?clave=".htmlspecialchars(urlencode($clave))."> Choose
your file </A>";?>
```

El script GuardaArchivo.php comienza dando de alta la sesión, ya que desde aquí sí se pueden manejar las sesiones y colocar un botón.

```
<?php
```

```
 session_start();
?>

<HTML><HEAD>
        <TITLE> Guardando Archivos </TITLE>
   </HEAD><br><br>
<BODY TEXT="#000000" LINK="#0000ff" >
```

Se define la variable de sesión que va a tomar la imagen.

```
<?php
$variables_formulario[16]=$clave;
?>
```

Para la búsqueda del archivo se realiza con el formato *multipart/form-data* y se presenta en una tabla para una mejor distribución de la información.

```
<form action="Guarda2.php"  method="post" enctype="multipart/form-
data">
<br>
   <center>
     <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>
   <TBODY>
   <tr>
    <center><img src="../g10_s66_46040.gif" width=520  height=100
align=bottom>
<tr><br><br></tr></tr>
   <TR VALIGN=TOP><TD>
     <p align=rigth><strong><font size=4  FACE=verdana SIZE=2>Choose
the files    </strong></FONT></P>
     </TD><TD WIDTH=289>
       <P><INPUT TYPE=FILE NAME="superdat" SIZE=30></P>
    </TD></TR>
 </TBODY>
 </TABLE>
```

Este es el botón que puede ser asignado para pasar las variables.

```
    <DIV ALIGN=CENTER>
     <P STYLE="margin-bottom: 0in"><INPUT TYPE=SUBMIT NAME="submit"
VALUE="   Add  File  "></P>
        </DIV>
  </FORM></BODY></HTML>
```

Al siguiente script pasan las variables de la imagen y se comprueba si el archivo está delimitado, si no es así, entonces se regresa a *adjuntar archivos*.

Si el nombre del archivo que contiene la imagen es correcto, entonces se realiza la copia al

sistema y se presentan las características principales de la imagen.

```
<?php
session_start();

 if (!$superdat_name="")
   {
     copy("$superdat",
"/export/local/httpd/htdocs/files/$superdat_name") or die("File no
copy");
   }

 else die("No file selected.");

?>
<html>
<head>
        <title> Agregando Archivo </title>
</head>

<body bgcolor=white text=black link=blue>
<br><br><br>
    <center><b><p align=center>You add the next file: </b><br><br>
   <? echo "Name:   ".$superdat_name;
      echo "<br> Size:    ".$superdat_size;
      echo "<br> File type:  ". $superdat_type;
```

Aquí se vuelve a definir la variable de la imagen como la *variable de sesión* que tiene ya el archivo guardado en el sistema.

```
  $variables_formulario[15]=$superdat_name;
  echo "<br>".$variables_formulario[15];
?>
```

Se cierra la página y regresa al formulario con la imagen ya en los archivos del sistema.

```
<form action= "formularioSess3b.php" method="post">

<input type ="submit" value=" Finish save the file " name="inconforme">

</center></body></html>
```

5.3 Programación de la base de datos y sus tablas

Las tablas para la base de datos se programan con comandos definidos para el sistema operativo linux, esto es, MySQL principalmente se maneja bajo software libre, aunque

una versión para windows ya es utilizada pagando la licencia a Microsoft.

Primero se necesita crear la base de datos para continuar con las tablas y su programación se realiza de la siguiente manera:

```
% create database imag;
Query OK, 1 row affected (0.02 sec)
```

Al tener la base de datos creada, se comienza a estructurar y delimitar las tablas. En sí, todas las tablas fueron programadas de manera muy similar y se observará en el siguiente script.

La primera tabla mostrada es ***'im'***, que tiene tres campos y es utilizada para *identificar a la imagen*.

I. *id_imagen*, es el número de identificación de los registros en la tabla y es la llave primaria.
II. *NombreImagen*, nombre del archivo de la imagen.
III. *Descripción*, es el escrito que se realiza para dar un breve resumen del objeto celeste y es el campo más grande.

```
%    create table im (id_imagen int unsigned not null auto_increment
primary key,
%   NombreImagen varchar(80),
%   DESCRIPCION varchar(250));
```

La segunda tabla es ***'autores'***, también tiene tres campos e *identifica al autor o autores*.

1.- El *id_autor*, es el número de registro en la tabla y es la llave primaria.
2.- El *autor*, es el nombre de los autores o autor.
3.- El *email*, es el correo electrónico de los autores de la imagen.

```
%   create table autores (id_autor int unsigned not null auto_increment
primary key,
%   Autor varchar(150),
%   email varchar(100));
```

La última tabla para el guardado de la información de las imágenes es ***'General'***, aquí se encuentran todas las *características para la observación y referencias de la imagen*, es por eso que tiene la mayoría de los registros, pero los campos son pequeños en tamaño.

- *Id*, es el número de identificación de los registros de la base.
- *Clave_imagen*, es la clave primaria del proyecto.

- *Titulo_proyecto*, nombre de la investigación.
- *Nombre_objeto*, nombre del objeto de acuerdo al catálogo elegido por los investigadores.
- *Tipo_objeto*, clasificación del objeto celeste.
- *Declinación*, coordenadas del objeto observado en el periodo de tiempo estudiado.
- *Ascención_recta*, coordenada junto con la declinación para delimitar la posición del objeto.
- *Telescopio*, clasificación o nombre del telescopio ocupado.
- *Detector*, tipo de aparato utilizado para el telescopio.
- *Filtro*, el utilizado para realizar la observación.
- *Frecuencia*, ocupada para la observación de la imagen.
- *Ligas*, referencias en internet sobre el proyecto.
- *Referencias*, son los artículos, libros o información que el investigador crea conveniente.

```
%    create table GENERAL (id int unsigned not null auto_increment
primary key,
%   CLAVE_IMAGEN varchar(10),
%   DECLINACION varchar(20),
%   NOMBRE_OBJETO varchar(150),
%   TITULO_PROYECTO varchar(200),
%   TIPO_OBJETO varchar(100),
%   ASCENSION_RECTA varchar(10),
%   FRECUENCIA varchar(20),
%   TELESCOPIO varchar(20),
%   DETECTOR varchar(20),
%   FILTROvarchar(5),
%   LIGAS varchar(100),
%   REFERENCIAS varchar(150));
```

La última tabla es para identificar quienes pueden entrar a modificar los datos en la base ya sea un *administrador* o un *investigador*; para éste último, la tabla ayuda a delimitar las letras para estructurar su clave y su identificación, por lo que tiene los siguientes campos:

1.- *Lgin*, coloca un nombre genérico para el investigador.
2.- *Password*, es la contraseña del usuario.
3.- C*laveAutor*, las letras para generar las claves particulares de los proyectos.
4.- *NombreAutor*, es el nombre completo del autor que está dentro de la base de datos.

Y la programación es de la siguiente manera:

