UVAQ
UNIVERSIDAD VASCO DE QUIROGA
Repositorio Digital Institucional

REPOSITORIO ACADÉMICO DIGITAL INSTITUCIONAL

# Sistema de atención ciudadana 2000 (análisis y actualización)

**Autor: Patricia Edith Zapien Rodriguez**

**Tesina presentada para obtener el título de:**
**Lic. Sistemas computarizados [sic]**

**Nombre del asesor:**
**Héctor Guerrero Guadarrama**

**Este documento está disponible para su consulta en el Repositorio Académico Digital Institucional de la Universidad Vasco de Quiroga, cuyo objetivo es integrar organizar, almacenar, preservar y difundir en formato digital la producción intelectual resultante de la actividad académica, científica e investigadora de los diferentes campus de la universidad, para beneficio de la comunidad universitaria.**

**Esta iniciativa está a cargo del Centro de Información y Documentación "Dr. Silvio Zavala" que lleva adelante las tareas de gestión y coordinación para la concreción de los objetivos planteados.**

**Esta Tesis se publica bajo licencia Creative Commons de tipo "Reconocimiento-NoComercial-SinObraDerivada", se permite su consulta siempre y cuando se mantenga el reconocimiento de sus autores, no se haga uso comercial de las obras derivadas.**

Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 3.0 Unported.

# UNIVERSIDAD VASCO DE QUIROGA

RECONOCIMIENTO DE VALIDEZ OFICIAL DE ESTUDIOS
POR ACUERDO 952006 DE LA S.E.P. CLAVE 16PSU0014Q

LICENCIATURA EN SISTEMAS COMPUTARIZADOS

## "SISTEMA DE ATENCION CIUDADANA 2000"
(ANALISIS Y ACTUALIZACION)

# TESINA

QUE PARA OBTENER EL TÍTULO DE:

**LICENCIADA EN SISTEMAS COMPUTARIZADOS**


PRESENTA:

**PATRICIA EDITH ZAPIEN RODRIGUEZ**

ASESOR:
ING. HECTOR GUERRERO GUADARRAMA


MORELIA, MICH., FEBRERO DE 2000

# AGRADECIMIENTOS


**A Dios:**

*Por la oportunidad de ser.*
*Por rodearme de familiares y amigos,*
*que me han apoyado incondicionalmente*
*en la elaboración de este trabajo.*

*Con especial cariño y dedicación*
*ofrezco la presente, a* **MIS PADRES:**
*Quienes con su ejemplo, apoyo y*
*consejos me han motivado para*
*alcanzar las metas que me he propuesto*
*durante la vida.*

**Ing. Héctor Guerrero Guadarrama:**

*Gracias por su valiosa aportación y consejos*
*en el transcurso de mi carrera, especialmente*
*en el desarrollo y culminación de este trabajo.*

*Gracias a* **Inelvo Pineda González** *por*
*su paciencia y cariño, por ser un motivo y*
*ejemplo de superación para mí.*

# Contenido

# INTRODUCCION

Dentro del mundo de la computación y en especial al desarrollar un sistema, independientemente de su tamaño y complejidad, se requieren muchas actividades coordinadas y el empleo de una diversidad de herramientas y modelos.

Para desarrollar un sistema se deben establecer y definir los requerimientos iniciales y tener presente que durante su evolución surgirán nuevas necesidades, problemas o sugerencias. Pero lo importante de esto radica en un buen desarrollo y en tratar de reducir continuas modificaciones que muchas veces se siguen presentando aún sobre el sistema puesto en marcha.

No debe olvidarse que los sistemas en uso deben recibir mantenimiento y actualizaciones, y que muchas veces este proceso se complica.

Entre las razones para realizar el análisis de un sistema, se encuentran:

- Mejoras al sistema existente; ajustándose a la estrategia de la organización y proporcionando apoyo a las metas de la misma. La mejora al sistema trae también una mejora en el desempeño gerencial, reducción de costos, emisión de reportes más rápidos y completos.
- Nuevo requerimiento; es decir, nuevos procedimientos para satisfacer ciertas necesidades requeridas, leyes o políticas.
- Aplicación de una nueva idea o tecnología; al realizar una revisión constante del sistema para determinar si existe una nueva forma o componente tecnológico para mejorar su rendimiento o reducir costos.
- *Mantenimiento del sistema en sí o por una falla que requiere corrección.*

La persona que se inicia en el desarrollo de sistemas no percibe las posibles áreas de estudio que rodean esta actividad, dando como resultado remarcadas deficiencias en el desarrollo de software.

En virtud de lo anterior, surge la necesidad de aplicar un análisis completo, que nos permita identificar plenamente las deficiencias o problemas no previstos y por consiguiente, realizar las modificaciones necesarias, tomando en cuenta nuevas tecnologías o modelos de programación que permitan hacer más sencillo el mantenimiento del sistema.

Bajo este enfoque hemos desarrollado el Sistema de Atención Ciudadana (2000), para lo cual presentamos esta lectura en 6 capítulos:

Los dos primeros capítulos están dedicados al análisis del sistema para determinar los requisitos básicos, donde lo más importante es que se vea el alcance del problema, sus repercusiones y características. En esta etapa se llevaron a cabo entrevistas y encuestas que nos dieron información y tamaño del problema real.

En el tercer capítulo se especifica el diseño y estructuración del sistema. Se aplica un modelo orientado a objetos, con el propósito de cubrir de entrada los requisitos, como son la modularidad, adaptabilidad, entre otras.

El capítulo cuatro detalla el desarrollo del sistema. En esta etapa se utilizó el lenguaje Visual Fox Pro, ya que además de cubrir las características básicas de los modelos orientados a objetos, tiene como atributos adicionales su rapidez, facilidad de programación y su excelente interfaz para la etapa de desarrollo.

El quinto capítulo describe el proceso de prueba y mantenimiento efectuado a lo largo del desarrollo del sistema.

Finalmente, en el último apartado se detalla la Bibliografía empleada durante el transcurso del presente trabajo, así como una referencia rápida de estructuras de datos empleadas en el sistema.

# I Identificación de Problemas, Oportunidades y Objetivos

El diseño y análisis de sistemas se emplea para analizar, diseñar y realizar mejoras en el funcionamiento de los negocios, las cuales pueden ser logradas por medio del uso de sistemas de información computarizados. Busca analizar sistemáticamente la entrada de datos o el flujo de datos, el proceso o transformación de los datos, el almacenamiento de datos y la salida de información dentro del contexto de un negocio particular.

Gran parte del análisis y diseño de sistemas involucra el trabajo con los usuarios actuales y eventuales de los sistemas de información.

En esta primera fase se requiere identificar y resaltar los problemas, identificar los objetivos de la organización y finalmente hacer el estudio de factibilidad que contiene una definición del problema y la sumarización de los objetivos. En base a los resultados de esta etapa, los administradores tomarán la decisión de continuar o no con el proyecto propuesto.

## El Problema

Un proyecto de sistemas puede iniciar por muchas causas diferentes, en este caso en particular vemos la necesidad de reconocer oportunidades y hacer mejoras mediante la actualización o realización de nuevos sistemas, pues se requiere adaptarse y enfrentarse al avance que marca la tecnología.

Existen diversos factores internos y externos, que nos permiten observar el desempeño bueno o malo de una organización. Un factor muy importante por considerar, es la *realimentación* sobre qué tan bien está satisfaciendo la Coordinación de Enlace

Ciudadano con sus objetivos planteados. Dicha realimentación ha llegado en forma de queja o sugerencia, pues al parecer la comunidad solicitante exige una atención más rápida y la demanda de apoyos se ha visto disminuida en comparación con años anteriores. Es posible que ésta disminución se deba a los trámites que involucra el llevar a cabo una solicitud.

Aunado a lo anterior, el sistema actual da "facilidades" para que en la captura de las peticiones se cometan un sin número de errores. En algunos casos el trabajo se realiza en forma incompleta y es notoria la insatisfacción, que provoca incluso ausentismo del personal.

Otros factores, íntimamente ligados con el sistema, que afectan en el logro de los objetivos de la Coordinación, los hemos detectado como resultado de las entrevistas efectuadas al personal de dicha institución.

Es necesario realizar entrevistas para recolectar información, obtener la opinión y sentimientos del personal, acerca del estado actual del sistema y los diferentes procedimientos. Para ello seleccionamos personas clave de todos los niveles que serán afectados por el sistema de alguna manera.

La entrevista, conversación basada en preguntas y respuestas abiertas, resultó ser fácil y sencilla por existir la confianza suficiente con los entrevistados, en este caso compañeros de trabajo. Varias de las preguntas que forman parte de las entrevistas, las podemos observar en la fig. 1. Ciertamente algunas personas (enlaces) mostraron gran interés en algunas preguntas, sobre todo cuando se mencionaba la oportunidad de facilitar sus actividades laborales.

El personal de la Coordinación manifestó en las entrevistas algunos comentarios y sugerencias, hemos agrupado y clasificado los más significativos de cada módulo del sistema, siendo estos:

Módulo CAPTURA:

Todos los entrevistados coincidieron en que la captura de datos se realiza de una manera rápida y sencilla. Algunos comentaron que la secuencia de los campos en la pantalla de captura no coincide con la secuencia del formato de codificación, motivo suficiente para confundirse y por consiguiente cometer errores al ingresar los datos.

Módulo de CONSULTA:

Comentan que la consulta actual es eficiente, pero les gustaría tener más alternativas tales como presentación en pantalla de otros catálogos, poder establecer las condiciones o criterios de búsqueda y que la consulta pueda realizarse utilizando mayúsculas o minúsculas (búsqueda fonética).

Módulo de REPORTES:

En general los reportes impresos contienen la información completa requerida, sin embargo, algunos opinan que se podrían simplificar, consideran que podrían ser elaborados en forma más genérica.

Otras sugerencias y comentarios:

Proponen que es conveniente el implementar el uso del "mouse" y cambiar la interfaz del usuario, haciéndola más amigable.

En condiciones de múltiples usuarios, el proceso se torna lento.

Sin embargo, consideramos que el sistema actual es completo y se pueden hacer mejoras en algunos procesos que forman parte del mismo, que nos traerán como resultado aumento de beneficios que bien valen la pena, tales como:

- Aceleración de procesos.
- Eliminación de pasos innecesarios o duplicados.
- Reducción de errores en la entrada por medio de cambios en formas y pantallas.
- Mejora en la integración de módulos o procesos del sistema.
- Mejora de la satisfacción del trabajador o usuario.
- Lograr plenamente los objetivos de la Coordinación.

Algunas mejoras han sido sugeridas por el mismo personal de la dependencia, dichas sugerencias son también de suma importancia pues consideramos que las personas que están en contacto diario con el sistema, pueden ser mejores fuentes de información acerca de las mejoras que se deben hacer; sólo falta determinar si valen la pena y la forma en que pueden ser realizadas.

* En tu posición de (enlace, ejecutivo, operador, etc.), consideras que el sistema SAC facilita tu trabajo dentro de la Coordinación de Enlace Ciudadano?

SI ___ NO ___ En qué medida?

* Consideras que el sistema SAC contribuye en el logro de los objetivos de la Coordinación?

* En los diferentes módulos de (CAPTURA, CONSULTA, etc.), haz notado alguna deficiencia ó algún obstáculo durante su ejecución, que de alguna manera interfiera en la rapidez al desarrollar tus actividades?

* De los diferentes reportes que son emitidos por el sistema, consideras que contienen la información suficiente?

* En términos generales, ¿cuál es tu opinión acerca de SAC?

* Tienes alguna propuesta o sugerencia que consideres impactaría en el desempeño y rapidez del sistema?

* Consideras que al usar un ambiente de trabajo gráfico, es decir, con botones de acción y consultas en ventanas múltiples, agilizaría tu trabajo dentro de la Coordinación?

**Fig. 1 Algunas preguntas clave, planteadas durante las entrevistas.**

## Definición de Objetivos

Recordemos que la Coordinación de Enlace Ciudadano fue creada como dependencia gubernamental, cuya finalidad es dar pronta atención y seguimiento a las demandas o gestiones de apoyo planteadas al Ejecutivo del Estado, por los diferentes sectores necesitados que integran la población michoacana. Para ello se requirió de un sistema computarizado que permitiera llevar a cabo la tarea de una manera más fácil y rápida.

Dichos objetivos se cumplen, pero no de la forma deseada. Estamos seguros que los cambios propuestos impactarán finalmente sobre la organización completa; además cabe mencionar que se cuenta con el respaldo y aprobación de la administración y debemos aprovechar la oportunidad de poder llevarlos a la práctica gracias a los recursos y capacidades con los que cuenta la Coordinación.

**Determinación de la Factibilidad**

El estudio de factibilidad incluye la recopilación de datos para la administración, los cuales les permitirán tomar la decisión sobre si se debe continuar con el análisis y desarrollo del sistema. La factibilidad significa que el proyecto propuesto ayuda a que la CEC (Coordinación de Enlace Ciudadano) logre sus objetivos generales con los recursos actuales de la organización.

Los datos para el estudio de factibilidad serán recolectados por medio de entrevistas, aplicadas principalmente a los administradores o jefes de área. Primeramente, debemos considerar si el proyecto y las propuestas contribuyen verdaderamente en el logro de los objetivos de la organización.

Entre los objetivos que se pretenden alcanzar con este proyecto de sistemas están:

1. Reducir errores y mejorar la precisión de la entrada de datos.
2. Reducir el costo de la salida del sistema mediante la agilización y eliminación de reportes duplicados o innecesarios.
3. Mejorar la atención al cliente.
4. Acortar el tiempo de procesamiento de datos.

Posteriormente, la factibilidad es valorada en 3 formas principales: operacional, técnica y económicamente.

"La factibilidad operacional depende de los recursos humanos disponibles para el proyecto, e involucra proyectar si el sistema operará y será usado una vez que esté instalado". De esta forma consideramos que el proyecto sí es factible, pues el mismo personal ha expresado la necesidad de realizar algunos de los cambios, además pretendemos basarnos en el sistema existente; sólo será modificado y no se desechará por completo. El programa será lo único a desechar.

El proyecto es factible en el área técnica, ya que los recursos técnicos permiten mejorar el sistema existente; además, se cuenta con la tecnología adecuada para satisfacer las especificaciones del sistema.

También es factible económicamente, pues si consideramos el tiempo propio y el del equipo de sistemas, como tiempo invertido en horas de trabajo para realizar funciones que nos corresponden, realmente no constituye un costo adicional, por el contrario representa un mejor aprovechamiento del tiempo.

Por otra parte, el costo de hacer un estudio de sistema completo y el costo del tiempo de los empleados para el estudio no es significativo, pues para ello es aprovechada una mínima parte de su tiempo libre. Además existe gran disponibilidad por parte del personal para aportar con lo que se requiera, en beneficio de la CEC y de ellos mismos.

Así mismo, el costo estimado del hardware y del paquete de software ya está absorbido, pues existe lo básico necesario para la aplicación de una nueva tecnología.