```
%   create table Authent (lgin varchar(20) not null primary key,
%   pswd varchar(30),
%   claveAutor varchar(10),
%   NombreAutor varchar(150));
```

Como observamos, la programación para generar las tablas es relativamente fácil y el modificar datos desde el sistema operativo son funciones cortas y eficientes.

Al terminarse la programación del código escrito, entonces el siguiente paso es hacer la referencia en donde funcionará el sistema, la instalación final, entre otras cosas que se detallan en el capítulo siguiente.

VI

ANÁLISIS DE RESULTADOS Y ESPECIFICACIONES FINALES

6. Análisis de resultados y especificaciones finales

La última parte del sistema, de acuerdo a la metodología que se sigue, es la instalación del sistema en el lugar donde va a quedar definitivamente, especificaciones y comentarios finales, así como las conclusiones respecto al BIA.

6.1 Estructura final del sistema

Exclusivamente para el sistema se tienen tres computadoras con las siguientes características:

1.- Pentium IV 1.5 Ghz con disco duro 20 Gb, Tarjeta de Red, Tarjeta de Vídeo 32 Mb, Monitor 19", CD CREATIVE 52X y CDWRITER.

2.- AMD 1800 con disco Duro 40 Gb, Tarjeta de Red, Tarjeta de sonido 32 bits, Tarjeta de Vídeo 32Mb, Monitor 19", CD CREATIVE 52X y CDWRITER.

3.- Silicon Graphics 320 con Pentium III 500 Mhz, disco Duro 40 Gb, Tarjeta de Red, Tarjeta de sonido 32 bits, Tarjeta de Vídeo 32 Mb, Monitor 19", CD CREATIVE 52X y CDWRITER.

Y se cuenta con seis discos duros, cada uno de 20 Gb haciendo un total de 120 Gb.

Para almacenar la información del banco se necesita formar un arreglo entre los discos duros de tal forma que se realice una distribución eficaz de la información (imágenes, tablas y scrips principalmente), además que el sistema tenga seguridad en caso de que un disco o configuración falle, que sea rápido y accesible y que no requiera demasiados comandos para su administración.

La principal tecnología que existe en el mercado con respecto a los arreglos de discos es la RAID presentada en 1998 por la Universidad de California en Berkeley. Es la unión física de varios discos duros, pero lógicamente se toma como uno sólo de mayor capacidad y fiabilidad que cada uno de los discos duros que lo componen por separado, además de proporcionar un mejor desempeño en la transferencia de datos y tolerancia a fallas.

El *controlador RAID* maneja la forma del almacenamiento de los datos, donde se acceden a ellos a través de los arreglos físicos y lógicos, ayuda a los usuarios a asegurar que el sistema operativo sólo vea las unidades lógicas y que tenga un fácil manejo en la administración general del esquema.

Se va cumpliendo para la tecnología RAID que el mejor desempeño es cuando se distribuye la carga de trabajo en paralelo entre todos los discos duros haciendo espejos de la información, esto va haciendo peso en la desventaja que a mayor número de discos, mayor es el desperdicio de memoria. Por lo que dependiendo del espacio y la distribución de los espejos es como surgen las diferentes estructuras de RAID.

El ***RAID nivel 0*** es el arreglo de discos más rápido que pueda tener, ya que combina todos los discos duros en uno sólo, cuya capacidad es la suma de todos los discos, de esta forma, si hacemos un sistema RAID con 4 discos duros de 40GB el sistema verá un único disco duro de 160GB. Además los datos son repartidos por *todos* los discos duros de forma que la velocidad de acceso también aumenta considerablemente, ya que los datos son leídos de todas las unidades a la vez. El desempeño es mejor que un sólo disco debido a que la carga de trabajo está balanceada entre los miembros del arreglo.

Para el arreglo se recomienda usar unidades de disco idénticas, ya que la capacidad del arreglo de discos es igual al número de miembros multiplicado por el miembro de menor capacidad. Por ejemplo, un disco de 40GB y uno de 60GB formarán un arreglo de 80 GB (40GBx2).

Esta estructura se toma como un método de mapeo de disco orientado al desempeño, pero no para la seguridad, ya que cuando un disco dentro del arreglo falla, lo afecta todo; se podría ejemplificar como la estructura de un puente, si un pedazo de éste es roto, todo el puente queda inutilizado. Por lo que se concluye que el principal problema de RAID 0 es que un fallo en uno de los discos duros implica la pérdida de todos los datos, pues se hallan repartidos uniformemente entre todas las unidades.

La Fig. 6-1 muestra el esquema de un arreglo RAID 0.

La configuración ***RAID nivel 1*** mantiene dos discos duros iguales, esto significa hacer un espejo completo de la información que se tiene, si un disco sufre una falla mecánica o no responde, la unidad secundaria continuará funcionando y proporcionando los datos correctos o si una de las unidades tiene un error de sector físico, el disco espejo continua la función sin poner en peligro la información del sistema, debido a esto, la capacidad del arreglo es la mitad de la capacidad de las unidades de disco, por ejemplo dos unidades de 40GB tienen una capacidad combinada de 80GB pero tendrán una capacidad de 40GB de almacenamiento usable.

En esta arquitectura el sistema operativo presenta al usuario un único disco, por lo que no se puede acceder al disco 2 (disco de respaldo) bajo ningún concepto, sólo el sistema puede usarlo y lo ideal es usar dos discos duros idénticos, ya que en caso de usar uno más grande que otro la controladora RAID verá ambos con el tamaño del disco más pequeño, es decir, si se usa un disco de 40GB y otro de 20GB la configuración nos dará un disco de 20GB.

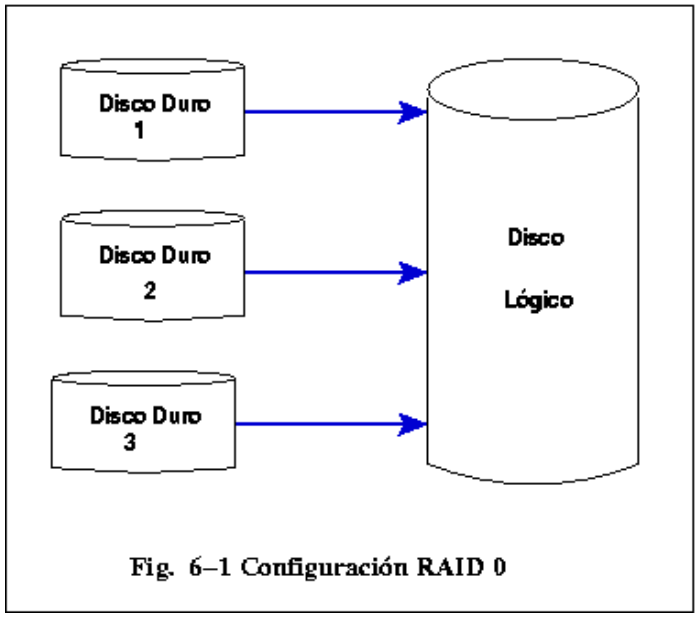


Fig. 6–1 Configuración RAID 0

Es el método más lento debido a que la información se debe copiar en los dos discos a un mismo tiempo, sin embargo, es el método más simple para proporcionar alta confiabilidad. RAID 1 incrementa el costo porque usa el doble de discos duros para construir el arreglo. La Fig. 6-2 muestra el arreglo de RAID 1.

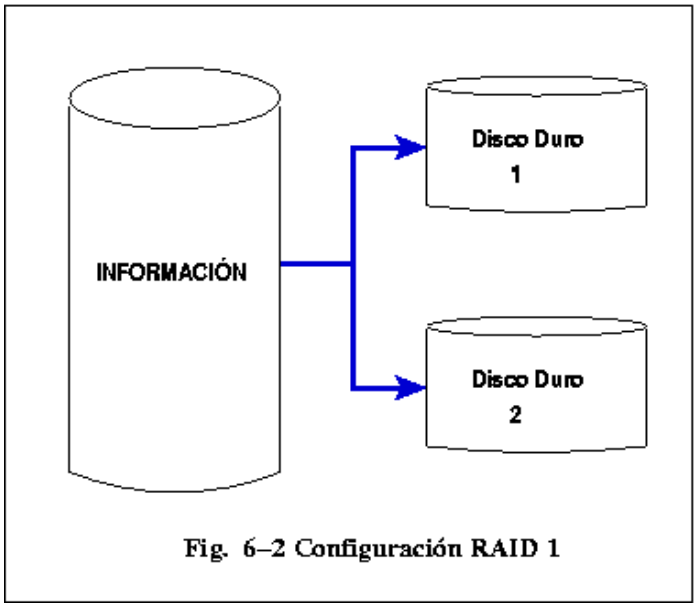


Fig. 6–2 Configuración RAID 1

La configuración ***RAID nivel 4*** tiene que ser implementado a más de tres discos, guarda la información en un disco y guarda la paridad completa en otro disco, esto es, RAID nivel 4 graba

la información a nivel de blocks entre varios discos, grabando la paridad en uno sólo. Si un disco falla, la paridad permite recuperar toda la información (excepto si el disco es el de *paridad*), pero si son dos los discos que fallan entonces ya no podrán recuperarse los datos. La arquitectura de nivel 4 es muy buena para lecturas, pero para la escritura es más lenta, ya que requiere la grabación adicional de la paridad. El costo por espacio no es tan drástico ya que sólo un disco es el que graba la paridad.

La Fig. 6-3 muestra una configuración RAID 4.

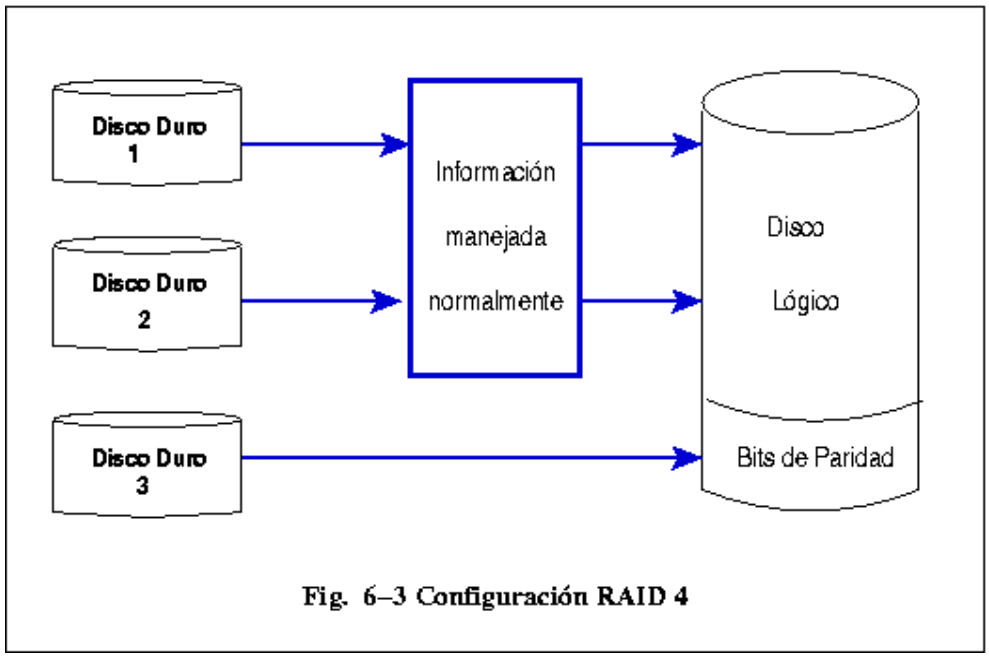


Fig. 6–3 Configuración RAID 4

La configuración ***RAID nivel 5***, es la más utilizada siempre y cuando se tengan más de tres discos físicos para el arreglo y se tenga poca redundancia. Este nivel es similar al anterior, con la diferencia que el almacenaje de los bits de paridad se hace de forma distribuida.

También es basado en el RAID nivel 0 ya que toda la información es repartida en todas las unidades; pero se calcula un bit de paridad, que son repartidos en toda la estructura del arreglo. Por lo tanto, cada disco contiene una porción de bits de paridad.

El arreglo disminuye el rendimiento, pero tiene una redundancia que ayuda a recuperar la información cuando un disco se daña, gracias a los datos del resto de los discos, la tabla de bits de paridad y matemática binaria.

La capacidad de una unidad RAID 5 depende fundamentalmente del número de discos en el arreglo, los bits de paridad usan tanto espacio de almacenamiento como un disco duro del

conjunto, por lo que si tenemos un total de cuatro discos, la capacidad del conjunto será la suma de tres discos físicos.

En el arreglo existe una alta probabilidad de que se recupere un disco si ha sido dañado, pero las probabilidades disminuyen drásticamente para recuperar un segundo disco si son dañados simultáneamente. Pero su desventaja principal es el complejo cálculo de paridades, que hace que el sistema que lo tenga resulte más lento. La Fig. 6-4 muestra la configuración de RAID 5.

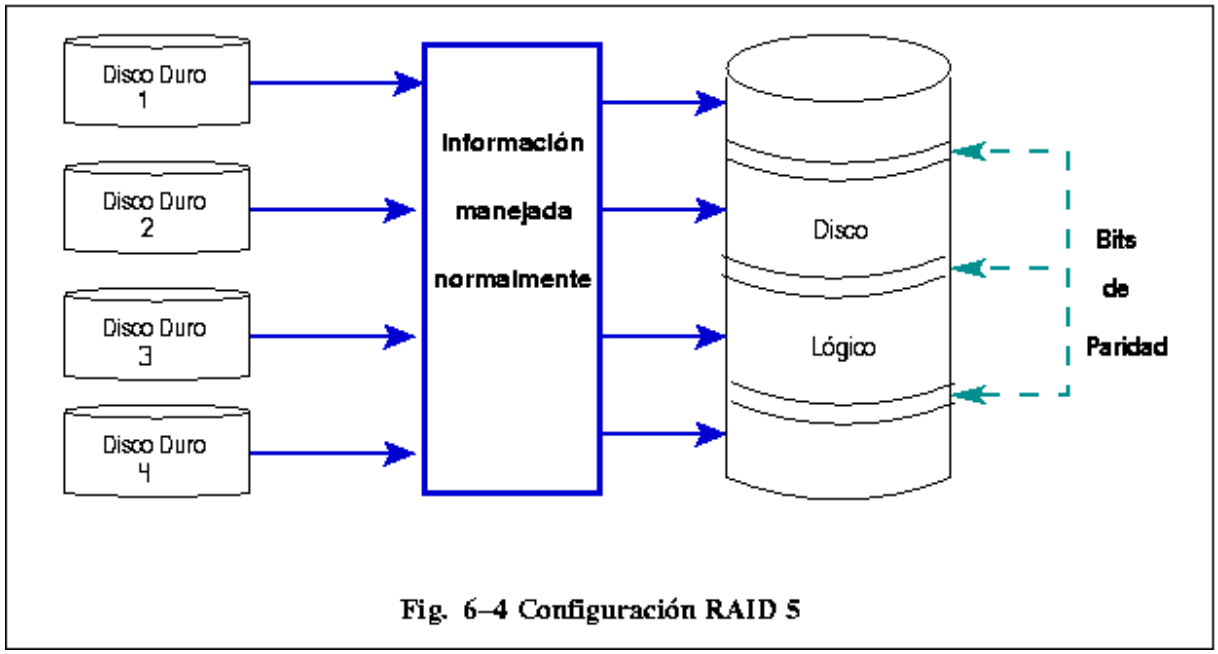


Fig. 6–4 Configuración RAID 5

Para el Banco de Imágenes Astronómicas se instaló el último arreglo (RAID 5), ya que en comparación con RAID 0 y RAID 1, el arreglo en el nivel 5 tiene las ventajas de éstos pero disminuyendo sus desventajas y a un costo mucho menor.

Un plan de contingencia es darle seguridad al sistema (descrito en capítulos anteriores), en nuestro caso si llegase a presentar algún problema, por ejemplo, perder algún dato del BIA, perderse scrips, etc., el arreglo de RAID 5, permitirá recuperar los datos y darle respaldo al sistema.