# II Determinación de los Requerimientos de Información

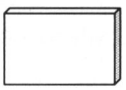
En esta fase se tratará de comprender qué información necesitan los usuarios para realizar su trabajo. Varios de los métodos para determinar los requerimientos de información involucran la interacción directa con los usuarios (entrevistas, cuestionarios), algunos otros son la investigación de datos relevantes, el comportamiento de los tomadores de decisiones y su ambiente de trabajo para conocer los detalles de las funciones actuales del sistema: quién (las personas que están involucradas), qué (la actividad del negocio), dónde (el ambiente donde se lleva a cabo el trabajo), cuándo (en qué momento) y cómo (de qué manera se desarrollan los procedimientos actuales).

Para realizar el análisis de las necesidades del sistema, recurrimos a herramientas y técnicas especiales (Diagramas de Flujo de Datos) que ayudan en la determinación de los requerimientos.

Los Diagramas de Flujo de Datos, representan gráficamente los procesos y el flujo de los datos en un sistema de negocios, es decir, muestran un panorama amplio de posibles entradas, procesos y salidas del sistema.

Un Diagrama de Flujo de Datos, es la conceptualización de la forma en que los datos se mueven a través de la organización, los procesos o transformaciones que sufren los datos y lo que son las salidas.

La simbología utilizada en este tipo de diagramas, está conformada por:

| Símbolo | Significado |
|---|---|
|  | Una *entidad* es una persona, grupo, departamento o cualquier sistema que recibe u origina información o datos. |
|  | Un *flujo de datos* muestra que la información es pasada desde o hacia un proceso, es decir, muestra el movimiento de los datos de un punto a otro. |
|  | Un *proceso* significa que se realizan algunas acciones. Los procesos transforman los datos de entrada en información de salida. |
|  | Representa un *almacén de datos*, muestra solamente un recipiente para los datos que permita adición y recuperación de datos. |

Con la combinación de estos 4 símbolos se puede representar gráficamente el sistema completo, incluyendo los subsistemas.

Haciendo un análisis de procesos que se llevan a cabo desde que se recibe la solicitud de demanda o apoyo, hasta que se envía una respuesta al peticionario, obtenemos el siguiente diagrama de flujo de datos:

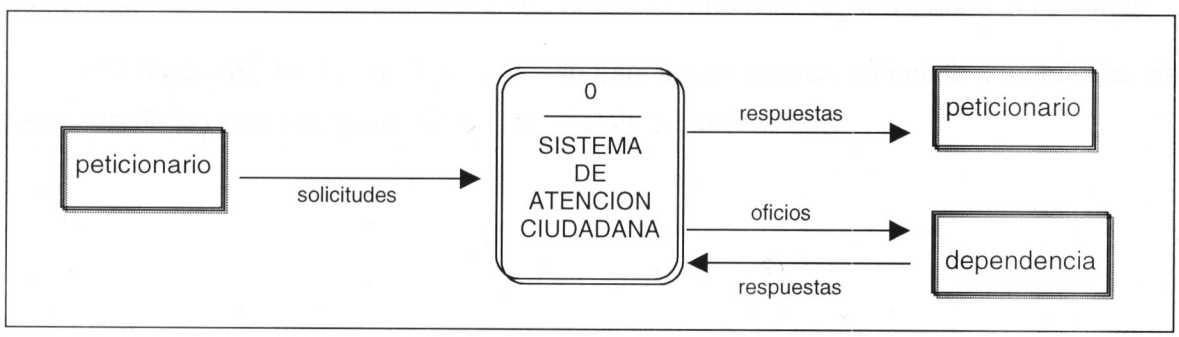



**Fig. 2 Diagrama de Flujo de Datos, identificando las entidades y procesos.**

En el diagrama anterior podemos identificar básicamente 2 entidades externas, "el peticionario" y "la dependencia", personas e instituciones que proporcionan datos al sistema y reciben información del mismo; esta información es resultado de un proceso completo y sistematizado denominado "Sistema de Atención Ciudadana".

Observamos también que las solicitudes o demandas, así como las respuestas y oficios impresos constituyen el tipo de datos que viajan por el sistema, datos que son transformados en los procesos del mismo sistema. El sentido de las flechas nos muestran hacia donde se traslada la información.

En este diagrama tenemos una visión global del sistema, pero es necesario identificar los subsistemas que forman parte del mismo. Desglosando el proceso principal obtenemos un segundo diagrama (fig. 3).

En la nueva representación gráfica (fig. 3) observamos que el sistema original está constituido por 3 procesos internos:

- Clasificación y codificación.
- Registro computarizado de peticiones.
- Impresión de reportes.

y que los datos siguen siendo generados por el peticionario, para finalmente llegar al peticionario y a las dependencias en forma de respuestas y oficios, respectivamente.

Finalmente, obtenemos un tercer diagrama más detallado donde los procesos están plenamente identificados y se incluyen algunas cajas de almacenamiento. Cabe mencionar que las entidades identificadas desde un principio, siguen siendo las mismas.

El diagrama de la fig. 4 es una representación gráfica completa y detallada, de cada uno de los procesos que conforman el sistema actual.

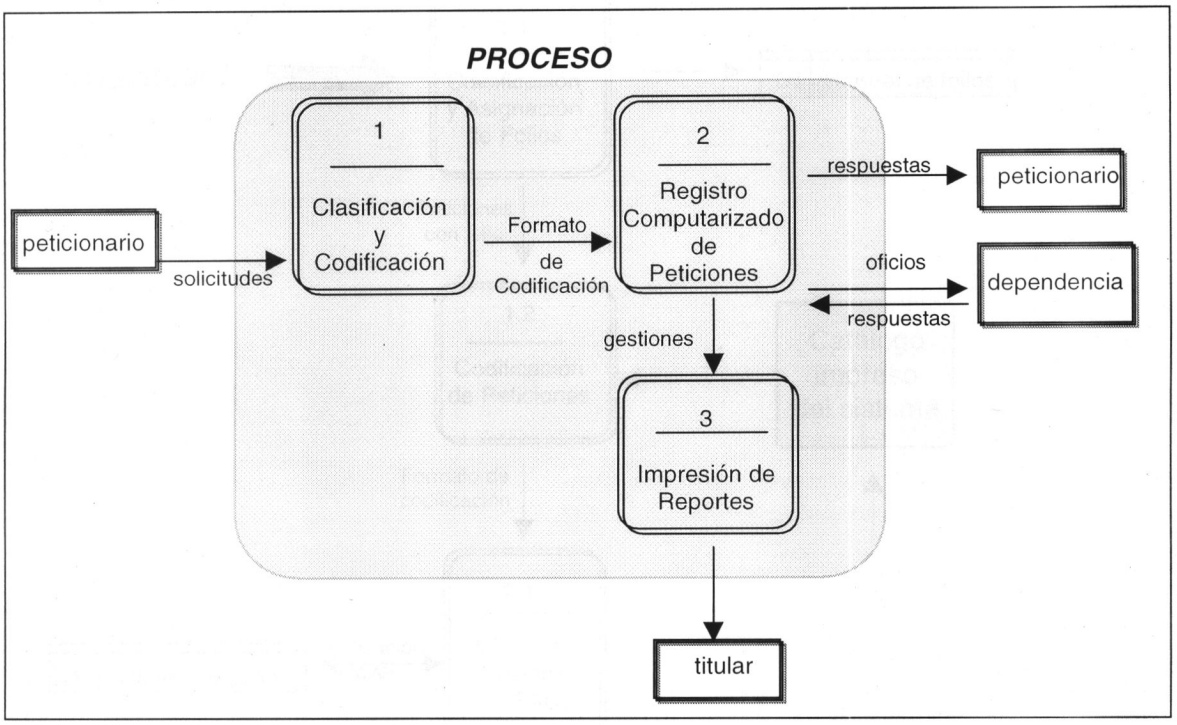



**Fig. 3 Diagrama detallado del flujo de datos del sistema**

## Descripción de los Procesos y Flujo de Datos del Sistema

En el Diagrama de Flujo de Datos final (Fig. 4), las solicitudes de apoyo hechas por los peticionarios son los datos de entrada. Dichas solicitudes recibidas durante gira de trabajo del C. Gobernador o vía correspondencia, deben ser clasificadas para posteriormente asignarles un folio, el cual está formado por 5 dígitos numéricos.

El Control de Folios es un proceso que se realiza manualmente, contiene números consecutivos empezando cada año con el "00001" y agrupados según la forma de captación de las peticiones (gira o correspondencia).

Cada enlace dentro de la Coordinación tiene asignadas determinadas instituciones o dependencias, con el fin de dar seguimiento a las peticiones turnadas a dichas dependencias. Los enlaces se encargan de registrar las solicitudes en el sistema, para ello las peticiones foliadas deben ser codificadas usando un *formato de codificación* (ver fig. 5).

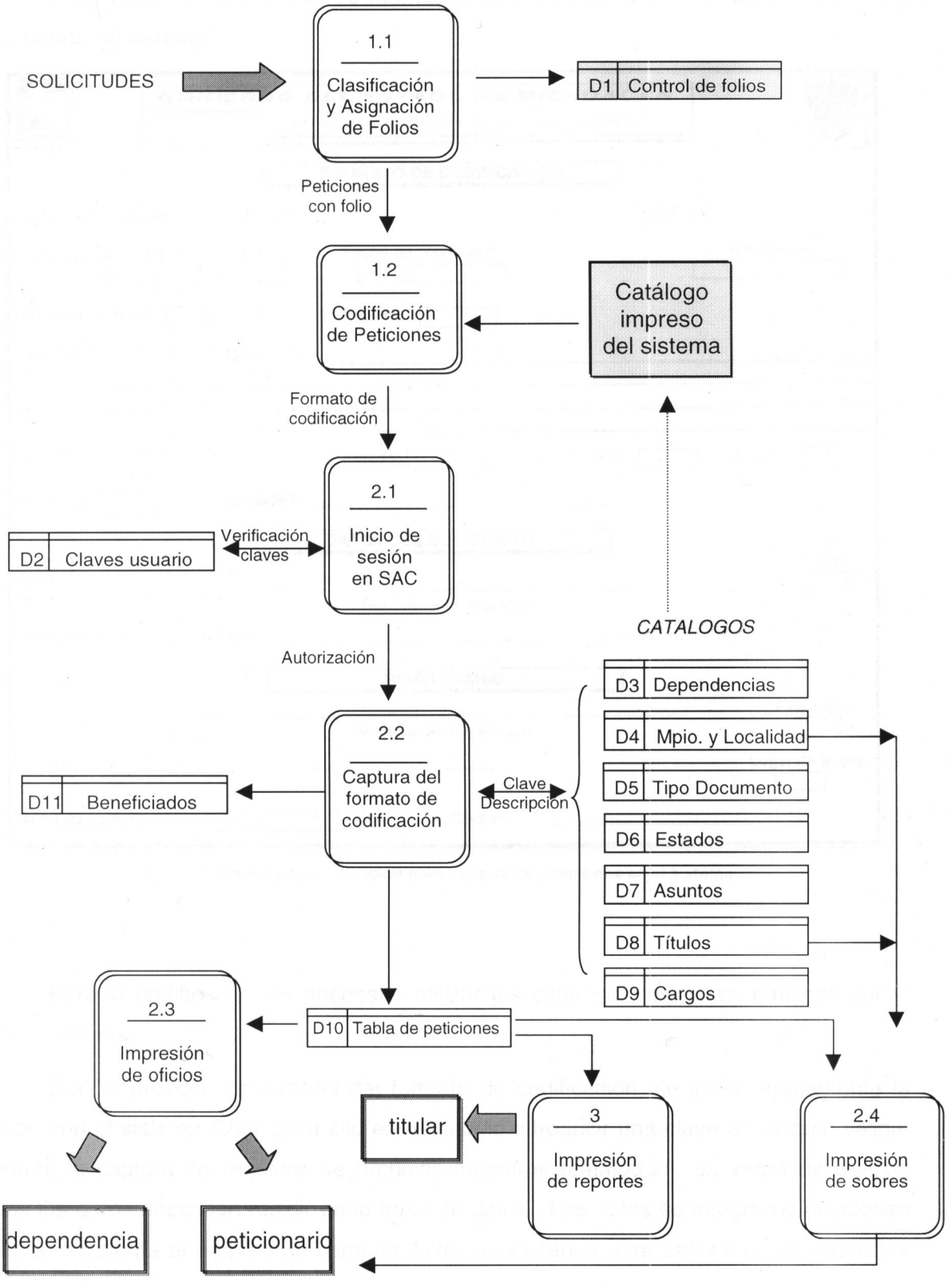



**Fig. 4 Diagrama de flujo de datos final**

El formato está distribuido de manera similar a la secuencia usada en la pantalla de captura del sistema.

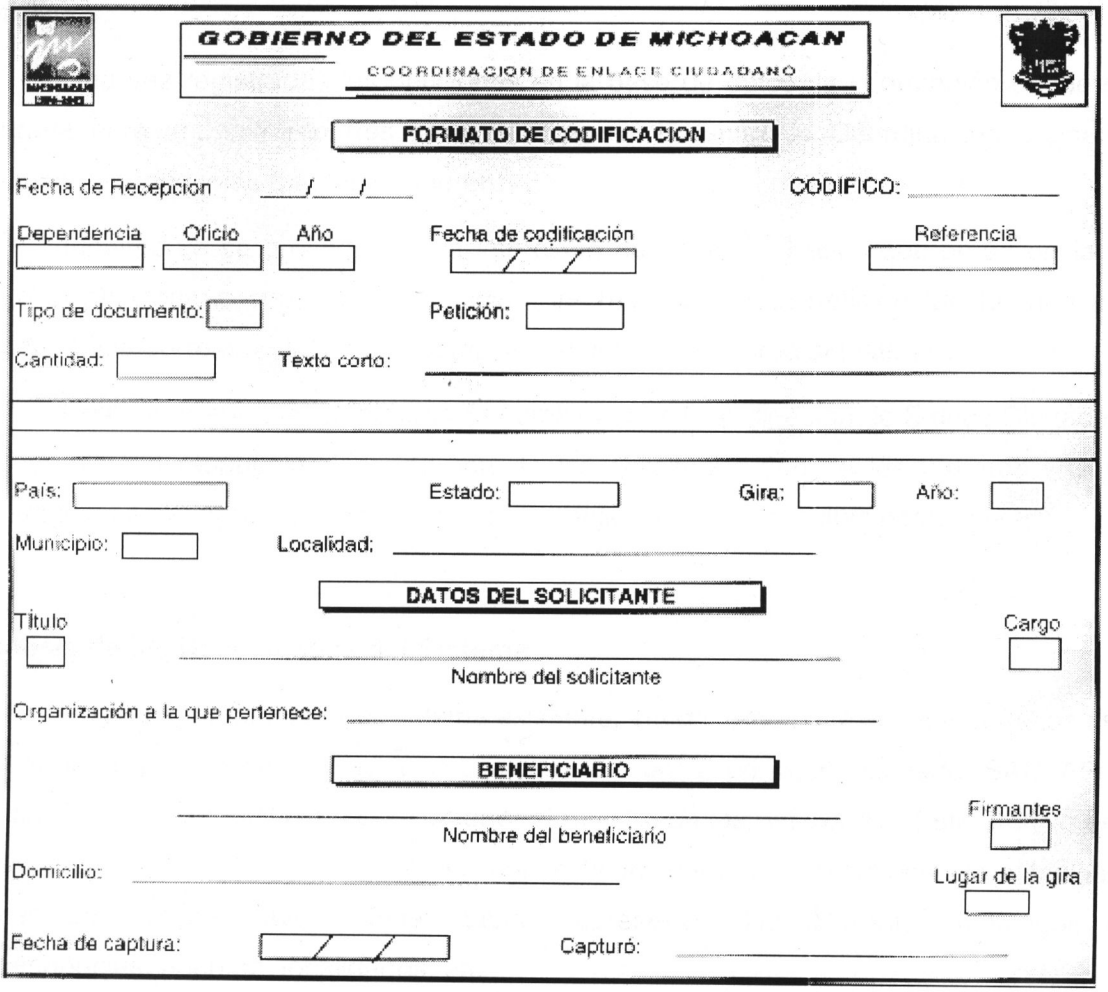



**Fig. 5 Formato de Codificación para captura de peticiones en el sistema**

Para la codificación es necesario utilizar los catálogos impresos, emitidos por el mismo sistema.

Con el proceso de *captura* del formato de codificación, se inicia propiamente la sesión con el sistema SAC; para ello es necesario introducir una clave de acceso válida. Durante la captura se requiere de archivos maestros (catálogos), en virtud de que no todos los datos necesitan residir en la base de datos. Los datos se integran y relacionan entre sí mediante el empleo de campos llave, coordinados para satisfacer los requisitos de los usuarios.

Con la finalidad de recuperar posteriormente las peticiones para consulta o en caso de traspapelarse, se conserva un respaldo digitalizado de cada una de las peticiones.

Una vez registradas las peticiones en la base de datos, la información puede ser utilizada para imprimir reportes solicitados por el titular de la Coordinación o por las mismas dependencias que los requieren.

Finalmente se procede con la impresión de sobres y oficios que serán enviados tanto al peticionario como a la institución, informando que la petición fue turnada a la dependencia correspondiente, así como la situación que guarda actualmente.

Posteriormente, las dependencias emitirán a la Coordinación de Enlace Ciudadano su veredicto o respuesta de cada una de las solicitudes que se les turnaron. Dicha información será ingresada al sistema por los enlaces, para consultas posteriores.

**Análisis de las Necesidades del Sistema**

Haciendo un análisis del sistema existente, encontramos que la base de datos está conformada por varias tablas o archivos. La tabla principal, llamada SAC_AR01, continuamente es modificada por contener el registro de las peticiones. Este archivo está basado en el modelo de datos relacional, está íntimamente ligado con otras tablas a través de campos llave. Estas tablas representan los diferentes catálogos de dependencias, asuntos, localidades, etc.

La estructura y descripción de estos archivos se encuentra detallada en el Apéndice 1.

Algunos campos no son utilizados actualmente, otros dejaron de tener vigencia a raíz de modificaciones en el código fuente.

Otros más contienen información redundante, existe una mínima posibilidad de duplicidad de información.

# III Diseño del Sistema

En virtud de que la mayor parte de los sistemas nuevos en las organizaciones se basan hasta cierto punto en los procedimientos existentes, el primer paso para el diseño de un sistema de información es el análisis del sistema, para posteriormente reunir los datos sobre las operaciones actuales.

Para las operaciones de reunión de datos se utilizaron técnicas y herramientas tales como entrevistas, observaciones y diagrama de flujo del sistema. Lo más importante es saber dónde y cómo van a organizarse los datos.

El objetivo de diseñar la base de datos incluye el almacenamiento eficiente de los datos, así como su actualización y recuperación. Los datos deben ser precisos y consistentes (deben poseer integridad). La información obtenida de los datos almacenados debe estar en un formato útil para la administración, planeación, control, o toma de decisiones.

Una base de datos es un almacén de datos formalmente definido y controlado para ser usado en muchas aplicaciones diferentes.

La efectividad de la base de datos incluye:

- Asegurarse de que la base de datos pueda ser compartida entre los usuarios de una diversidad de aplicaciones.
- Mantener datos que sean precisos y consistentes.
- Asegurarse de que todos los datos requeridos para las aplicaciones actuales y futuras estén fácilmente disponibles.

- Permitir que la base de datos evolucione y que las necesidades de los usuarios crezcan.
- Permitir que los usuarios construyan su vista personal de los datos sin preocuparse de la forma en que estén físicamente guardados.

**Descripción de la Base de Datos**

Los campos que contiene cada tabla se explican por sí mismos, ya que se ha tratado de nombrarlos de la manera más clara posible; consistiendo de una lista con el nombre completo, el tipo y la longitud de cada campo.

La tabla principal la identificamos con el nombre de "peticiones" y el listado del esquema correspondiente se presenta en la siguiente figura:

| Nombre | Tipo | Longitud | Descripción (PETICIONES) |
|---|---|---|---|
| *Tipo* | *Carácter* | *1* | *Identificador del tipo de registro.* |
| *Dep* | *Carácter* | *4* | *Dependencia a la que será turnada la petición.* |
| *Oficio* | *Carácter* | *5* | *Número progresivo de folio.* |
| *Ejercicio* | *Carácter* | *4* | *Ejercicio del registro del formato.* |
| *Referencia* | *Carácter* | *30* | *Referencia relacionada con la persona que gestiona.* |
| *Documento* | *Carácter* | *2* | *Tipo de documento que se va a registrar.* |
| *Petición* | *Carácter* | *4* | *Clave de la Petición.* |
| *Cantidad* | *Numérico* | *4* | *Cantidad de lo solicitado.* |
| *PeticionTx* | *Memo* | | *Texto corto adicional a la petición.* |
| *Pais* | *Carácter* | *15* | *País.* |
| *Estado* | *Carácter* | *2* | *Clave del Estado donde recibirá notificaciones el peticionario.* |
| *Gira* | *Carácter* | *3* | *Número de control de gira realizada por el C. Gobernador.* |
| *Municipio* | *Carácter* | *3* | *Clave del municipio donde radica el peticionario.* |
| *Localidad* | *Carácter* | *4* | *Clave de la localidad donde radica el peticionario.* |

| *Título* | *Carácter* | *2* | *Título del peticionario.* |
|---|---|---|---|
| *NombreSol* | *Carácter* | *75* | *Nombre del peticionario.* |
| *Cargo* | *Carácter* | *3* | *Cargo del peticionario.* |
| *Asociación* | *Carácter* | *50* | *Organización a la que pertenece el peticionario.* |
| *Sexo* | *Carácter* | *1* | *Sexo del peticionario.* |
| *NombreBen* | *Carácter* | *75* | *Nombre del beneficiario.* |
| *Firmantes* | *Carácter* | *3* | *Número de firmantes.* |
| *Domicilio* | *Carácter* | *70* | *Domicilio del peticionario.* |
| *LugarGira* | *Carácter* | *3* | *Clave del municipio donde se realizó la gira.* |
| *FechaCap* | *Fecha* | | *Fecha de captura.* |
| *Capturista* | *Carácter* | *6* | *Iniciales de la persona que capturó la petición.* |
| *ModRes* | *Carácter* | *6* | *Iniciales de la persona que modificó la respuesta.* |
| *ModPet* | *Carácter* | *6* | *Iniciales de la persona que modificó la petición.* |
| *StatusPet* | *Carácter* | *25* | *Status que guarda la petición (positiva, negativa, en proceso, etc.)* |
| *Respuesta* | *Memo* | | *Respuesta emitida por la dependencia.* |
| *Medio* | *Carácter* | *15* | *Programa de radio o t.v.* |
| *ModFecha* | *Fecha* | | *Fecha de última modificación.* |
| *Entrega* | *Fecha* | | *Fecha de entrega para dependencias.* |
| *Vence* | *Fecha* | | *Fecha límite de vencimiento.* |
| *Intento* | *Numérico* | *1* | *Número de recordatorios/intentos de solicitud de respuesta a la dependencia.* |
| *Impreso* | *Fecha* | | *Fecha de impresión de oficios.* |
| *StatusReg* | *Carácter* | *1* | *Status que guarda el registro de la petición.* |

La tabla que contiene el catálogo de "dependencias" es la siguiente:

| Nombre | Tipo | Longitud | Descripción (DEPENDENCIAS) |
|---|---|---|---|
| *Tipo* | *Carácter* | *1* | *Identificador del tipo de registro.* |
| *Clave* | *Carácter* | *4* | *Clave que identifica a cada una de las dependencias.* |
| *Nombre* | *Carácter* | *100* | *Descripción o nombre de las dependencias.* |
| *Domicilio* | *Carácter* | *60* | *Domicilio completo de las oficinas de las diferentes dependencias.* |
| *Teléfonos* | *Carácter* | *20* | *Teléfonos de las dependencias.* |
| *Titular* | *Carácter* | *40* | *Contiene el nombre del titular de la dependencia.* |
| *TitCargo* | *Carácter* | *60* | *Cargo del titular de las dependencias.* |
| *Enlace* | *Carácter* | *40* | *Nombre de la persona-contacto en cada una de las dependencias.* |
| *EnlCargo* | *Carácter* | *40* | *Nombramiento del enlace, dentro de las dependencias.* |
| *Sexo* | *Carácter* | *1* | *Identifica el sexo del enlace de las dependencias.* |
| *Contacto* | *Carácter* | *30* | *Nombre del enlace de la Coordinación.* |
| *Notas* | *Carácter* | *250* | *Contiene las anotaciones hechas por los enlaces, considerando la situación de respuesta por parte de las dependencias.* |

En la siguiente tabla de "varios" están contemplados los datos que pueden corresponder, según el *Tipo*, a los catálogos de títulos, cargos, tipo de documento, asuntos y otros más.

| Nombre | Tipo | Longitud | Descripción (VARIOS) |
|---|---|---|---|
| *Tipo* | *Carácter* | *1* | *Identificador del tipo de registro o catálogo a consultar (títulos, cargos, tipo de documento, beneficiarios, asuntos, estados).* |
| *Clave* | *Carácter* | *5* | *Clave que identifica a los elementos de cada catálogo.* |
| *Nombre* | *Carácter* | *100* | *Descripción de cada elemento de los catálogos.* |
| *Texto* | *Memo* | | *Contiene texto que forma parte de los oficios impresos.* |

La siguiente tabla corresponde a la de los "municipios", donde también podemos encontrar la lista de localidades y distritos, según la *clave* del municipio:

| Nombre | Tipo | Longitud | Descripción (MUNICIPIOS) |
|---|---|---|---|
| *Tipo* | *Carácter* | *1* | *Identificador del tipo de registro, si se trata de distrito, municipio o localidad.* |
| *Clave* | *Carácter* | *3* | *Clave que identifica al municipio.* |
| *LocDis* | *Carácter* | *4* | *Clave de la localidad o distrito para el municipio "X".* |
| *Nombre* | *Carácter* | *30* | *Contiene la descripción o nombre del distrito, municipio o localidad.* |
| *CodPos* | *Carácter* | *5* | *Código postal de cada municipio.* |

Por último tenemos una tabla de control, la cual contiene las claves de "accesos" de los usuarios, válidas para el sistema:

| Nombre | Tipo | Longitud | Descripción (ACCESOS) |
|---|---|---|---|
| *Tipo* | *Carácter* | *1* | *Identificador del tipo de registro.* |
| *Usuario* | *Carácter* | *6* | *Clave del usuario que tiene acceso al sistema.* |
| *Nombre* | *Carácter* | *35* | *Contiene el nombre del usuario.* |
| *Iniciales* | *Carácter* | *6* | *Contiene las iniciales del usuario.* |
| *Opc_1_1* | *Lógico* | *1* | *Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene acceso o no a ésta opción del menú.* |
| *Opc_1_2* | *Lógico* | *1* | *Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene acceso o no a ésta opción del menú.* |
| *Opc_2_1* | *Lógico* | *1* | *Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene acceso o no a ésta opción del menú.* |
| *Opc_2_2* | *Lógico* | *1* | *Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene acceso o no a ésta opción del menú.* |
| *Opc_2_3* | *Lógico* | *1* | *Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene acceso o no a ésta opción del menú.* |
| * | * | * | * |

**Diagrama de Procesos del Sistema**

Una Base de Datos es una colección integrada de datos, almacenados en diferentes archivos. Una *tabla* es un conjunto de datos que tienen entre sí algo en común. En el caso de una base de datos relacional, los aspectos comunes están organizados mediante campos, y estos a su vez forman registros.

En la siguiente representación gráfica (fig. 6), podemos observar que la base de datos del sistema está integradas por cuatro tablas básicas: Peticiones, Varios,

Dependencias y Municipios. Además está contemplada una tabla adicional para organizar y controlar las claves de acceso.

Las tablas están relacionadas entre sí, pero a la vez independientes; pueden ser consultadas y modificadas aún fuera de la captura de peticiones, y pueden ser utilizadas en procesos externos al sistema.

El almacenamiento de datos es bastante sencillo. Lo más importante que debe entenderse acerca de las Bases de Datos, es que los datos deben integrarse y relacionarse entre sí mediante el empleo de campos llave coordinados, para satisfacer los requerimientos de los usuarios.

Una base de datos relacional consiste en una colección de *tablas*, a cada una de las cuales se asigna un nombre único.

El modelo relacional representa los datos y las relaciones entre los datos mediante una colección de tablas. Este modelo conecta registros mediante los valores que éstos contienen.