RAID 5 tiene una redundancia que no tiene el nivel 0 y disminuye el número de discos de respaldo respecto al nivel 1. Es muy similar en comparación con el nivel 4, ya que sólo es la distribución de los bits de paridad a una sola sección del disco lógico, pero si esta sección es la dañada, entonces se pierde cualquier opción de respaldo para la información de RAID 4.

La desventaja general del arreglo es que el cálculo de paridad lo hace complejo, por lo que no se puede esperar una gran velocidad y mucho menos si se utiliza un controlador RAID por software (como se utiliza en nuestro sistema); aunque la velocidad es importante para cualquier sistema, el

BIA es un sistema de búsqueda y almacenamiento de información, por lo que se requiere de una alta velocidad, pero en comparación con un arreglo para un clusters u otros sistemas, la velocidad en nuestro sistema puede ser tolerable fácilmente.

6.2 Especificaciones

El Banco de Imágenes Astronómicas (BIA) es la parte de *investigación* del proyecto denominado "Creación de un Banco de Imágenes Astronómicas para la investigación", donde colaboran el Instituto de Astronomía de la Universidad Nacional Autónoma de México (IAUNAM) con el apoyo del Consejo Nacional de Ciencia y Tecnología (CONACyT).

El sistema podrá ser utilizado por cualquier persona que esté interesada en la astronomía y guardarán información aquellas personas que sean acreditadas por el instituto.

Se tuvo la participación mural en el XVII Congreso Nacional de Astronomía, 21, 22, y 23 de Mayo de 2003. El formato del mural se muestra en la Fig. A-11 en el Apéndice.

La principal dirección electrónica del BIA es:
http://bia.astroscu.unam.mx

Y la dirección para la parte de investigación es la siguiente:
http://bia.astroscu.unam.mx/research/

CONCLUSIONES

Conclusiones

Con la realización del proyecto Banco de Imágenes Astronómicas (BIA), se logra un apoyo para cualquier persona que esté interesado en la astronomía, ya sea por curiosidad o porque se desenvuelva en alguna rama de esta ciencia.

Este sistema es el resultado de un estudio para obtener una mayor independencia entre un sistema de cómputo y un administrador a cargo, mas sin embargo, la independencia no se realiza completamente, ya que esta última parte tiene un papel sumamente relevante y que en ningún momento se puede prescindir de ella.

El proyecto también estimula la responsabilidad de mantener la información correcta y actualizada por parte de la gente que la trabaja, esto es, si alguna persona quiere poner un trabajo en el sistema, es sólo cumplimiento de ésta para que la añada o actualice.

Uno de los principales objetivos del BIA es fomentar la divulgación entre los investigadores y entre el público en general. Entre los primeros, porque pueden compartir información y observar imágenes que otros han estudiado. Y entre los segundos, fomenta la curiosidad y el interés para adentrarse en su estudio, esto conlleva a que más gente se decida a estudiar la rama de las ciencias y dedicarse a la investigación, opción que actualmente en el país no está muy difundido entre la gente joven que entra estudiar a la Universidad.

El problema de que la matrícula disminuye conforme pasa el tiempo en algunas áreas de ingeniería o de ciencias se debe en parte a que no se estimula, ni se tiene un conocimiento sobre éstas, por ejemplo, lugares de trabajo, oportunidades, porcentajes de estudios, etc., por lo que un joven estudiante opta por carreras más comerciales o tradicionales.

Cuando las personas comienzan a interactuar con imágenes, eventos astronómicos o personas, se rompe una barrera imaginaria muy marcada entre un investigador con estudios doctorales y un estudiante de secundaria.

El BIA pretende ser un enlace de los muchos que debemos fomentar para tener un mejor futuro en el país, no sólo en esta área, sino en la gran diversidad de investigaciones que existen en la actualidad.

En el ámbito de la ingeniería del software, se concluye que el utilizar una metodología es sumamente relevante, ya que se lleva una organización del proyecto y gracias a esto, se puede tener un control sobre los riesgos internos y externos que lo rodean, como los son el personal, tratos con el cliente, problemas con el software, etc.

Más adelante, el proyecto posiblemente tendrá que modificarse, ya que las circunstancias que rodean a cualquier sistema cambian conforma pasa el tiempo. En el BIA, la forma de distribución de la página final o tal vez los registros que se guardan, son los que tienen una mayor probabilidad de cambiar, ya que la tecnología, la nueva implementación de telescopios, sistemas de redes, etc., influyen considerablemente en la forma de observar y obtener resultados.