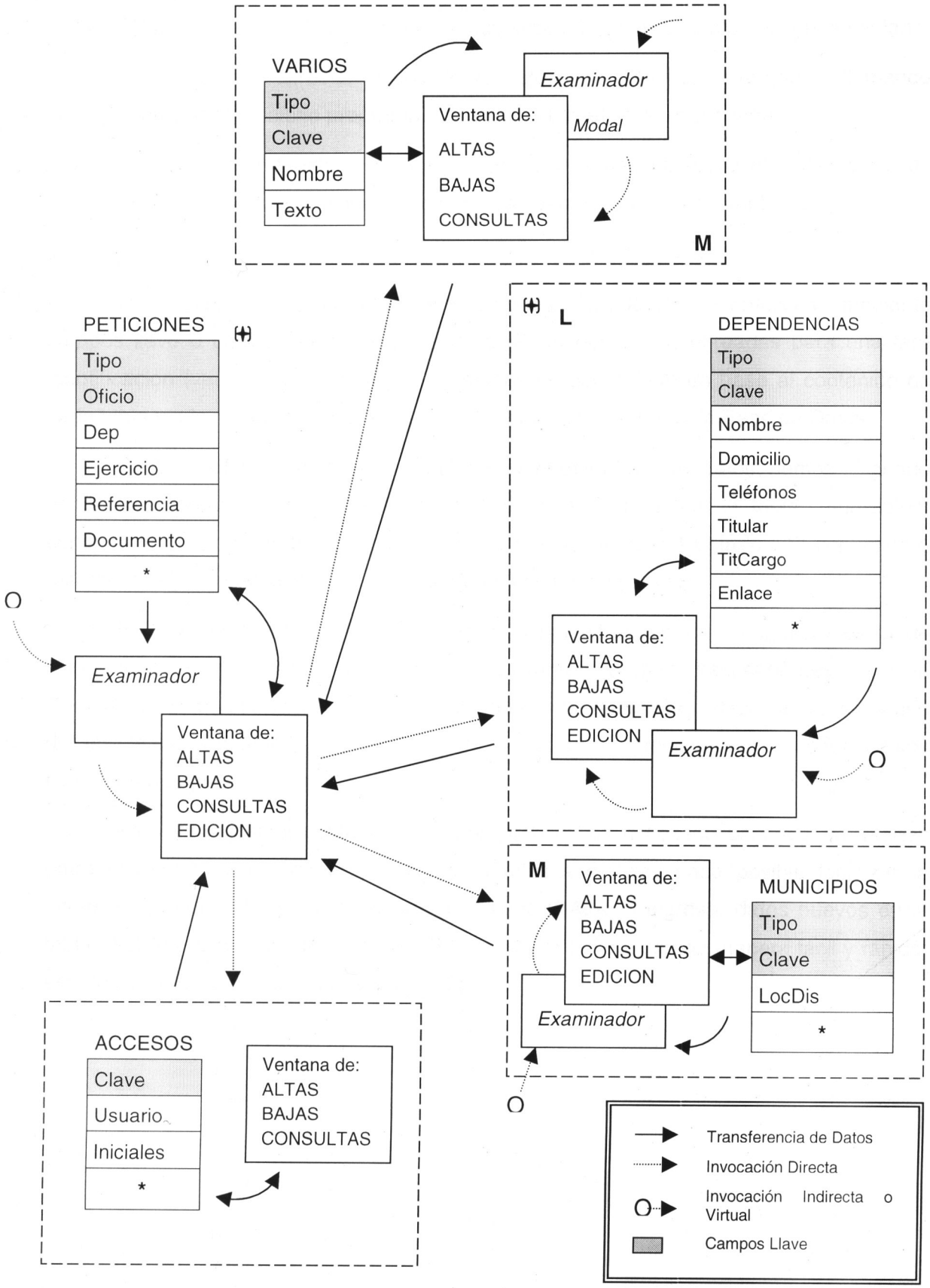
El modelo de datos relacional, es el modelo más utilizado entre los sistemas de bases de datos de las empresas, en virtud de que:

- Las bases de datos relacionales son conceptualmente muy simples y fáciles de entender.
- Las relaciones quedan implícitas en los valores de los datos.
- Puesto que las relaciones no requieren ser predefinidas, la base de datos puede transformarse para satisfacer condiciones cambiantes.

Para permitir el acceso aleatorio rápido a los registros de un archivo, se utiliza una estructura de índice. Cada estructura de índice está asociada con una clave de búsqueda determinada.

Elegimos incluir varios índices en diferentes claves de búsqueda, el índice cuya clave de búsqueda especifica el orden del archivo es el *índice primario*; los demás se llaman *índices secundarios*. La clave de búsqueda de un índice primario, es normalmente la clave primaria.

**Fig. 6 Diagrama de procesos del sistema**



En el esquema anterior (fig. 6):

* Transferencia de Datos, significa recuperar o guardar desde o hacia una ventana.

* Invocación Directa, significa la ejecución de un programa llamando explícitamente dicho programa (construye el objeto que lo contiene).

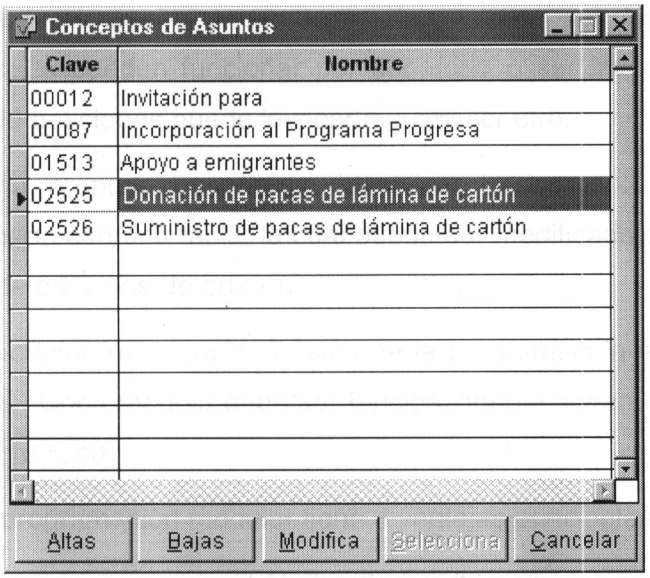
* Invocación Indirecta o Virtual, significa que el usuario utiliza el menú, barra de herramientas o la reactivación con mouse para despertar una ventana.

En la Base de Datos del sistema propuesto, las tablas se relacionan mediante campos llave o claves primarias, las cuales se encuentran sombreadas para una fácil identificación (ver fig. 6). Las claves primarias se determinan en base al contenido de cada tabla y a la relación que tendrá con otros archivos, dentro de la Base de Datos.

Para identificar cada una de las tablas de nuestra base de datos, hemos asignado un nombre tomando en cuenta su contenido; de tal forma que la tabla "municipios" contiene la clave y nombre de los municipios y localidades del Estado. El contenido y estructura de cada tabla lo hemos detallado en el capítulo anterior.

Es importante aclarar que la tabla principal en el sistema propuesto es la de *peticiones*, ya que contiene los datos relevantes y más trascendentes para la Coordinación. Es el archivo más grande de la base de datos y se relaciona con cada una de las tablas restantes, es decir, acude a los otros archivos para recabar datos complementarios.

Para visualizar el contenido de cada archivo, es necesario usar alguna interfaz entre el usuario y el archivo; el uso de pantallas gráficas hace posible localizar la información requerida, permite eliminar los datos (BAJA) o ingresar datos nuevos en la tabla (ALTA) o simplemente para MODIFICAR el contenido, de una manera fácil y rápida; esto lo podemos apreciar en la siguiente fig.:




Estas múltiples opciones forman parte de las pantallas de consulta (examinador). Cabe mencionar que se usará una plataforma de lanzamiento para las ventanas, la cual puede ser de tipo "Modal " o "Libre / No Modal"; también es importante señalar que para efectos de arranque y demostración usaremos tablas libres, aunque la intención posterior será la de utilizar tablas relacionadas.

Los examinadores (browsers) de las diferentes tablas contienen información relevante, son presentados al usuario como de sólo lectura o consulta y serán de tipo *libre*, para permitir la interacción con otras ventanas o pantallas. Son módulos que obedecen a eventos.

En cambio, las pantallas de edición o diálogo serán lanzadas como *modales*, es decir, se requiere de modificar algún dato o ingresar otros, por lo cual no permitirá salir de esa pantalla hasta no haberse registrado una confirmación o en su defecto una cancelación por parte del usuario. Este modo inhibe los menús y desactiva temporalmente otras ventanas.

En el diagrama podemos observar que entre las ventanas de edición y las tablas, hay una transferencia de datos en ambos sentidos, esto es, cuando se realiza una consulta la transferencia es de la tabla hacia las ventanas, pero cuando se efectúa una modificación a la tabla (alta, baja o edición), se transfieren datos hacia la tabla.

Para simplificar el diseño y desarrollo del sistema se dividió en módulos o subsistemas, los cuales pueden funcionar por separado o ser invocados por otro. Por ejemplo, el módulo de peticiones puede invocar a cualquier otro.

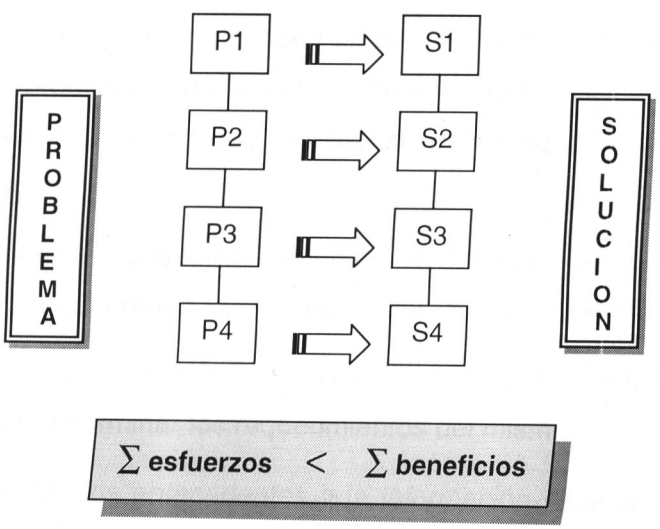
En el sistema propuesto podemos identificar básicamente 5 módulos. Cada módulo está relacionado con una tabla, la cual puede ser modificada o examinada por los usuarios por medio de pantallas de edición.

El enfoque modular involucra la división de la programación en partes o módulos lógicos y manejables. Cada módulo debe ser funcionalmente coherente, por lo que debe realizar determinada función.

El diseño de programación modular tiene 3 ventajas principales:

1. Los módulos son más fáciles de escribir y depurar. Un problema o error en un módulo no debe causar problemas a otros.
2. Los módulos son más fáciles de mantener. Las modificaciones estarán limitadas a unos cuantos módulos y no estarán dispersas por todo el programa completo.
3. Los módulos son más fáciles de entender, debido a que son subsistemas autocontenidos.

El sistema propuesto es sinérgico, ya que se tienen beneficios o ganancias. El beneficio final será mayor que el esfuerzo aplicado en sus partes (módulos), lo cual se refleja en la facilidad de documentación, facilidad de operatividad y mantenimiento del sistema.



# IV Desarrollo

Esta etapa es la que requiere más documentación. Por lo general los programadores o desarrolladores hacemos algunas consideraciones y especulaciones que se nos olvida documentar. Dichas omisiones repercuten tanto en el desempeño de los programadores o analistas futuros, y consecuentemente en la funcionalidad del mismo sistema.

La mayoría de los sistemas no son considerados desechables y algunos necesitarán recibir mantenimiento, por lo que se requiere que los programas estén documentados adecuadamente. Se debe documentar el software para que los programadores y analistas conozcan el sistema internamente sin necesidad de interactuar con él; la documentación de los procedimientos es necesaria para la nueva gente que aprende el sistema y como recordatorio para aquellos que usan el programa con poca frecuencia.

Alguna documentación proporciona un panorama del sistema mismo y, en cambio, la documentación procedural da detalles sobre lo que debe hacerse para ejecutar software en el sistema y la documentación del programa detalla el código de programación usado.

La documentación consistente, acortará la cantidad de horas requeridas por el nuevo personal para que aprendan el sistema antes de realizar el mantenimiento.

El conocimiento más profundo del sistema aunado a la documentación existente, facilitó el camino para determinar los requerimientos del mismo.

Una vez reunidos los antecedentes y la información necesaria para el desarrollo del sistema, consideramos que se necesita utilizar un software que facilite el diseño de interfaces gráficas. Otro punto importante a considerar en la selección del software

adecuado, es la facilidad que ofrece para recuperar y manipular datos de una base de datos.

Los lenguajes "*orientados a objetos*" ofrecen nuevas estructuras que facilitan el mantenimiento del programa y hacen que grandes partes de los programas sean reutilizables; como consecuencia del reciclado de partes de programa se reducen los costos de desarrollo de sistemas, además las técnicas orientadas a objetos permiten el mantenimiento, adaptaciones y rediseño a los sistemas de información.

**Programación Orientada a Objetos (POO)**

El sistema de programación orientada a objetos, permite una jerarquía abstracta y modular, así como las características de polimorfismo, herencia y encapsulamiento.

Otras facilidades que nos proporciona la Programación Orientada a Objetos:

- Reutilización de código.
- Modificar de una sola vez, varias partes de una aplicación.
- Adaptar código a otras aplicaciones diferentes.
- Interactuar con el usuario y controlar mejor sus acciones.
- Eliminar redundancias de código, entre otras.

Sin lugar a duda, el realizar algunas aplicaciones (conjunto de objetos, formularios, controles) y tener código para ser reutilizado, facilitará el desarrollo de nuevas aplicaciones en menos tiempo y con más fiabilidad, con esto la Programación Orientada a Objetos comienza a ser rentable.

Siete ideas o conceptos básicos caracterizan a la POO:

- Clases
- Objetos
- Propiedades, Métodos y Eventos
- Encapsulación
- Herencia
- Polimorfismo
- *Abstracción*

### *Clases*

Las clases y los objetos están estrechamente relacionados, pero no son lo mismo. Una clase contiene información sobre cuál debe ser la apariencia y el comportamiento de un objeto. Las clases son intangibles, es la abstracción, plano o esquema de un objeto. A partir de una clase se puede crear un objeto, es decir, primero es el diseño o modelo. Los objetos se agrupan en clases. Una clase es una categoría de objetos similares.

Una clase define el conjunto de atributos y comportamientos compartidos que se encuentran en cada objeto de la clase, es decir, define las características de un objeto y describe qué apariencia y comportamiento debe tener el objeto.

La Biblioteca de Clases de Visual FoxPro facilita la tarea de diseño de pantallas, contiene figuras, botones de comandos y otras clases, cuyas propiedades pueden ser modificadas, cambiando de esta forma su apariencia.

### *Objetos*

La representación en computadora de alguna cosa o evento del mundo real, la podemos llamar objeto.

Cada objeto tiene *atributos* y *comportamientos*, es decir, las pantallas o formularios o cualquier otro objeto tienen propiedades que están determinadas por la clase en la que se basa el objeto, las cuales establecerán la apariencia, posición, tamaño y color, junto con aspectos de su comportamiento (¿su tamaño es ajustable?).

Un objeto es algo tangible con características y un comportamiento; es creado a partir de una clase (instancia de clase) que combina datos y procedimientos.