Con el BIA se han logrado muchos resultados y ha tenido un exitoso camino, mas sin embargo, todavía le falta mucho por recorrer, más imágenes, más información, pero todo esto se dará conforme transcurra el tiempo y las personas vayan estimulando su curiosidad.

APÉNDICE

Tabla 3-c GANT del BIA

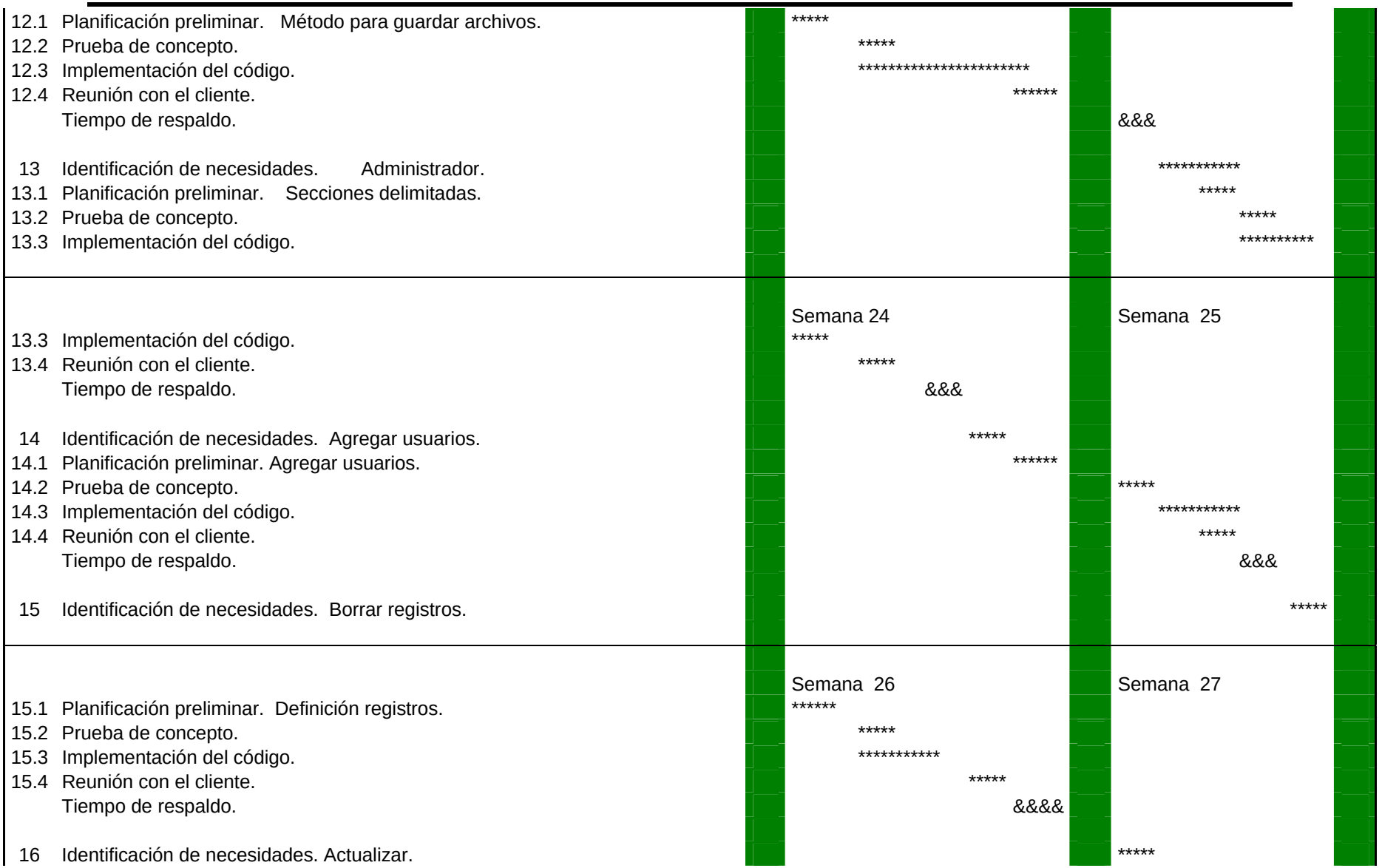
	TAREAS	Semana 1	Semana 2
1	Identificación de necesidades y recursos. Otras bases de datos.	**********	
1.1	Reunión con los clientes.	**********	
1.2	Identificación de limitaciones. Equipo-Personal.	**********	
1.2	Establecer la declaración del producto tipo software.	**********	*****
	Tiempo de respaldo.		&&&
2	Planificación preliminar.		**********
2.1	Valoración del riesgo tecnológico HW instalado.		**********
2.2	Instalación software.		***** *****
2.3	Prueba de concepto. Pruebas de instalación.		*****

		Semana 3	Semana 4 - 5 - 6 - 7
2.1	Valoración del riesgo tecnológico. HW instalado.	**********	
2.3	Prueba de concepto. Pruebas de instalación.	**********	
2.4	Reunión con el cliente.	*****	
	Tiempo de respaldo.	&&&&&&&	
3	Identificación de necesidades. Aprendizaje del software.		**********
3.1	Reunión con el cliente.		*****
	Tiempo de respaldo.		&&&

		Semana 8	Semana 9
4	Identificación de necesidades. Diseño de las tablas de la base de datos.	**********	
4.1	Planificación preliminar. Bosquejo de las tablas y sus relaciones.	******	
4.2	Prueba de concepto.	**********	
4.3	Implementación del concepto.	**********	
4.4	Reunión con el cliente.	******	
	Tiempo de respaldo.		&&&
5	Identificación de necesidades. Diseño de la interfaz de usuario: página principal.		**********

5.1 Planificación preliminar. Bosquejo de los programas y sus ligas.		***********
	Semana 10	Semana 11
5.2 Prueba de concepto.	******	
5.3 Implementación del concepto. Página principal.	**********************	
5.4 Reunión con el cliente.	******	
Tiempo de respaldo.		&&&&&&
6 Identificación de necesidades. Interfaz de usuario. 1era. búsqueda.		*****
6.1 Planificación preliminar.		***********
6.2 Prueba de concepto.		*****
6.3 Implementación del concepto.		*****
	Semana 12	Semana 13
6.3 Implementación del concepto	****************	
6.4 Reunión con el cliente.	*****	
Tiempo de respaldo.	&&&	
7 Identificación de necesidades - Manejo de imágenes.	******	
7.1 Planificación preliminar. Presentación de imágenes para la base de datos.		***************************
	Semana 14	Semana 15
7.2 Prueba de concepto.	******	
7.3 Implementación del concepto.	*****************************	
Tiempo de respaldo.		&&&&&&
8 Identificación de necesidades. Interfaz de usuario – página final.		***********
8.1 Planificación preliminar. Presentación de la búsqueda final.		***********
	Semana 16	Semana 17
8.2 Prueba de concepto.	******	

8.3 Implementación del concepto.	***********	
8.4 Reunión con el cliente	*****	
Tiempo de respaldo	&&&&	&&&
9 Identificación de necesidades.Diseño de tablas para agregar en base de datos.		*****
9.1 Diseño de la interfaz usuario Login-Password.		***********
9.2 Planificación preliminar Login-Password.		*****
9.3 Implementación del Código.		****************
9.4 Reunión con el Cliente.		&&&
	Semana 18	Semana 19
Tiempo de respaldo	&&&	
10 Identificación de Necesidades. Formulario.	*****	
10.1 Planificación Preliminar. Características del formulario.	***********	
10.2 Prueba de concepto.	******	
10.3 Implementación del código.		***********
10.4 Reunión con el cliente.		*****
Tiempo de respaldo		&&&
11 Identificación de necesidades. Muestra de la página.		*****
	Semana 20	Semana 21
11.1 Planificación preliminar. Mostrar prototipo de la página final.	***********	
11.2 Prueba de concepto.	*****	
11.3 Implementación del código.	*****************	
11.4 Reunión con el cliente.		*****
Tiempo de respaldo.		&&&&&&
12 Identificación de necesidades. Adjuntar archivos.		*****
12.1 Planificación preliminar. Método para guardar archivos.		*****
	Semana 22	Semana 23


12.1 Planificación preliminar. Método para guardar archivos.
12.2 Prueba de concepto.
12.3 Implementación del código.
12.4 Reunión con el cliente.
Tiempo de respaldo.
13 Identificación de necesidades. Administrador.
13.1 Planificación preliminar. Secciones delimitadas.
13.2 Prueba de concepto.
13.3 Implementación del código.
Semana 24
Semana 25
13.3 Implementación del código.
13.4 Reunión con el cliente.
Tiempo de respaldo.
14 Identificación de necesidades. Agregar usuarios.
14.1 Planificación preliminar. Agregar usuarios.