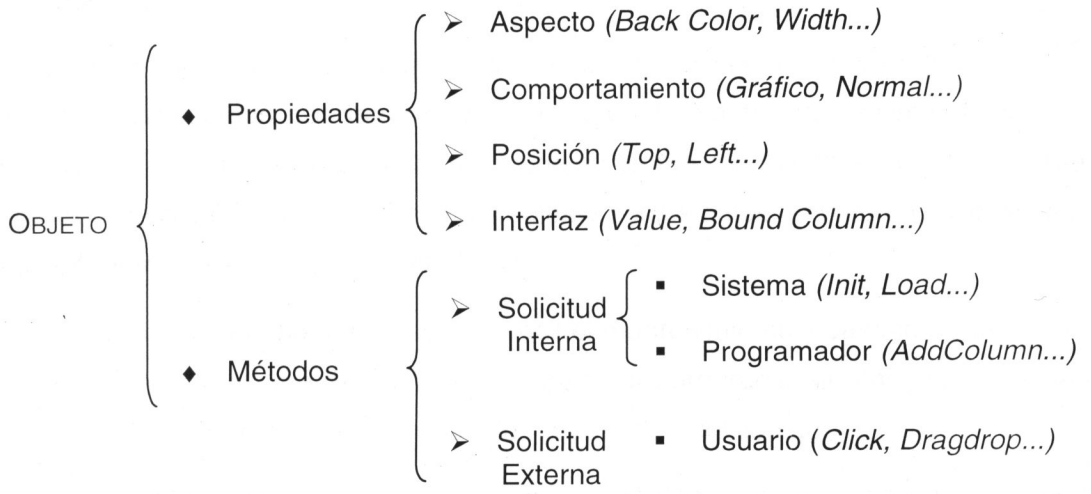
Con varios objetos es posible construir pequeños módulos a los que se puede invocar en cualquier momento.

### *Propiedades, Métodos y Eventos*

Los objetos tienen *propiedades* que determinan su aspecto, posición, comportamiento, y métodos específicos que pueden ser desencadenados interna o externamente.

Las propiedades son las características o atributos de un objeto, que definen su aspecto, comportamiento o posición; mientras que los métodos son procedimientos que

pueden activar otros procedimientos o acciones mediante una solicitud del sistema o por un mensaje de un objeto a otro. Los procedimientos que son desencadenados por otros procedimientos son llamados *eventos*.




Los objetos pueden ser desencadenados y modificados tanto por solicitud interna como externa. Internamente al solicitar la creación de un objeto se activa el procedimiento Load Event (justo antes de crear un objeto) e Init Event (ocurre cuando se crea el objeto). Otros procedimientos son desencadenados por el programador desde el código de programa, los cuales pueden cambiar el estado de los mismos objetos.

Los objetos pueden ser modificados externamente por el usuario, por ejemplo al presionar algún botón del mouse o al cambiar el tamaño de una ventana, se activará el procedimiento *Click Event*. Estas modificaciones son también consideradas eventos por ser desencadenadas por otra acción.

Si esto lo aplicamos a la programación, los eventos serán acciones del usuario sobre nuestra aplicación y los métodos serán los procedimientos asociados a un objeto. Los eventos pueden estar generados por una acción del usuario, como hacer clic con el mouse o presionar una tecla.

Los métodos permiten modificar el efecto o función que realizará cada objeto. Los métodos pueden existir independientemente de los eventos.

Los formularios también pueden responder a eventos iniciados por un usuario o desencadenados por el sistema (cambiar el color del formulario al hacer clic en él).

*Encapsulación*

La encapsulación es un término de la POO que se aplica a la posibilidad de proteger y empaquetar tanto las propiedades como el código de un objeto. El encapsulamiento aísla la complejidad interna del funcionamiento de un objeto del resto de la aplicación.

La información acerca de un objeto, está encapsulada por su comportamiento. Los datos encapsulados pueden ser protegidos en forma tal que solamente el objeto mismo puede hacer tales cambios por medio de su propio comportamiento. Esto facilita la construcción de objetos que son muy confiables y consistentes, debido a que ellos tienen control completo sobre sus propios atributos.

La encapsulación hace mucho más fácil el mantenimiento y cambio de programas; con este aislamiento se puede cambiar una parte del programa sin causar problemas a otras partes del programa, lo que ayuda a la reutilización del código.

Al crear clases, lo que estamos haciendo implícitamente, es agrupar propiedades y métodos para que cuando tengamos que hacer otra aplicación, tomemos la clase y hacemos uso de ella en la nueva aplicación sin tener que cambiar nada.

*Herencia*

Es la posibilidad de que una clase adopte las características de la clase en la que se basa. Si las características de la clase primaria cambian, la clase derivada heredará dichas características.

Una clase puede ser creada a partir de otra. La clase original, o madre, es llamada "clase base". La clase hija es llamada "clase derivada".

Una clase derivada puede heredar todos los atributos y comportamientos de la clase base, aunque puede tener también atributos y comportamientos adicionales, sobreescribiendo determinado método o propiedad que no nos sirva.

Las clases u objetos derivados, tienen las propiedades y métodos definidos en la clase base, por lo que no tendremos que repetirlos, esto ahorra tiempo y trabajo.

La herencia reduce la labor de programación, reutilizando fácilmente objetos antiguos; es decir, hay que poner el código en una clase y, sólo con esto, todas las clases

y objetos creados partiendo de esa clase tendrán ese código, sin tener por tanto que reescribirlo.

De igual forma, al modificar alguna propiedad de un objeto, sólo bastará con corregir la propiedad en la clase base o el método que está mal, y el cambio se reflejará en todas las clases y los objetos creados a partir de dicha clase.

Otra característica de la herencia es que las propiedades y métodos quedan protegidos, impidiendo que sean modificados en algún sitio que no sea la clase base.

*Polimorfismo*

Esta característica de la POO, se refiere a la posibilidad de tener métodos con el mismo nombre, pero distinto contenido, para clases relacionadas. Consiste en que objetos, aunque partan de distintas clases, tengan métodos o propiedades con el mismo nombre y que actúen de distinta forma.

Permite que alguna clase que heredó atributos y comportamientos de otra clase, se comporte de diferente manera a su clase base o de sus clases derivadas parientes.

Tiene relación directa con la herencia, ya que un método o propiedad heredado se puede sobreescribir y por tanto actuar de distinta forma de la que hacía en la clase base.

*Abstracción*

Es una característica que nos da la ventaja de ignorar los detalles internos de un objeto, para poder centrarse en los aspectos del objeto que necesitamos utilizar. Con la abstracción, no importa cómo está hecha una clase u objeto internamente.

Consiste en identificar las características distintivas de una clase u objeto sin tener que procesar toda la información sobre dicha clase u objeto. Cuando se crea una clase, por ejemplo una serie de botones para desplazamiento por tablas, puede usarla como una única entidad en lugar de hacer un seguimiento de los componentes individuales y cómo interactúan.

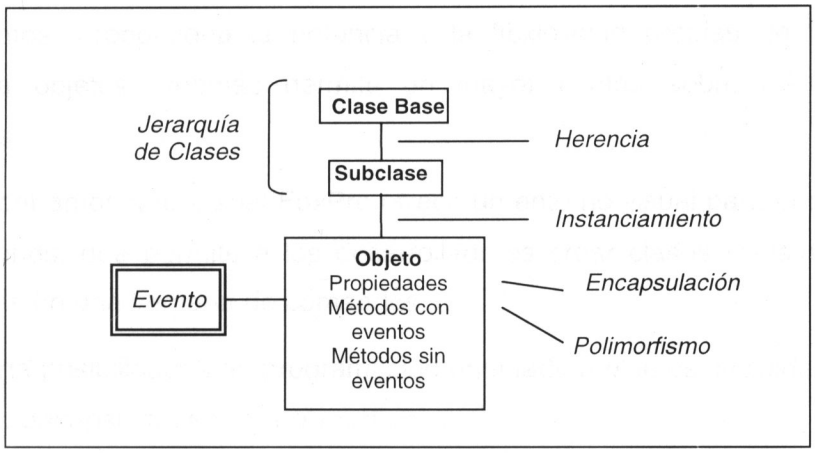



**Fig. 7 Componentes de la POO**

## Visual FoxPro

En virtud de que el problema a resolver está bien definido, hemos buscado diferentes herramientas que nos ayuden a resolverlo. Necesitamos un lenguaje de programación que permita el desarrollo de aplicaciones gráficas, que en tiempo de ejecución sea rápido, que sea fácil de usar y fácil de conseguir, entre otras.

Algunos lenguajes, si no es que la mayoría, permiten desarrollar aplicaciones gráficas, otros más lo hacen de una manera rápida; sin embargo, también debemos considerar la facilidad pues finalmente nosotros haremos el trabajo. Para qué complicarnos la existencia tratando de usar herramientas no adecuadas, que lejos de ayudarnos nos dificultan la solución del problema.

Entonces, por tratarse de una necesidad específica (manejo de gran volumen de datos), nuestras herramientas se van limitando a aquellos lenguajes que permiten manipular bases de datos, de entre estos últimos debemos seleccionar el que permita realizar el desarrollo *más fácilmente* y más atractivo para el usuario final.

De lo anterior consideramos que el lenguaje de programación más apto a nuestras necesidades es Visual FoxPro, es una nueva propuesta de tecnología en lo que se refiere a software. Ahora bien, ¿por qué no programar en otro lenguaje visual?.

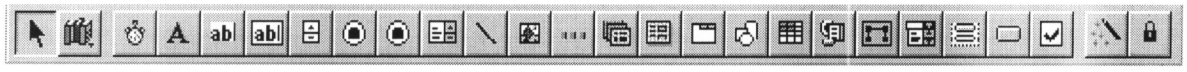
Aunque Visual FoxPro sigue permitiendo la programación estándar por procedimientos, proporciona la potencia y la flexibilidad propias de la programación orientada a objetos, además permite un mayor control sobre los objetos de las aplicaciones.

Encontramos que Visual FoxPro ofrece un entorno visual para el desarrollo rápido de aplicaciones, que permite a los desarrolladores crear clases visualmente, así como programarlas en una ventana de comandos.

Aporta posibilidades de programación orientada a objetos, incluidas características de herencia, encapsulamiento y polimorfismo.

Proporciona una depuración de funciones completas con variables de vigilancia, ventana de seguimiento y puntos de ruptura.

Incluye un conjunto de controles estándar, incluidos los siguientes: Cuadrícula, Etiqueta, Casilla de verificación, Cuadro combinado, Botón de comando, Cuadro de lista, Forma y Marco de página.



Contiene un visualizador de clases incorporado que permite crear, manipular y reutilizar clases en una aplicación.

Permite crear archivos ejecutables ( .EXE ) independientes, que se pueden distribuir de manera ilimitada.

Incluye controladores de escritorio ODBC (*Conectividad Abierta de Bases de Datos)*, protocolo estándar, que permiten a VFP conectarse a las bases de datos y tener acceso a los datos Btrieve, dBASE, Microsoft Excel, Paradox, Microsoft Access y otros datos de FoxPro.

Antes de comenzar la programación, hemos determinado que la técnica de diseño a utilizar será la *modularidad*, con el fin de dividir el esfuerzo de programación en módulos manejables. En lugar de pensar en el flujo del programa desde la primera hasta la última línea de código, se debe pensar en la creación de objetos: componentes que tienen funcionalidad privada.

Hay diferentes módulos, algunos ejecutarán una sola tarea (impresión de reportes), aunque puede tener varias tareas secundarias (impresión de reportes

calculando totales para cada municipio); un módulo de control (acceso a cierta información) y, módulos de funciones (edición de catálogos) que serán los responsables del desplegado de las pantallas.

El módulo de claves de acceso (control) contiene la lógica para ejecutar módulos de otros niveles. Por lo general la lógica de control es la más difícil de diseñar, por lo tanto no debe ser de tamaño muy grande.

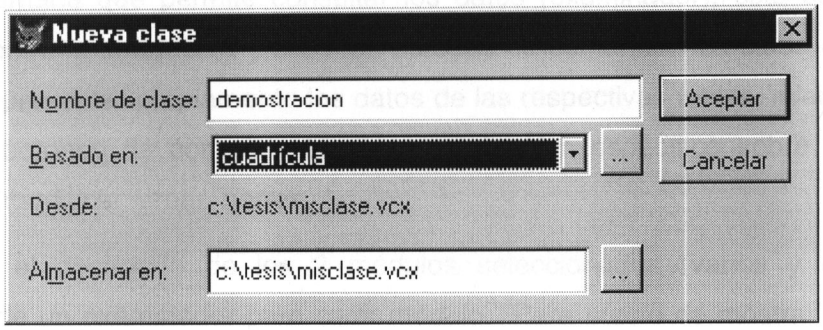
El sistema propuesto está constituido por varios archivos de programa:

| | |
|---|---|
| Sac.exe | Archivo de arranque o *principal* |
| SacCrea.prg | Permite crear la estructura de las tablas de base de datos al iniciar el ejercicio. |
| SacUsa.prg | Activa las diferentes tablas de la base de datos. |
| SacIni.prg | Su función es la de cargar la configuración de la aplicación. |
| SacPro.prg | Contiene los procedimientos que se utilizarán durante el desarrollo del sistema. |
| SacClase.vcx | Es la biblioteca de clases, contiene el resto del código. |

El programa *sacini.prg* es corto y sencillo, contiene una sección donde se limpia la pantalla, se cierran todo tipo de archivos y se libera la memoria, se establece la biblioteca de clases con la que va a funcionar la aplicación, se crean las variables públicas y se abren o activan las tablas de datos con sus respectivos índices y alias, también se establece que los diferentes procedimientos se buscarán en el archivo *sacpro.prg*.

**Clases**

*Las clases de VFP nos facilitarán el diseño de subclases personalizadas, ya que podemos modificar las propiedades de algunas de ellas y crear nuestra propia biblioteca de clases que usaremos en nuestra aplicación o sistema. Crear subclases y conservarlas en una biblioteca reduce la cantidad de código que hay que escribir. Todas las clases diseñadas se almacenan en una biblioteca de clases con la extensión de archivo .VCX.*




En la biblioteca de clases (*misclases.vcx*) están definidas las diferentes clases basadas en las de VFP, con características específicas que serán usadas en varias aplicaciones, de tal forma que si voy a utilizar un botón de comando llamado *Salir* basta con especificar sus propiedades y código de los métodos, y simplemente tomar la clase e insertarla en el formulario deseado, sin necesidad de reescribir el código cada vez que utilice esa clase.

Aunque no pongamos código en ellas, podemos darles el aspecto que se requiere y en caso de no gustarnos, lo cambiamos; gracias a la herencia se cambiará en todos los lugares donde se hayan creado objetos derivados de esa clase. Creamos clases con una apariencia característica y la almacenamos dentro de una biblioteca, de tal modo que al colocarlas en una aplicación tendrán la misma apariencia que la clase base, sin necesidad de redefinir sus propiedades y métodos cada vez que sean utilizadas.

El archivo *sacclase.vcx* contiene objetos o formularios diseñados específicamente para el proyecto, con apariencia de pantallas de captura o edición. Estos nuevos objetos o clases están constituidos con clases basadas en la biblioteca *misclases.vcx*.

Para identificar a cada uno de los objetos o clases se les asigna un nombre, normalmente se recomienda usar la notación húngara que consiste en agregar al nombre de los objetos un prefijo de 2 caracteres, lo cual nos facilitará la identificación del tipo de clase base (ejemplo: a un nuevo objeto llamado clave y basado en la clase textbox, lo llamaremos tbClave).

Tenemos así algunos examinadores o browsers, que por sus características de presentar la información en columnas y filas, la clase que nos facilita el diseño es un **grid** o **cuadrícula**. Esta clase tiene varias propiedades y métodos que permiten establecer el número de columnas a presentar, así como los encabezados de cada columna y el tipo de fuente tanto del encabezado como del contenido, entre otras.

La ventana que permite consultar los datos (examinador), está conformada por varios objetos: una cuadrícula y 2 o más botones de comando. La clase *cuadrícula* nos servirá para presentar propiamente los datos de las respectivas tablas, mientras que cada uno de los botones de comando tendrán una función específica sobre la información presentada (modificar, crear, borrar...).

Para el desarrollo de los 2 módulos seleccionados (varios y dependencias) requerimos de un examinador para cada módulo. Para efecto de mostrar las facilidades que proporciona VFP en el desarrollo de aplicaciones, uno de los módulos será lanzado en una ventana de tipo modal y el otro en una ventana no modal, por lo que algunas propiedades entre las 2 ventanas son distintas para lograr el efecto requerido.

Recordemos que una ventana de tipo *no modal* o *libre* permite interactuar entre 2 o más ventanas.

Todas las clases de base de Visual FoxPro reconocen como mínimo las siguientes propiedades:

*Class* especifica el tipo de clase de que se trata.

*BaseClass* indica la clase de base de la que se deriva (Form, Custom, etc.)

*ClassLibrary* la biblioteca de clases en la que está almacenada.

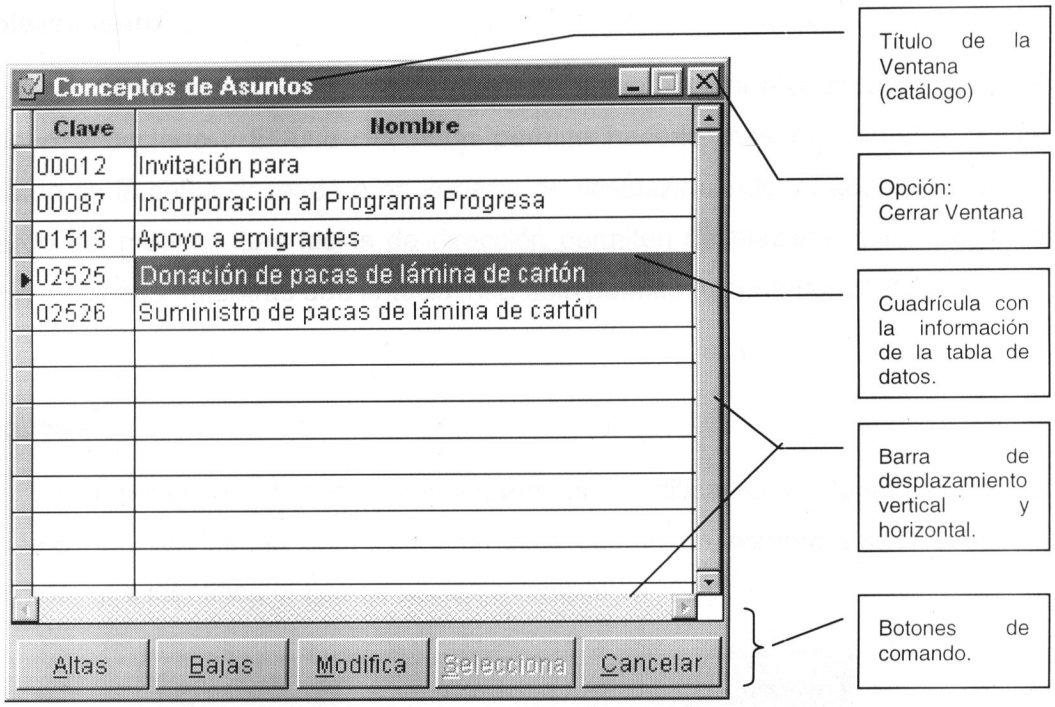
Y como mínimo los siguientes eventos:

*Init* ocurre cuando se crea el objeto.

*Destroy* ocurre antes de que el objeto se libere de la memoria.

*Error* ocurre siempre que tiene lugar un error en procedimientos de evento o de método de la clase.

La clase *frVarios* (examinador) contiene varios objetos: cuadrícula y varios botones de comandos que permiten la edición del contenido; para cada botón de comando se especifican las características, así como sus respectivos procedimientos (se escribe un trozo de código en el método correspondiente que le indica la función precisa a realizar, ya sea guardar, cancelar, borrar, modificar, etc.). Ver siguiente figura:




Esta pantalla de consulta permite modificar el contenido de la tabla *varios.sdb*, la cual contiene los catálogos de asuntos, estados, títulos, cargos, entre otros.

**Contenido de la ventana.-**

Podemos identificar básicamente 3 secciones:

En la parte superior de la pantalla hay una barra de título, la cual contiene una frase que identifica el tipo de información presentada (por ejemplo: Conceptos de Asuntos), también se puede percibir un botón en el extremo derecho  que permite *Cerrar la Ventana.*

*En la sección central, ocupando la mayor parte de la ventana,* se encuentra propiamente la información ordenada dentro de una cuadrícula (filas y columnas), así como una barra de desplazamiento vertical y otra horizontal.

La última sección está conformada por los diferentes botones de comandos, que permiten modificar la información de la tabla de datos.

**Desplazamiento.-**

Para desplazarse y ver todos los datos del examinador (cuadrícula), puede usar el mouse o el teclado. El uso del ratón permite hacerlo más rápidamente, basta con colocarse en la celda deseada o en la barra de desplazamiento y hacer clic con el botón izquierdo del mouse. Las teclas de dirección permiten desplazarse celda por celda, en todas direcciones, mientras que la tecla TAB solo permite hacerlo horizontalmente.

**Opciones.-**

Los botones de la parte inferior permiten modificar el contenido de las tablas. Cada uno de ellos tienen como característica común un nombre y una de sus letras subrayada, que identifica el tipo de acción a realizar.


Hay 2 formas de accionar alguno de los botones disponibles:

- Haciendo clic en la opción deseada, o bien,
- Pulsando la combinación de teclas Alt + letra_subrayada.

 Este icono, ubicado en la parte superior derecha de las ventanas, permite cerrar la ventana activa. También lo puede conseguir pulsando la tecla ESC en el teclado o haciendo clic en la opción Cancelar.


Esta opción permite agregar nuevos datos a la tabla *varios.sdb*. Siempre estará disponible en el examinador. Habilita una ventana en donde se capturan la información que se agregará en la tabla de datos.

Internamente, en el Click Event, se utiliza una copia de la tabla identificada con un nombre o alias. Recordemos que las tablas de la base de datos tienen un campo llamado "tipo", cuyo contenido permite identificar el catálogo al que pertenece la información.

Por ejemplo:

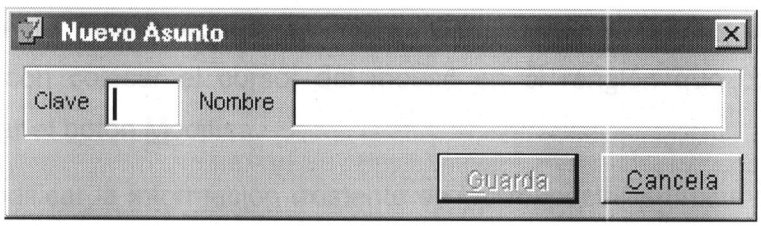
| *Tipo* | *Identifica* |
|---|---|
| "A" | Catálogo de Asuntos |
| "C" | Catálogo de Cargos |
| "D" | Tipo de Documento |
| "E" | Catálogo de Estados |
| "1" | Enlace/Contacto en la Coordinación |

Si la ventana activa es el examinador del catálogo Asuntos, sólo se presentarán los datos cuyo valor en el campo *tipo* sea "A".

Este acceso a cierto intervalo de registros se basa en las claves de índice de la tabla, es un filtro en el índice. Mediante la instrucción *SET KEY TO cTipo* es posible acceder únicamente a los registros con claves de índice coincidentes en su valor. La tabla *varios.sdb* está indexada por la llave: tipo+clave, lo que permite realizar el acceso a registros referentes a un catálogo específico (asuntos).

Al presionar el botón Altas, internamente se crea un objeto con características de pantalla de edición y se pasa como parámetro el tipo de catálogo seleccionado; de tal forma que al ingresar nuevos datos, el valor del campo *tipo* será el mismo que el del catálogo que lo invocó.

En esta segunda ventana se solicitan los datos de la tabla respectiva, como lo son la clave, nombre o descripción, texto, según sea el caso. Cuando esta ventana se activa, inhibe las demás ventanas y las opciones del menú.




Nótese que inicialmente el botón Guarda permanece inactivo mientras no sea capturado algún valor desde el teclado, además los recuadros de captura están en blanco

o vacíos. Mediante los métodos *actualiza, botones* y la propiedad *actualizado*, ambos definidos por el programador, se detecta alguna modificación en la pantalla de captura.

El formato de los datos se valida automáticamente desde el momento de capturarlos, gracias a la máscara de datos preestablecida internamente en cada objeto de la clase (por ejemplo: si el texto será en mayúsculas o minúsculas). Esto reduce tiempo de validación y evita errores desde la captura.

Una vez ingresados los datos en la pantalla de captura, la tabla de datos se actualizará al presionar el botón Guardar. Esto siempre y cuando no falte algún dato o la clave no esté duplicada con alguna de las existentes.

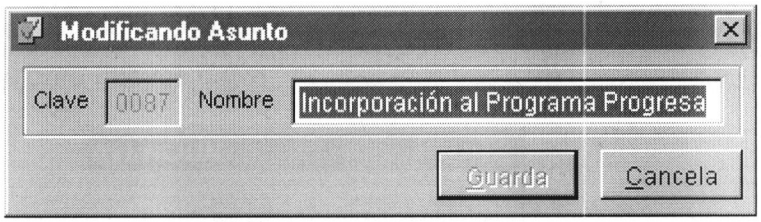
En el caso de faltar algún dato de los solicitados, el sistema mostrará un pequeño mensaje en la parte superior derecha de la pantalla, requiriendo capturar el dato faltante. Por otra parte, la nueva clave será comparada tanto en su *tipo* como la *clave* misma con las existentes en *varios.sdb*; de tal forma que si existiese una clave igual el sistema notificará de igual manera, que la clave ya existe y el cursor se colocará en el recuadro de la clave esperando que sea capturada una nueva.

Es importante señalar que al tratar de crear un nuevo registro (Altas), se utiliza una pantalla de edición, la cual contiene los campos ordenados de tal forma que sea fácil y práctica de utilizar. Además aquellos registros marcados para ser eliminados ("x") son reutilizados como nuevos registros y solo en caso de no existir alguno disponible, se agregan registros en blanco a la tabla activa para almacenar los nuevos datos.

Por otra parte, el botón Cancelar permite cerrar la ventana sin guardar los cambios. También puede pulsar la tecla ESC o hacer clic con el mouse en el icono *cerrar ventana*, localizado en la esq. superior izq. de la ventana.


Esta opción funciona similar a la de Altas, ya que sus ventanas de edición tienen la misma apariencia y contenido. Para modificar algún registro específico, basta con colocar el cursor del mouse en el renglón que contiene dicha información y pulsar el botón Modifica.

Permite modificar la información existente en las respectivas tablas de la base de datos, invocando de igual forma una pantalla de edición (ver sig. fig.).

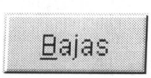



La diferencia radica en que esta ventana contiene la información del registro actual y la clave está protegida contra escritura como medida de seguridad. También contiene los métodos que detectan si realmente el usuario efectuó una modificación y si los datos están completos (*actualiza*).

Tanto para la opción Altas y Modifica se invoca a otros diseños de formas dependiendo del tipo de datos o catálogos a modificar:

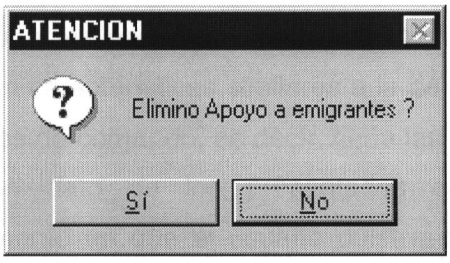
- frVariosDoc
- frVariosAsu
- frVariosCar
- frVariosEst
- frVariosTit
- frContacto

Aquí es donde vemos las facilidades brindadas por VFP al crear pantallas similares, pues permite crear clases de formulario con una apariencia o comportamiento personalizado para que sirvan como plantillas para todos los formularios que cree, de tal forma que frVariosDoc, frVariosAsu y las otras pantallas tienen apariencia y funcionamiento similar pues se basan en la misma clase *forma* de la biblioteca *misclase*.



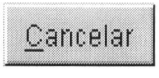

Para eliminar un registro de la tabla, es presentado un cuadro de diálogo en la parte central de la pantalla, en el que se pide al usuario confirmar que el elemento seleccionado se borrará.

Esta opción no elimina totalmente los datos del registro, es decir, físicamente pone una marca ("x") en el campo *tipo* como identificador de los registros no visibles para el usuario y reutilizables para nuevos datos.




El cuadro de diálogo anterior contiene 2 botones:

Al hacer clic sobre la opción *Sí*, el registro será eliminado del examinador. Por seguridad de los datos se establece como predeterminada la opción *No*. De lo contrario el usuario podría eliminar algún dato inconscientemente, es decir, pulsar <*ENTER*> y perder información.




Permite cerrar la ventana activa. Destruye el objeto mediante su método *Click Event*.

El Destroy Event permite cerrar el área de trabajo actual al pulsar la tecla ESC o al hacer clic ya sea en el icono *cerrar* de la parte superior de la ventana o en el botón de comando *Cancelar*.

El método Init es el primero en ejecutarse, ocurre cuando se crea un objeto, lo que hace es preparar tanto la apariencia como el contenido del *grid*. Permite abrir el alias de la tabla a usar, es recomendable no trabajar directamente sobre la tabla original, evita lentitud de operaciones y daños o pérdida de información.

KeyPress Event, es el método que ocurre cuando el usuario presiona y suelta una tecla. El objeto con el enfoque recibe el evento cuando el Form no contiene controles, o ninguno de sus controles está visible y activado.

Al pulsar con el ratón en el botón Altas, se desencadena el evento *Click*, que según el tipo de datos mostrados en *frVarios*, creará un objeto (pantalla de captura) y lo mostrará al usuario.

El evento *Click* del botón Bajas, activará el cuadro de diálogo donde se requiere al usuario confirmar si realmente desea eliminar ese dato.

La clase *frDependencias* permite la representación de los datos mediante una ventana o examinador, con características similares a la de *frVarios*. La diferencia radica en que no contiene botones de comando, es decir, tiene las mismas opciones de agregar, modificar o borrar algún dato de los presentados en el examinador, pero se desencadenan en el momento en que el usuario pulsa alguna tecla en específico, por ejemplo:

- Para realizar alguna modificación se debe presionar la *barra espaciadora*, así el sistema invocará a la pantalla de edición de la tabla Dependencias (*frDependenciasEdit*). Esta nueva ventana de edición contiene los diferentes campos de la tabla Dependencias, con la diferencia de que se utilizan otras clases como lo son *OptionGroup* para capturar el sexo del enlace, *ComboBox* para seleccionar algún dato de una lista de posibles valores, *TextBox* para ingresar nombres y descripciones.