14.2 Prueba de concepto.
14.3 Implementación del código.
14.4 Reunión con el cliente.
Tiempo de respaldo.
15 Identificación de necesidades. Borrar registros.
Semana 26
Semana 27
15.1 Planificación preliminar. Definición registros.
15.2 Prueba de concepto.
15.3 Implementación del código.
15.4 Reunión con el cliente.
Tiempo de respaldo.
16 Identificación de necesidades. Actualizar.

16.1 Planificación preliminar. Datos para actualizar. 16.2 Prueba de concepto. 16.3 Implementación del código. 16.4 Reunión con el cliente.		***** ***** ********** *****
Tiempo de respaldo 17 Identificación de necesidades. Detalles interfaz de usuario. 17.1 Planificación preliminar. 17.2 Prueba de concepto. 17.3 Implementación del código.	Semana 28 &&& ********** 	Semana 29 ********** ***** *****
17.3 Implementación del código. 17.4 Reunión con el cliente. Tiempo de respaldo.	Semana 30 **********	Semana 31 ********** ********** &&&&&&
Tiempo de respaldo 18 Identificación de necesidades. Presentación Final. 18.1 Planificación preliminar. Estructura gráfica final. 18.2 Prueba de concepto. 18.3 Implementación del código.	Semana 32 &&& ********** **********	Semana 33 ********** ***** **********
18.3 Implementación del código. 18.4 Reunión con el cliente y finalización del proyecto. Tiempo de respaldo.	Semana 34 ********** ***** &&&&	Semana 35 &&&&&&

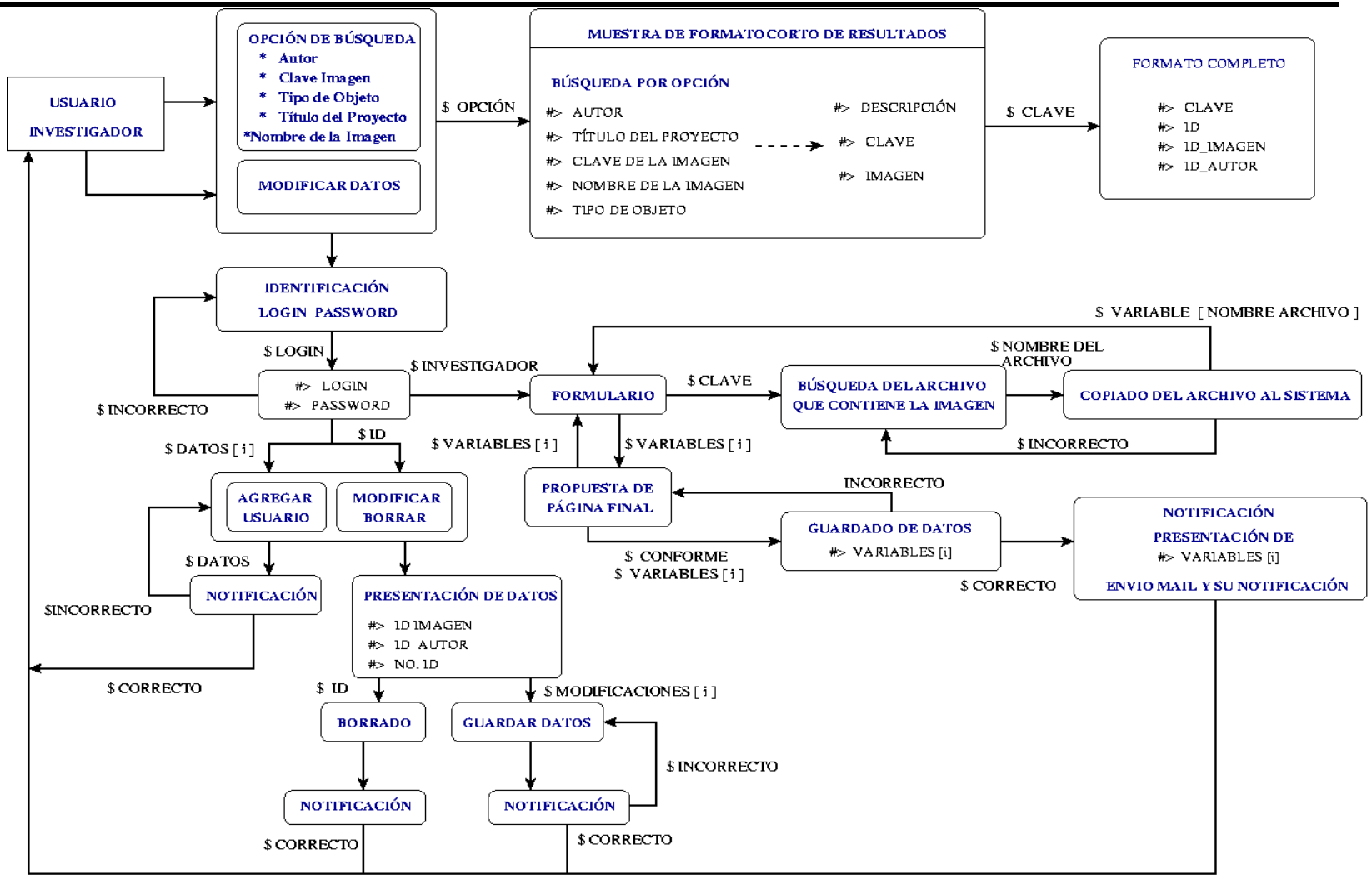


Fig. 4-3 Diagrama entidad-relación del BIA

Fig. A-1 Presentación del BIA para el Usuario

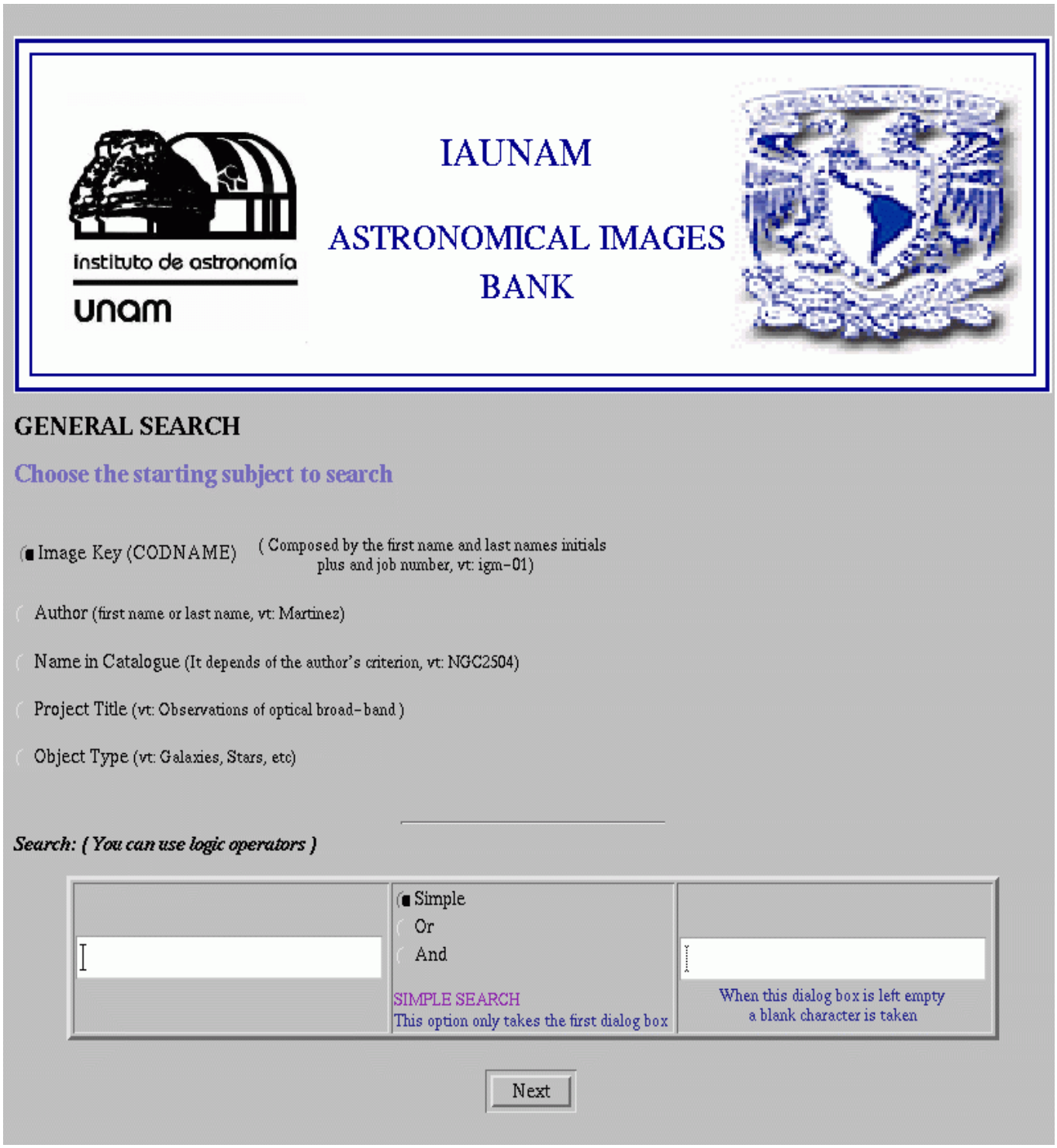


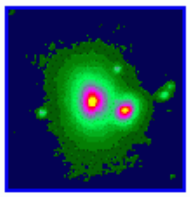
Fig. A-2 Presentación de Opciones

Results

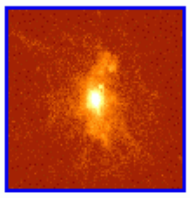
Search on AUTOR
with carrillo simple
Records found: 3

BACK

1.

No. Ident. **(ID)**	6
	R. Carrillo, I. Cruz-Gonzalez
rcm-03	Description: This is a pair of elliptical galaxies with traces of interaction, here the eastern component corresponds to the radio emission. The contour map of this E+E pair shows a common halo and external contour distorsions. The r -1/4 profiles shows a kin

2.

No. Ident. **(ID)**	4
	R. Carrillo, I. Cruz-Gonzalez
rcm-01	Description: Observations of optical broad-band of the NGC1275

3.

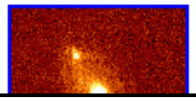
No. Ident. **(ID)**	5
	R. Carrillo, J. A. Garcia-Barreto

Fig. A-3 Presentación Final para el Usuario

Astronomical Images Bank

NGC3504

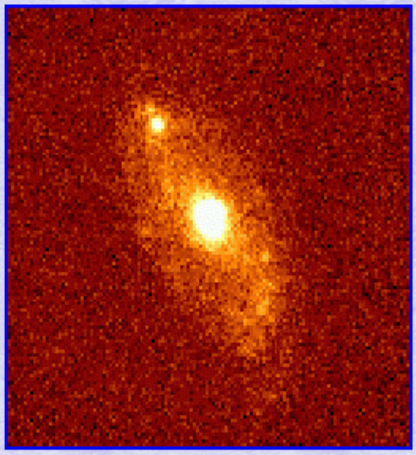


Image description
Observations of optical broad-band of the NGC3504

Keyword Values

AUTHOR	R. Carrillo, J. A. Garcia-Barreto
TITLE	Observations of optical broad-band of the NGC3504
RIGHT_ASCENSION	11h 03m 11
DECLINATION	+27° 58' 20"
OBJECT_NAME	NGC3504
OBJECT_TYPE	Galaxy
TELESCOPE	2.1 m SPM-OAN
DETECTOR	CCD (1024X1024)
FILTER	R
FREQUENCY	4.28X1014 Hz
CODNAME	rcm-02
REFERENCE	
LINKS	

Contact: **R. Carrillo, J. A. Garcia-Barreto** Email: rene@astroscu.unam.mx

BACK

Fig. A-4 Opción para modificar registros: Investigador o Administrador

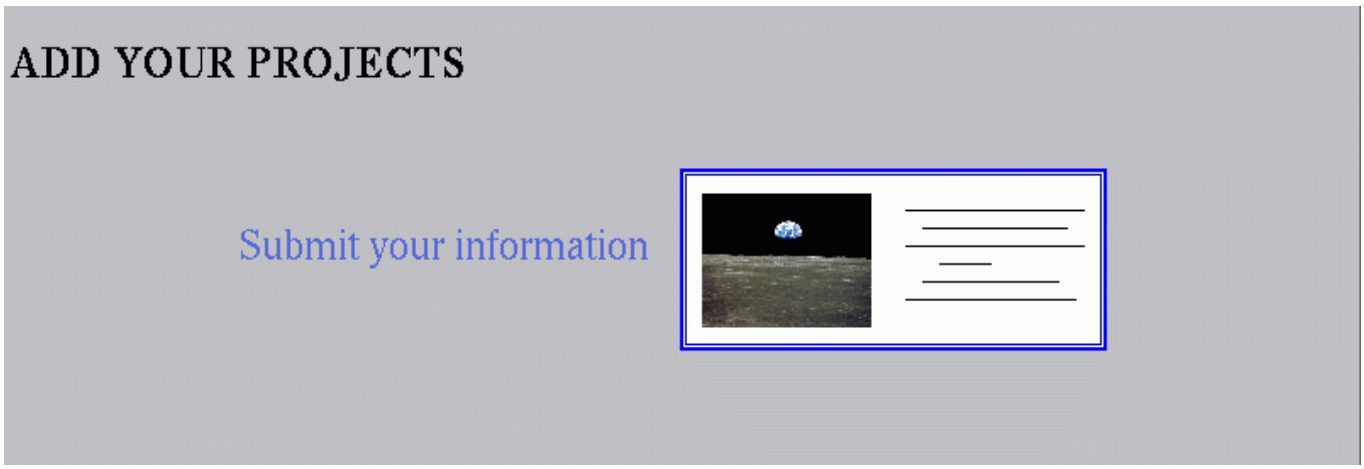


Fig. A-5 Login y Password

Fig. A-6 Formulario del BIA

Fig. A-6 Continuación del Formulario del BIA

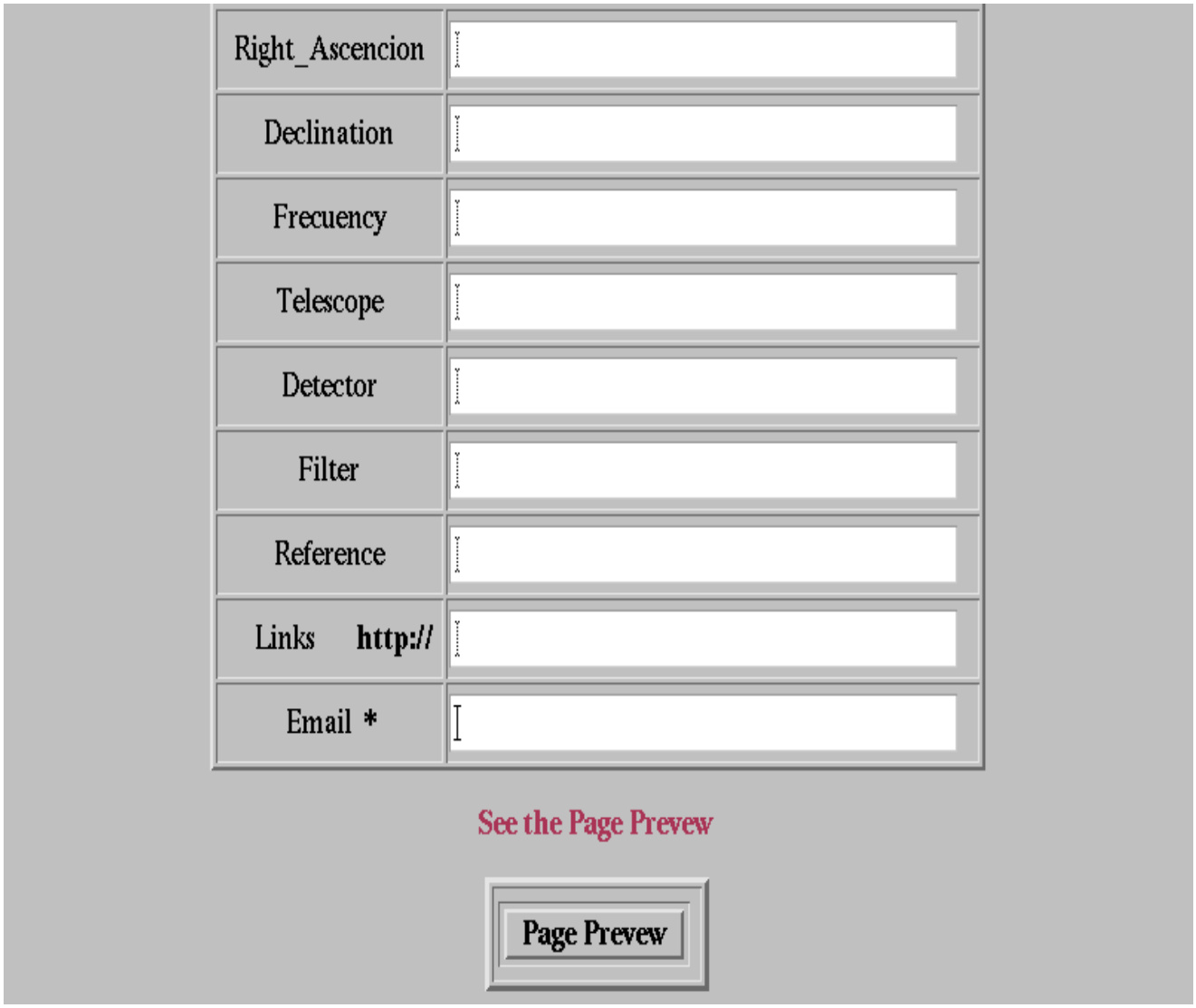


Fig. A-7 Página Previa

Astronomical Images Bank

NGC1275

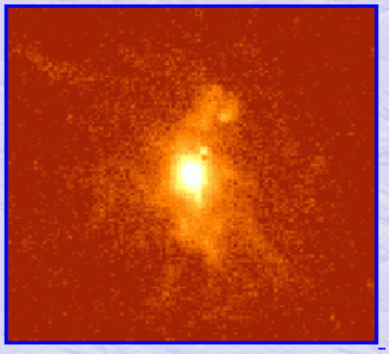


Image description
Observations of optical broad-band of the NGC1275

Keyword Values

AUTHOR	R. Carrillo, I. Cruz-Gonzalez
TITLE	Observations of optical broad-band of the NGC1275
RIGHT_ASCENSION	
DECLINATION	
FRECUENCY	
OBJECT NAME	NGC1275
OBJECT TYPE	Galaxy
TELESCOPE	
DETECTOR	
FILTER	
CODNAME	rcm-01
REFERENCES	
LINKS	

Contact: **R. Carrillo, I. Cruz-Gonzalez** Email rene@astroscu.unam.mx

Save Modify

Fig. A-8 Parte Administrador Agregar Nuevos usuarios.

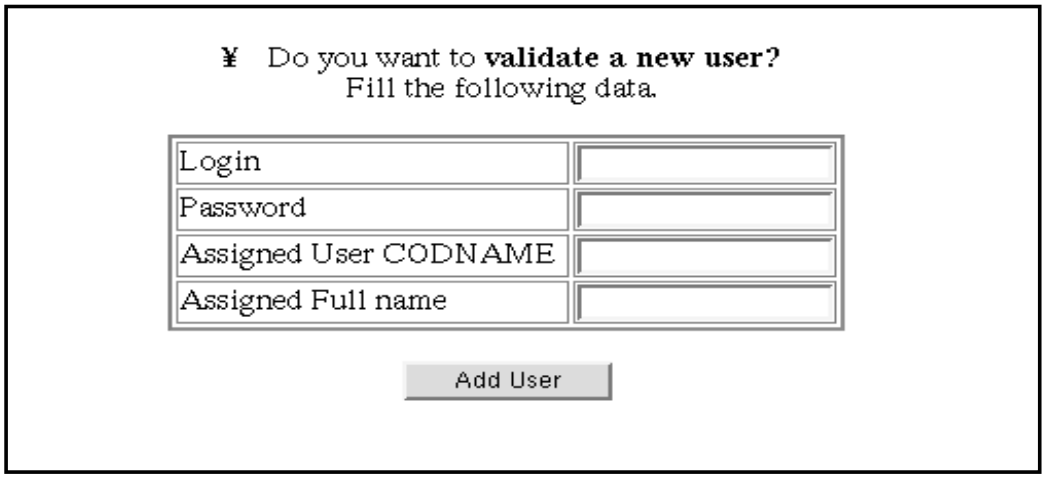


Fig. A-9 Parte Administrador Modificar o Borrar registros

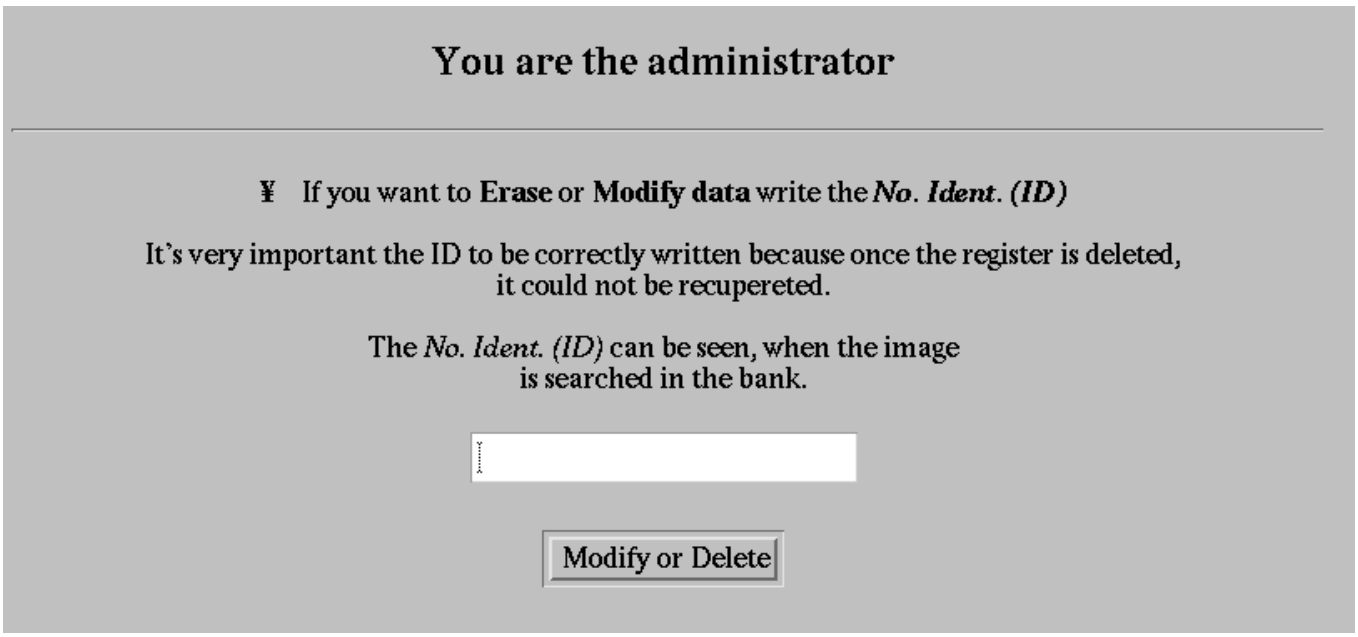
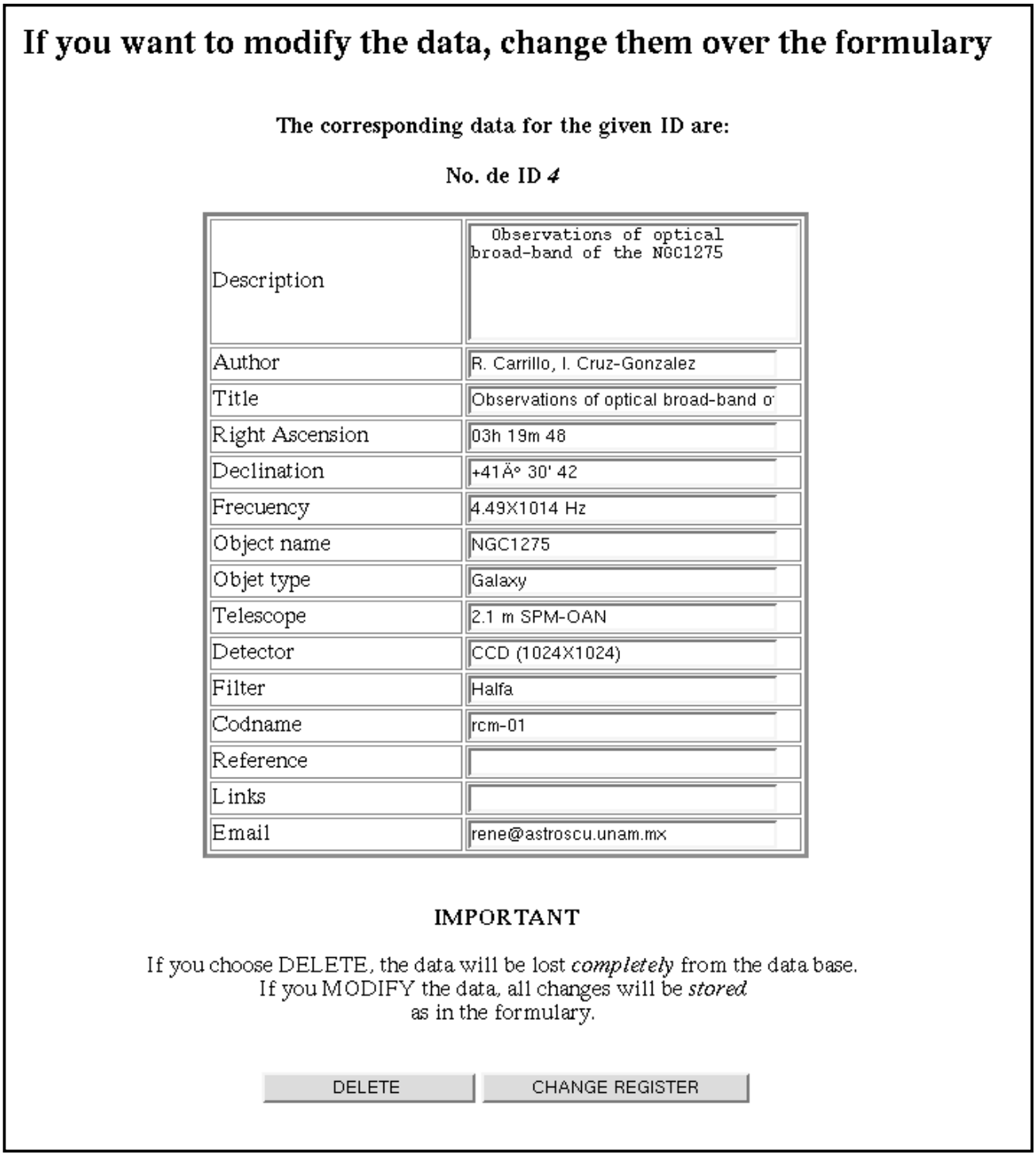


Fig. A-10 Sección Modificar o Borrar registros



If you want to modify the data, change them over the formulary

The corresponding data for the given ID are:

No. de ID *4*

Description	Observations of optical broad-band of the NGC1275
Author	R. Carrillo, I. Cruz-Gonzalez
Title	Observations of optical broad-band o
Right Ascension	03h 19m 48
Declination	+41Ä° 30' 42
Frecuency	4.49X1014 Hz
Object name	NGC1275
Objet type	Galaxy
Telescope	2.1 m SPM-OAN
Detector	CCD (1024X1024)
Filter	Halfa
Codname	rcm-01
Reference	
Links	
Email	rene@astroscu.unam.mx

IMPORTANT

If you choose DELETE, the data will be lost *completely* from the data base.
If you MODIFY the data, all changes will be *stored*
as in the formulary.

DELETE CHANGE REGISTER

BANCO DE IMAGENES ASTRONOMICAS
IA - UNAM

http://bia.astroscu.unam.mx

García Martínez Iliana - Zavala Pérez Gilberto

IAUNAM - Ciudad universitaria

El avance de la Astronomía en los últimos años, ha traído la necesidad del empleo de nuevos sistemas para el manejo y difusión de los datos obtenidos.
En el Instituto se cuenta con un gran acervo de imágenes astronómicas las cuales, por falta de un medio adecuado para su difusión no están disponibles a la comunidad.
El recopilar un gran acervo de imágenes astronómicas, integrar este acervo en un banco y difundirlo ante la comunidad son los principales objetivos de éste proyecto.

El sistema esta compuesto por tres partes principales, la primera es el **obtener de la información.**

Las dos secciones restantes son:
Agregar Información y la Administración.
En ambas se pide que se registre un login y un password y, dependiendo de la persona identificada el sistema muestra el formato correspondiente al tipo de usuario. Si no es acreditado, entonces se le especifica el motivo.

Para la obtener la información se necesitan dos criterios de búsqueda.
El primero es elegir un tema entre los siguientes:
a) Clave de la Imágen.
b) Autor.
c) Nombre en Catalogo.
d) Titulo del Proyecto.
e) Tipo de Objeto.

Y el segundo es una o más palabras para poder delimitar la búsqueda, entre éstas se puenden utilizar operadores lógicos:
OR ó AND.

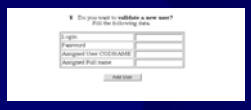
Se presenta al usuario una serie de bloques con la información que cumplió con los criterios de búsqueda pedidos.
Estos recuadros tienen la imagen, la opción de búsqueda, la clave del proyecto y el no. de identificación (*No.Id*).
Cada uno de estos bloques son una liga para accesar a la pagina principal del proyecto, donde se muestra la información específica de la imágen.

Aqui se muestran datos sobre la imagen, artículos relacionados e identificación del autor:
- La descripción sobre el proyecto,
- El nombre del o los autores,
- El título del proyecto,
- Las coordenadas ecuatoriales del objeto:
  la ascención recta y la declinación,
- El nombre del objeto, que depende del catálogo que haya seleccionado el autor,
- El tipo de objeto,
- El telescopio, su ubicación y alguna caracteríztica sobre éste,
- El detector,
- El filtro y
- La frecuencia en la que fue tomada la imagen,
- El código asignado a la imagen en éste banco,
- Referencias o artículos del mismo tema,
- Ligas a otras páginas con temas relacionados, según concideración del autor y,
- Email, correo electrónico del autor.

Para agregar información al BIA solo hay que llenar un formulario con los datos de la imagen y del autor, la descripción y adjuntar el archivo para realizar la copia de la imagen al sistema.
La clave se asigna automáticamente dependiendo de los trabajos que tenga el mismo autor en la base de datos.
No todos los datos son escenciales, pero algunos si lo son, como la imagen, el nombre del autor o el tipo de objeto, ya sea para la búsqueda en el banco o para identificar al autor.

El Administrador tiene dos funciones principales :
Agregar nuevos usuarios en la base de datos y,
Borrar o Modificar los registros que ya se encuentren en la base.

Para *agregar nuevos* usuarios se llena un formulario con cuatro campos:
» Login.
» Password.
» La clave para identificar los trabajos en el BIA y,
» Nombre completo del Autor.



Si se quiere borrar o modificar los registros se pide el Número de Identificación, (éste se muestra cuando se realiza la búsqueda en general).
Con el Número de Identificación (*No.Id*) se muestra toda la información que haya respecto a éste y directamente se realiza la selección de borrado o, si es para corrección, allí mismo se registran las modificaciones.
Finalmente pueden ser guardados los cambios.

Al ir llenando los datos, se puede visualizar la página final, ya que se tiene una página previa.
Si se encuentra algún defecto ó error, solo se regresa para modificar los datos.
Esto se puede hacer cuantas veces se desee.

Cuando se esta de acuerdo con forma final de la página solo queda guardar los registros en la base.
Si existiese algún problema al guardar la información el sistema reporta la dificultad y se tendrá que regresar a la pagina previa.
Si los datos son correctos el sistema manda un mensaje a la persona encargada de la administración y se presentan los datos que fueron agregados.

PERSPECTIVAS