  Además también contiene los botones para Guardar o Cancelar, cuyas funciones son las mismas que las de la ventana de edición de la tabla Varios.

- La tecla *Insert* permite agregar nuevos datos a la tabla Dependencias. En esta opción también se invoca a *frDependenciasEdit*.
- Pulsando la tecla *Supr* es invocado el cuadro de diálogo en donde se solicita al usuario que confirme si desea eliminar el dato seleccionado.

Con todo lo anterior comprobamos que Visual FoxPro ofrece muchas ventajas y facilidades para desarrollar sistemas al gusto del usuario, pero lo más importante es que se logra con poco esfuerzo y en caso de requerirse una modificación se puede conseguir fácil y rápido gracias a las características de la programación orientada a objetos.

# V Prueba y Mantenimiento del Sistema

Todos los programas de aplicación recientemente escritos o modificados, así como los nuevos manuales de procedimientos, nuevo hardware y todas las interfaces de sistema deben ser probadas extensamente para asegurar su calidad.

Sin embargo, las pruebas deben llevarse a cabo durante el desarrollo del sistema y no simplemente al final. Las pruebas se realizan en cada subsistema o módulo de programa conforme el trabajo avanza, esto disminuye la cantidad de errores o corrección de fallas al final.

Antes de que el sistema sea puesto en producción, todos los programas deben ser probados en el escritorio, revisados con datos de prueba y datos reales para ver si los módulos trabajan juntos entre ellos, tal como se planeó.

Los programadores, analistas, operadores y usuarios juegan papeles diferentes en los diversos aspectos de la prueba. Para los programadores y analistas lo importante es la facilidad para entender y modificar tanto el sistema mismo, como la base de datos y mantener la seguridad de los mismos.

Las reacciones de los operadores o capturistas al interactuar por vez primera con el sistema, permiten evaluar la facilidad y comodidad para ingresar datos haciendo uso de las pantallas de edición del sistema. También se puede detectar si realmente están validados los datos y si existe alguna confusión al ejecutar determinada opción.

Finalmente, los ejecutivos o jefes de área evaluarán si el sistema es realmente funcional, ya que la información que ofrezca el sistema les permitirá tomar decisiones sobre aspectos del negocio o empresa. Para ellos lo importante es que el sistema emita

la información de una manera fácil y clara, la mayoría de las ocasiones en forma de reportes.

Por lo anterior un sistema debe someterse a pruebas finales en *operatividad* y *funcionalidad*. Sin embargo, es recomendable revisar módulo por módulo para detectar a tiempo posibles adecuaciones al problema y llevar a cabo su solución.

Por llevar a cabo la realización de algunos módulos del sistema, sólo podemos hablar de una evaluación parcial.

La implantación se efectuará una vez terminados los módulos restantes. Se cuenta con el equipo necesario para su buen funcionamiento, pero se sugiere hacer pruebas con datos reales y funcionando simultáneamente los dos sistemas.

Desde el diseño del sistema global y el desarrollo de los módulos seleccionados, se está contemplando el funcionamiento en un entorno de red. Estas condiciones ya están dadas en la institución.

Así mismo, en lo que toca a las capacidades de cada uno de los elementos de la red todos ellos cubren más que suficiente los requisitos que impone la instalación de un sistema desarrollado en Visual Fox Pro.

# Conclusiones

Tal como se planeó en un principio, se efectuó un análisis general del sistema profundizando en aquellas áreas de mayor importancia o relevancia.

Al término de todo un proceso de investigación y análisis, surgieron múltiples opciones para efectuar mejoras al sistema vigente. Algunas de ellas parecían fáciles de desarrollar, otras eran costosas y a final de cuentas ninguna reflejaría un beneficio significativo.

Para llevar a cabo un buen análisis es necesario recopilar información, tanto de los testimonios registrados durante las entrevistas aplicadas a las personas que trabajan con el sistema, como de la documentación existente en la empresa.

Sin embargo, considero que es necesario e indispensable observar y conocer el funcionamiento del sistema, incluyendo la forma manual de trabajo. Lo anterior contempla los procedimientos más básicos y sencillos hasta los más complicados y sistematizados, de igual manera se deben identificar las fallas o problemas que se presentan en cada uno de ellos.

Se debe tener especial cuidado en las entrevistas, estas deben contener preguntas que nos permitan identificar las características del sistema y sus deficiencias, así como las sugerencias y opiniones que puedan surgir de los entrevistados. Es importante observar el comportamiento de los entrevistados, ya que en ocasiones las respuestas no son muy objetivas.

Muchas veces se interpreta de manera inadecuada las respuestas de los entrevistados, llevándonos a supuestas necesidades o falsos requerimientos, que finalmente al desarrollar la solución habremos perdido tiempo y esfuerzo.

Durante el desarrollo de este trabajo se presentaron algunos obstáculos o factores que retrasaron su terminación, tales como el tiempo, la disposición de las personas y la mía propia, el desconocimiento de algunos conceptos, entre otros.

Otro factor importante a considerar es la aceptación, del personal y del mismo jefe, a la posible actualización o incluso aplicación de una tecnología diferente. Por otra parte, el aplicar una tecnología o modelo nuevo implica estudiar nuevos conceptos, adquirir conocimientos y realizar múltiples pruebas.

Hemos comprobado que en el mercado y en especial en el mundo de la computación, hay muchas herramientas que cada vez son más específicas para problemas reales. Un buen desarrollo depende de un previo diseño de soluciones, además se debe seleccionar el software adecuado que nos facilite el trabajo. Se debe buscar siempre el máximo beneficio.

Para que el sistema sea vigente y consistente, se consideró la continua evolución y aparición de nuevas herramientas y técnicas en el desarrollo de sistemas, tratando de integrar al máximo estos elementos. En ocasiones los sistemas usan la tecnología de punta, pero la estructuración y diseño interno es obsoleto e inadecuado.

El basarse en un modelo de programación anterior es uno de los errores en que los programadores caemos con facilidad. Por lo general nos acostumbramos o encerramos en un estilo que nos ha dado buenos resultados y que aplicamos con más frecuencia, lo que nos dificulta el aceptar un modelo nuevo. El seguir los patrones del diseño anterior, como lo son las estructuras de base de datos y procedimientos, nos conducirán a desarrollar un sistema que quizá tecnológicamente sea vanguardista, pero funcionalmente tenga las mismas deficiencias y errores que el anterior.

El aplicar un modelo de programación orientado a objetos, me ayudó a comprender mejor los conceptos de herencia y encapsulación; además, que la programación modular realmente tiene más ventajas que la programación tradicional estructurada.

Quizás para un analista o programador que desconozca el sistema, sea un poco difícil tratar de armar las partes del sistema. Para ello es indiscutiblemente necesario documentar cada uno de los programas, así como llevar un registro impreso de su estructura, diseño, funcionamiento, especificaciones, posibles errores y su corrección.

En la programación modular es más fácil y rápido identificar y corregir los errores conforme van surgiendo (en cada módulo), evitando tener al final una lista enorme de problemas desconocidos.

Por otra parte, al aplicar ésta nueva tecnología de programación visual orientada a objetos (*Visual FoxPro*), hemos comprobado los beneficios y ventajas que ofrece a los usuarios del sistema, y más aún a los programadores.

El trabajar en un ambiente gráfico es mucho más agradable y fácil para los usuarios, esto se pudo constatar al realizar las pruebas parciales del avance que se tiene. Para los programadores el diseñar aplicaciones gráficas requería más esfuerzo y tiempo. Este nuevo lenguaje proporciona facilidades, pues en sus bibliotecas vienen prediseñadas algunas clases con apariencia de botones, flechas de avance, cuadrículas, entre otras; con esto los programadores sólo se deben preocupar por las funciones que realizarán cada uno de los objetos.

Además, los nuevos diseños se pueden probar inmediatamente sin necesidad de esperar a terminar todo el proyecto, esto disminuye los posibles cambios al final y la completa satisfacción del cliente.

Por otra parte, para lograr que el sistema propuesto tenga éxito es necesario que se sigan tomando en cuenta las características propias de la programación orientada a objetos, como son la modularidad, encapsulamiento, herencia, etc.

Considero que se cumplieron los objetivos y metas planteadas al inicio de este proyecto, cuyos resultados se fueron comprobando conforme avanzaba el desarrollo del mismo, sin necesidad de esperar al final.

# APENDICE

# Apéndice 1
# Base de Datos Anterior

En esta sección se muestra la estructura y el contenido de cada una de las tablas originales que conforman el sistema actual.

En la parte superior se encuentra el nombre de la tabla. El campo sombreado representa la clave primaria de cada tabla.

La tabla *sac_ar01.dbf* es la principal, en ella están registrados datos importantes como son las peticiones, el nombre y datos del solicitante, entre otros. Está relacionada con otras tablas, es decir, solicita datos para completar la información de cada registro.

Por otra parte *sac_ar02.dbf* representa el catálogo de dependencias a las que se turnan las peticiones, contiene la clave de la dependencia, así como el titular, domicilio, etc.

En *sac_ar03.dbf* se pueden identificar los municipios y el distrito local al que pertenece.

En la tabla *sac_aro4.dbf* son registrados los tipos de documentos, es decir, el origen de las solicitudes (gira, verbal, diputados, gestores, entre otros).

*Sac_ar05.dbf* contiene la lista de Estados de la República Mexicana y su respectiva clave.

El catálogo de localidades pertenecientes a cada municipio lo encontramos en *sac_ar06.dbf.* Los municipios también están registrados en *sac_ar11.dbf.*

La tabla *sac_ar07.dbf* contiene el catálogo de asuntos o peticiones.

Para el control de acceso a cada opción de los menús el sistema recurre a *sac_ar10.dbf*, donde se registra a los usuarios autorizados y su respectiva clave de acceso. Esta tabla sólo puede ser modificada por el administrador del sistema.

En *sac_ar12.dbf* se encuentra el catálogo de cargos de los peticionarios y la lista de títulos está registrada en *sac_ar13.dbf.*

Finalmente, los beneficiarios se pueden consultar en la tabla *sac_ar14.dbf.*

**1** Tabla: SAC_AR01.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Destino1 | Carácter | 4 | Esta clave de 4 dígitos representa la dependencia a la cual fue turnada la petición, según catálogo de dependencias contenido en la tabla SAC_AR02.DBF |
| Folio | Carácter | 5 | Representa el número de control consecutivo para cada petición. Es el campo llave de la tabla |
| Cons_Folio | Carácter | 2 | Representa más de una petición de un mismo peticionario. Actualmente el valor único predeterminado es "00" |
| Ano | Carácter | 2 | El valor predeterminado es "99" para este año |
| Sexo | Carácter | 1 | Representa el sexo del solicitante. Actualmente sin uso |
| Programa | Carácter | 20 | Contiene el nombre del Programa de Radio ó T.V., donde se registraron las peticiones |
| Medio | Carácter | 15 | Representa el medio de comunicación donde se registraron las peticiones |
| Referencia | Carácter | 30 | En este campo se registra el nombre de la persona, coordinación, distrito ó evento especial que gestionó la petición |
| Fecha_Cap | Fecha | 8 | Representa la fecha en que fue codificada la petición |
| Nom_Sol | Carácter | 75 | Contiene el nombre completo del solicitante, empezando por el nombre y después los apellidos |
| Sexo_Sol | Carácter | 1 | Representa el sexo del solicitante ("F" ó "M" ó "X" ó "Y") |
| Nom_Benefi | Carácter | 75 | Contiene el nombre completo del beneficiario |
| A_Pat | Carácter | 15 | Contiene el apellido paterno del beneficiario. Actualmente sin uso |
| A_Mat | Carácter | 15 | Contiene el apellido materno del beneficiario. Actualmente sin uso |
| Sexo_Benef | Carácter | 1 | Representa el sexo del beneficiado ("F" ó "M" ó "X" ó "Y") |
| Firmantes | Carácter | 3 | Representa el número de firmantes de la petición |
| Dir_Sol | Carácter | 70 | Contiene el domicilio completo del solicitante |
| Pais | Carácter | 15 | Representa el país donde radica el peticionario. El valor predeterminado es: "MEXICO" |
| Estado | Carácter | 2 | Representa el estado donde radica el peticionario, según catálogo de estados contenido en la tabla SAC_AR05.DBF. El valor predeterminado es: "16" (MICHOACAN) |
| Cve_Loc | Carácter | 4 | Representa la comunidad donde radica el peticionario, según catálogo de comunidades contenido en la tabla SAC_AR06.DBF |
| Cve_Muni | Carácter | 3 | Representa la clave del municipio del peticionario, según catálogo de municipios contenido en la tabla SAC_AR06.DBF |
| Mun_Gira | Carácter | 3 | Representa el lugar de la gira, es decir, el municipio visitado por el Ejecutivo del Estado |
| Asun_Corto | Carácter | 4 | Representa el asunto corto ó rubro que se solicita en la petición, según catálogo contenido en la tabla SAC_AR07.DBF |
| Cantidad | Carácter | 4 | Especifica la cantidad solicitada de determinado rubro |
| Tex_Asunto | Carácter | 250 | Contiene texto complementario a la descripción del Asun_Corto |
| Texto_Adi | Carácter | 200 | Contiene texto complementario a la descripción del Asun_Corto, pero sin incluir el nombre, título, ni cargo del solicitante |
| Fech_Env1 | Fecha | 8 | Representa la fecha de captura de la petición |
| Respuesta | Carácter | 25 | Contiene la respuesta de la petición (NO PROCEDENTE ó POSITIVA ó EN PROCESO A CORTO/LARGO PLAZO ó EN TRAMITE). |
| Tipo_Petic | Carácter | 2 | Representa el tipo de petición según clasificación contenida en la tabla SAC_AR04.DBF |
| F_Cap | Fecha | 8 | Representa la fecha del sistema |