La siguiente etapa de evolucion del BIA requiere de un mayor número de registros de datos por parte de los investigadores. Por lo mismo, se hace necesaria una participacion de éstos para que se convierta a corto plazo en una base de datos estándar para el Instituto de Astronomia.

BIBLIOGRAFÍA

Bibliografía

Libros

1. Alice Y., H. Tsai.
   Sistema de base de datos: Administración y uso.
   Prentice Hall Hisponoamericana S.A. Canada. 1990.

2. Bruegge, Bernd., Dutoit, Allen H.
   Ingeniería del Software orientado a Objetos
   Pearson Educación. México. 2002

3. Date, C. J.
   Bases de datos: Una guía práctica.
   Addison-Wesley Iberoamericana. Argentina. 1987.

4. De Miguel, Adoración. Piattini, Mario.
   Fundamentos y modelos de bases de datos.
   Alfaomega. España. 1999. 2a. edición.

5. Dubois, Paul.
   MySQL Edición Especial.
   Prentice Hall. Madrid. 2000.

6. Fairley, Richard E.
   Software Engineering Concepts.
   McGraw-Hill. México. 1987.

7. Ghezzi, Carlo.
   Fundamentals of software engineering.
   Prentice Hall, New Jersey. 1991.

8. Guilera Aguera, Llorenc.
   Introducción a la informática.
   Edunsa. Barcelona. 1988.

9. Hansen, Gary W., Hansen , James V.
   Diseño y Administración de Bases de Datos.
   Prentice Hall. España. 1997.

10. Kevin Yank.
Build Your Own Database Driven Website Using PHP & MySQL.
SitePoint. USA. August 2001 first edition.

11. Marty, Tim. Hartley, Tim.
DB2/SQL A professional Progammer's Guide.
McGraw Hill Book Company. New York. 1989.

12. Pressman, Roger S.
Ingeniería del Software: un enfoque práctico.
McGraw Hill. España. 1998. 3a. edición.

13. Saad, Antonio Miguel.
Manual del redactor.
Diana. México. 1992. 2a. edición.

14. Senn, James A.
Análisis y diseño de sistemas de información.
McGraw Hill. México. 1992.

15. Sommerville, Ian.
Software Engineering.
Addison Wesley. 2002. Harlow. 6a. edición.

16. Wiederhold, Gio.
Diseño de Bases de Datos.
McGraw Hill. México. 1988.

Artículos

17. Boehm, Barry W.
Software engineering As It Is.
Proceedings of the 4th international conference on Software engineering
Redondo Beach, CA.
September 1979.

18. Codd, E.F.
A relational Model of Data for Large Shared Data Banks.
IBM Research Laboratory, San Jose California.
Volume 13. Number 6. June 1970.

19. Royce, W. W.
International Conference on Software Engineering.
Proceedings of the 9th international conference on Software Engineering
Monterrey, California, United States . 1987.
328 – 338 pags.

Tutoriales y guías

20. Gil Rivera, Ma. del Carmen.
Las Bases de datos, importancia y aplicación en la educación.
http://www.cesu.unam.mx/iresie/revistas/ perfiles/perfiles-ant/65-6.htm

21. Manso Martínez, Ma. Esperanza.
Parte I Programación Modular.
U. de Valladolid. Valladolid.
http://www.infor.uva.es/~felix/datos/prii/prog2-modul.pdf

22. Morán y S., Alberto Leopoldo.
Metodología de la Programación II.
España. 2001.

23. Mota Herranz, Laura.
Bases de datos relacionales: teoría y diseño.
Universidad Politécnica de Valencia, Escuela Universitaria de Informática,
Departamento de Sistemas Informáticos y Computación. Valencia. 1994.

24. Rodríguez de Soto, Adolfo.
Metodología y tecnología de la programación. Apuntes de Curso.
Área de lenguajes y sistemas informáticos.
Universidad de León 2001.

25. Romero Guillen, Paola.
Análisis y diseño orientado a objetos.
Instituto Tecnológico de Laguna.
http://www.itlalaguna.edu.mx/academico/carreras/sistemas/
Analisis%20orientado%20a%20objetos/Jacobson.pdf

Ligas en Internet

26. Arreglos RAID.
http://www.viziogames.com/articulos/raid/index.asp

27. Diccionario de la Real Academia Española. DRAE.
http://www.rae.es

28. Diseño del software.
http://grulla.hispalinux.es/enunciados/design_stmas.pdf

29. Diseño de sistemas.
http://pisuerga.inf.ubu.es/icruzado/tfc/OMT_res.pdf

30. HTML
http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/guia_rap.htm#mtable

31. HTML
http://www.um.es/~psibm/tutorial/imagenes.htm

32. Ingeniería del software, conceptos y principios.
http://www.umsanet.edu.bo/docentes/gchoque/MAT426TEX.htm

33. James Rumbaugh capítulo 4 lista de los pasos para el diseño de datos.
http://www.cs.ualberta.ca/~pfiguero/soo/metod/omt.html

34. Java - PHP
http://www.tek271.com/articles/JavaOrPhp.html

35. Modelos RAID.
http://www.agalisa.es/web/article64.html

36. PHP
http://php3.de/manual/es/html/index.html

37. PHP
http://php.net

38. Planeación temporal.
http://148.225.83.24/ic2/ic2apuntes/planeacion.html

39. Procesos del software.
http://www.galileo.edu/wp/display/1480/