**1** Tabla: SAC_AR01.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Q_Cap | Carácter | 6 | Contiene las iniciales del usuario que capturó la petición, según tabla de usuarios SAC_AR10.DBF |
| Q_Modresp | Carácter | 6 | Contiene las iniciales del usuario que modificó la respuesta de la petición, según tabla de usuarios SAC_AR10.DBF |
| Q_Modif | Carácter | 6 | Contiene las iniciales del usuario que realizó alguna modificación a la petición, según tabla de usuarios SAC_AR10.DBF |
| Fech_Modif | Fecha | 8 | Representa la fecha en que se llevó a cabo la modificación de la petición |
| F_Entrega | Fecha | 8 | Representa la fecha en que se entrega la petición a la dependencia donde fue turnada |
| F_Venci | Fecha | 8 | Representa la fecha en que de vence el tiempo de respuesta por parte de la dependencia. Aprox. Son 3 para peticiones de carácter especial y 8 días para peticiones ordinarias |
| Recordat | Numérico | 1 | Indica el número de veces que se ha solicitado la respuesta a la dependencia. Se incrementa cada vez que se imprime el reporte de oficios vencidos |
| Fech_Imp | Fecha | 8 | Representa la fecha en fueron impresos los oficios que se envían a la dependencia y al peticionario |
| Observa | Memo | 10 | Contiene el texto complementario a la respuesta de la petición |
| Gira | Carácter | 3 | Representa el número de gira, esto para uso interno y reportes |
| Cve_Titu | Carácter | 2 | Contiene la clave del título del peticionario, según catálogo contenido en la tabla SAC_AR13.DBF |
| Cve_Cargo | Carácter | 3 | Contiene la clave del cargo del peticionario, según catálogo contenido en la tabla SAC_AR12.DBF |
| Ano_Gira | Carácter | 2 | Representan los últimos 2 dígitos del año de la gira |
| Origen | Carácter | 13 | Es usado como control o bandera en reportes internos |
| | | **955** | |

**2** Tabla: SAC_AR02.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Cve_Depe | Carácter | 4 | Contiene la clave que representa a cada una de las dependencias. Es el campo llave de la tabla |
| Nombre | Carácter | 40 | Contiene el nombre de los titulares de las dependencias |
| Cargo | Carácter | 60 | Contiene el cargo de los titulares de las dependencias |
| Domicilio | Carácter | 60 | Contiene el domicilio completo donde se ubican las oficinas de las dependencias |
| Descrip | Carácter | 100 | Contiene la descripción o nombre de las dependencias |
| Observacio | Carácter | 250 | Contiene las anotaciones hechas por los enlaces, considerando la situación de respuesta por parte de las dependencias |
| Enlace | Carácter | 40 | Representa el nombre de la persona-contacto en cada una de las dependencias |
| Telefonos | Carácter | 20 | Contiene los teléfonos de las dependencias |
| Nombra | Carácter | 40 | Representa el nombramiento del enlace, dentro de las dependencias |
| Sexo | Carácter | 1 | Identifica el sexo del enlace de las dependencias |
| En_Coord | Carácter | 30 | Contiene el nombre del enlace dentro de la misma Coordinación |
| | | **646** | |

**i5** Tabla: SAC_AR03.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Mpio | Carácter | 3 | Representa la clave de cada uno de los municipios |
| Clave | Carácter | 3 | Representa la clave del distrito local al que pertenece el municipio |
| Municipio | Carácter | 60 | Contiene la descripción o nombre del municipio |
| | | **67** | |

**3** Tabla: SAC_AR04.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Tipo_Peti | Carácter | 2 | Contiene la clave que representa a cada uno de los tipo de peticiones. Es el campo llave de la tabla |
| Des_Peti | Carácter | 30 | Contiene la descripción de los tipos de peticiones |
| Texto1 | Memo | 10 | Contiene el primer párrafo que forma parte del oficio que se envía a la dependencia |
| Texto2 | Memo | 10 | Contiene el segundo párrafo que forma parte del oficio que se envía a la dependencia |
| Texto3 | Memo | 10 | Contiene el párrafo final que forma parte del oficio que se envía a la dependencia |
| | | **63** | |

**4** Tabla: SAC_AR05.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Cve_Edo | Carácter | 2 | Contiene la clave que identifica a cada uno de los estados de la República Mexicana. Es el campo llave de la tabla |
| Descrip | Carácter | 22 | Contiene la descripción o nombre de cada uno de los diferentes estados de la República Mexicana |
| | | **25** | |

**5** Tabla: SAC_AR06.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Mpio | Carácter | 3 | Representa la clave que identifica a cada uno de los diferentes municipios del Estado. La llave está compuesta por este campo + el campo "loc" |
| Loc | Carácter | 4 | Representa la clave que identifica a cada una de las diferentes comunidades |
| Descrip | Carácter | 30 | Contiene la descripción del municipio o localidades |
| Titular | Carácter | 30 | Actualmente sin uso |
| Direc | Carácter | 60 | Actualmente sin uso |
| Cod_Pos | Carácter | 5 | Contiene el código postal de cada municipio |
| | | **133** | |

**6** Tabla: SAC_AR07.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Cve_Asun | Carácter | 4 | Representa la clave que identifica a los diferentes rubros o peticiones. Es el campo llave de la tabla |
| Des_Asun | Carácter | 100 | Contiene la descripción de los rubros o peticiones |
| | | **105** | |

**7** Tabla: SAC_AR10.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Usuario | Carácter | 6 | Contiene la clave del usuario para accesar al sistema |
| Nombre | Carácter | 35 | Contiene el nombre completo del usuario |
| Iniciales | Carácter | 8 | Contiene las iniciales que identifican al usuario en el sistema |
| Opc_1_1 | Lógico | 1 | Contiene un valor de Falso (F) ó Verdadero (T), indicando si tiene ó no acceso a determinada opción del sistema |
| Opc_1_2 | Lógico | 1 | " " |
| Opc_1_3 | Lógico | 1 | " " |
| Opc_1_4 | Lógico | 1 | " " |
| Opc_1_5 | Lógico | 1 | " " |
| Opc_1_6 | Lógico | 1 | " " |
| Opc_1_7 | Lógico | 1 | " " |
| Opc_2_1 | Lógico | 1 | " " |
| Opc_2_2 | Lógico | 1 | " " |
| Opc_2_3 | Lógico | 1 | " " |
| Opc_2_4 | Lógico | 1 | " " |
| Opc_3_1 | Lógico | 1 | " " |
| Opc_3_2 | Lógico | 1 | " " |
| Opc_3_3 | Lógico | 1 | " " |
| Opc_3_4 | Lógico | 1 | " " |
| Opc_3_5 | Lógico | 1 | " " |
| Opc_3_6 | Lógico | 1 | " " |
| Opc_3_7 | Lógico | 1 | " " |
| Opc_3_8 | Lógico | 1 | " " |
| Opc_3_9 | Lógico | 1 | " " |
| Opc_4_1 | Lógico | 1 | " " |
| Opc_4_2 | Lógico | 1 | " " |
| Opc_4_3 | Lógico | 1 | " " |
| Opc_4_4 | Lógico | 1 | " " |
| Opc_4_5 | Lógico | 1 | " " |
| Opc_4_6 | Lógico | 1 | " " |
| Opc_5_1 | Lógico | 1 | " " |
| | | **77** | |

**i5** Tabla: SAC_AR11.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Mpio | Carácter | 3 | Contiene la clave que identifica a cada uno de los municipios. Es el campo llave de la tabla |
| Loc | Carácter | 4 | Actualmente sin uso |
| Descrip | Carácter | 30 | Contiene la descripción o nombre de los municipios |
| Titular | Carácter | 30 | Actualmente sin uso |
| Direc | Carácter | 60 | Actualmente sin uso |
| Cod_Pos | Carácter | 5 | Contiene el código postal de cada municipio |
| | | **133** | |

**8** Tabla: SAC_AR12.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Cargo | Carácter | 3 | Representa la clave que identifica a cada uno de los cargos del peticionario en el catálogo. Es el campo llave de la tabla |
| Descrip | Carácter | 40 | Contiene la descripción de cada cargo |
| | | **44** | |

**9** Tabla: SAC_AR13.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Titulo | Carácter | 2 | Representa la clave que identifica a cada uno de los títulos del peticionario en el catálogo. Es el campo llave de la tabla |
| Descrip | Carácter | 40 | Contiene la descripción de cada título |
| | | **43** | |

**10** Tabla: SAC_AR14.DBF

| *Campo* | *Tipo* | *Ancho* | *Contenido* |
|---|---|---|---|
| Oficio | Carácter | 5 | Representa el folio de la petición |
| Nombre | Carácter | 45 | Contiene el nombre de la persona beneficiada |
| | | **51** | |

# Apéndice 2
# Base de Datos Propuesta

Tabla: Peticiones

| *Campo* | *Tipo* | *Ancho* | *Descripción* |
|---|---|---|---|
| TIPO | Carácter | 1 | Identificador del tipo de registro. |
| OFICIO | Carácter | 5 | Oficio. Número consecutivo en función de su codificación |
| DEP | Carácter | 4 | Dependencia a la que se turnará la petición |
| EJERCICIO | Carácter | 4 | Ejercicio del registro del formato |
| REFERENCIA | Carácter | 30 | Referencia relacionada con la persona que gestiona |
| DOCUMENTO | Carácter | 2 | Tipo de Documento |
| PETICION | Carácter | 4 | Petición |
| CANTIDAD | Numérico | 4 | Cantidad de lo solicitado |
| PETICIONTX | Memo | | Texto Corto |
| PAIS | Carácter | 15 | País |
| ESTADO | Carácter | 2 | Estado |
| GIRA | Carácter | 3 | Número de Gira |
| MUNICIPIO | Carácter | 3 | Municipio |
| LOCALIDAD | Carácter | 4 | Localidad |
| TITULO | Carácter | 2 | Título |
| NOMBRESOL | Carácter | 75 | Nombre del Solicitante |
| CARGO | Carácter | 3 | Cargo |
| ASOCIACION | Carácter | 50 | Organización a la que pertenece |
| SEXO | Carácter | 1 | Sexo del Solicitante |
| NOMBREBEN | Carácter | 75 | Nombre del Beneficiario |
| FIRMANTES | Carácter | 3 | Firmantes |
| DOMICILIO | Carácter | 70 | Domicilio del Solicitante |
| LUGARGIRA | Carácter | 3 | Lugar de la Gira |
| FECHACAP | Fecha | | Fecha de Captura |
| CAPTURISTA | Carácter | 6 | Iniciales de la persona que capturó la petición |
| MODRES | Carácter | 6 | Iniciales de la persona que modificó la respuesta |
| MODPET | Carácter | 6 | Iniciales de la persona que realizó modificacionesa la petición |
| STATUSPET | Carácter | 25 | Status que guarda la petición (positiva, negativa, etc.) |
| RESPUESTA | Memo | | Respuesta emitida por la dependencia |
| MEDIO | Carácter | 15 | Programa de Radio o T.V. |
| MODFECHA | Fecha | | Fecha de última modificación |
| ENTREGA | Fecha | | Fecha de entrega para dependencias |
| VENCE | Fecha | | Fecha límite de vencimiento |
| INTENTO | Numérico | 1 | Número de recordatorios de solicitud de respuesta a la dependencia |
| IMPRESO | Fecha | | Fecha de impresión de oficios |
| STATUSREG | Carácter | 1 | Status que guarda el registro de la petición |

Tabla: Dependencias

| *Campo* | *Tipo* | *Ancho* | *Descripción* |
|---|---|---|---|
| TIPO | Carácter | 1 | Identificador del tipo de registro. |
| CLAVE | Carácter | 4 | Contiene la clave que representa a cada una de las dependencias. Es el campo llave de la tabla. |
| NOMBRE | Carácter | 100 | Contiene la descripción o nombre de las dependencias. |
| DOMICILIO | Carácter | 60 | Contiene el domicilio completo donde se ubican las oficinas de las diferentes dependencias. |
| TELEFONOS | Carácter | 20 | Contiene los teléfonos de las dependencias. |
| TITULAR | Carácter | 40 | Contiene el nombre del titular de la dependencia. |
| TITCARGO | Carácter | 60 | Contiene el cargo del titular de la dependencia. |
| ENLACE | Carácter | 40 | Representa el nombre de la persona-contacto en cada una de las dependencias. |
| ENLCARGO | Carácter | 40 | Representa el nombramiento del enlace, dentro de las dependencias. |
| SEXO | Carácter | 1 | Identifica el sexo del enlace de las dependencias. |
| CONTACTO | Carácter | 30 | Contiene el nombre del enlace dentro de la misma Coordinación. |
| NOTAS | Carácter | 250 | Contiene las anotaciones hechas por los enlaces, considerando la situación de respuesta por parte de las dependencias. |

Tabla: Municipios

| *Campo* | *Tipo* | *Ancho* | *Descripción* |
|---|---|---|---|
| TIPO | Carácter | 1 | Identifica si el registro en la tabla se trata de distrito, municipio o localidad. |
| CLAVE | Carácter | 3 | Representa la clave del municipio. |
| LOCDIS | Carácter | 4 | Contiene la clave de la localidad o distrito para el municipio "X". |
| NOMBRE | Carácter | 30 | Contiene la descripción del distrito, municipio o localidad. |
| CODPOS | Carácter | 5 | Contiene el Código Postal de cada municipio. |

Tabla: Varios

| *Campo* | *Tipo* | *Ancho* | *Descripción* |
|---|---|---|---|
| TIPO | Carácter | 1 | Identifica el tipo de registro, el tipo de catálogo a consultar (títulos, cargos, tipo de documento, beneficiarios, asuntos, estados) |
| CLAVE | Carácter | 5 | Representa la clave que identifica cada dato contenido en los diferentes catálogos |
| NOMBRE | Carácter | 100 | Contiene la descripción de cada una de las claves de los diferentes catálogos |
| TEXTO | Memo | | Contiene texto que forma parte de los oficios impresos |

Tabla: Accesos

| *Campo* | *Tipo* | *Ancho* | *Descripción* |
|---|---|---|---|
| TIPO | Carácter | 1 | Identificador del tipo de registro. |
| USUARIO | Carácter | 6 | Representa la clave que identifica cada usuario. |
| NOMBRE | Carácter | 35 | Contiene el nombre del usuario. |
| INICIALES | Carácter | 6 | Contiene las iniciales del usuario. |
| OPC_1_1 | Lógico | 1 | Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene o no acceso a ésta opción del menú |
| OPC_1_2 | Lógico | 1 | Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene o no acceso a ésta opción del menú |
| OPC_2_2 | Lógico | 1 | Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene o no acceso a ésta opción del menú |
| OPC_2_3 | Lógico | 1 | Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene o no acceso a ésta opción del menú |
| OPC_2_4 | Lógico | 1 | Contiene un valor de .F. (falso) o .T. (verdadero), indicando si tiene o no acceso a ésta opción del menú |
| * | Lógico | 1 | * |

# Bibliografía

"Sistemas de Información para la Administración"
James A. Senn
Grupo Editorial Iberoamérica

"Análisis y Diseño de Sistemas"
Kendall & Kendall
Ed. Prentice Hall 3ª. Edición

"Visual FoxPro 3.0 y 5.0"
Manual de Programación
Les Pinter / John Pinter
Ed. Mc. Graw-Hill

"Visual FoxPro 5"
Fundamentos y Técnicas de Programación
Rubén Iglesias Balbás
Computec

"El MétodoCientífico"
Arturo Rosenblueth
Ed. LaPrensa Médica Mexicana
Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional de México

"Análisis y Diseño de Sistemas de Información"
James A. Senn
Ed. Mc. Graw-Hill

"Ingeniería de Software"
Roger S. Pressman
Ed.Mc.Graw-Hill

"Fundamentos de Bases de Datos"
Henry F. Korth y Abraham Silberschatz
Mc. Graw-Hill 2ª. Edición