CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL
Departamento de Ingeniería Eléctrica
Sección de Computación

CONSTRUCCIÓN DE UNA BASE DE DATOS PICTOGRÁFICOS CON APLICACIÓN A LA COLECCIÓN DE MICROORGANISMOS CDBB-500 DEL CINVESTAV-IPN

TESIS QUE PRESENTA
JESÚS MANUEL OLIVARES CEJA
PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE INGENIERÍA ELÉCTRICA

DIRECTOR DE LA TESIS:
DR. SERGIO VÍCTOR CHAPA VERGARA

México, D.F., septiembre de 1996

Me acerque al hombre sabio y le pregunté.
- ¿Cómo adquirió su sabiduría?
El me contestó.
- Preguntando.

Hace unos quinientos años existía en un valle un pais de borregos, en las montañas un país de águilas. Las águilas cruzaban el valle volando libremente mientras que los borregos se la pasaban pastando. Cerca de ahí estaba el país de los lobos que se alimentaban con los borregos y sentian envídia de las águilas por lo que decidieron atacarlas. Así un día los lobos invadieron el país de las águilas y mataron a todas las que pudieron. A las sobrevivientes les decian que no eran águilas sino borregos y que no debían volar, que vieran como los borregos andaban siempre en el suelo, pastando.

Actualmente en el país de las águilas la mayoría andan arrastrandose por el suelo y algunas han perdido la habilidad para volar. Cuentan que afortunadamente, de vez en cuando, pasa un águila volando y les grita a sus compañeras que recuerden que son águilas y no borregos, que pueden volar y son libres.

RESUMEN

Se presenta un modelo de base de datos para la consulta y registro de información acerca de la colección de microorganismos concentrados en CDBB-500. La idea principal es visualizar las cepas con sus propiedades y características de tal forma que en la pantalla se puedan distinguir: colores, formas, tamaños, etc. además de sus parámetros. La conceptualización de la base de datos fue mediante el enfoque relacional y posteriormente se replanteo con base en el paradigma orientado a objetos. De este último se implantaron clases tipo con métodos de actualización y desplegado de datos pictográficos.

ABSTRACT

This document presents a database model to query and record information of the CDBB-500 microorganisms culture collection. The main idea is to visualize the strains with their properties and characteristics on such a way that on the screen we can distinguish: colors, shapes, sizes, etc. besides their parameters. The conceptualization of the database was using the relational approach, after that it was reestructured based on the object oriented paradigm. Of this last, we implemented tipical classes with updating methods and pictographical data displaying.

ÍNDICE

INTRODUCCIÓN

El propósito de este trabajo es obtener el modelo de la base de datos para la automatización de la colección de microorganismos CDBB-500, usando una metodología cuyas etapas más importantes son: a) recopilación de información y detección de requerimientos, b) análisis de la información propuesta de un modelo de la base de datos, c) depuración del modelo, d) implantación de clases tipo.

El primer modelo de base de datos utilizado fue el relacional por un requerimiento de la Comisión Nacional para la Conservación de la Biodiversidad (CONABIO), mismo que se replanteó de acuerdo al paradigma orientado a objetos por la ventaja de poder almacenar en la misma base tanto datos nominales como pictográficos.

Este trabajo está dirigido a especialistas del área biológica, a quienes les puede brindar una idea de las implicaciones en la automatización de su información. También le es útil a los estudiosos de computación e informática como un caso práctico en donde es factible aplicar muchas de las técnicas de esas áreas. El documento es un elemento de consulta para los participantes actuales y futuros del proyecto de automatización de la colección CDBB-500 en cuanto a como se construyo el modelo de la base de datos pictográficos.

En el capítulo I de este documento se mencionan los antecedentes y las etapas de la metodología para obtener el modelo de la base de datos.

En el capítulo II se desglosan los resultados obtenidos de la recopilación de información, detección de requerimientos y la vista global de la base de datos.

El capítulo III detalla cada una de las vistas de la base de datos, en la cual se indican además el diseño de las pantallas propuestas para la interacción con el usuario.

El capítulo IV ilustra la propuesta del sistema de información basado en el paradigma orientado a objetos. Se complementa con las clases implantadas en Visual Works$^{(TM)}$ para actualización de datos nominales y despliegue de datos pictográficos.

El capítulo V explica la mayoría de conceptos encontrados en el enfoque orientado a objetos de las bases de datos.

En la última parte se indican las conclusiones y las referencias bibliográficas utilizadas en el desarrollo de este trabajo.

Los datos presentados en este documento referentes al área de aplicación son por cortesía de la colección de microorganismos del Departamento de Biotecnología y Bioingeniería del CINVESTAV-IPN (conocida mundialmente con el acrómino CDBB-500).

I ANTECEDENTES

Desde 1990 en el CINVESTAV se iniciaron los trabajos conjuntos entre la sección de computación y la colección de microorganismos CDBB-500, ambos del mismo centro. Para 1992 se logró obtener la primera versión de una base de datos que proporcionó algunos resultados.

En 1994 y como parte de este trabajo de tesis, se retomó el caso, detectando que se carecía de un modelo apropiado en cuanto a la base de datos y se requería integrar información pictográfica a la misma. El propósito fue estructurar un modelo que incluyera imágnes mediante la aplicación de una metodología consistente de las etapas siguientes:

a) Recopilación de datos y detección de requerimientos
b) Análisis de datos y diseño conceptual de la base de datos
c) Depuración del modelo
d) Implantación del modelo de datos

Los datos recabados fueron referentes a conocer que es una colección de microorganismos, sus usuarios, entidades con quienes tiene vínculos, las actividades realiza y sus requerimientos de información tanto actuales como futuros.

Con los datos recabados se procedió a su análisis para proponer un modelo conceptual de la base de datos, mismo que fue sometido a su revisión por varios compañeros de la sección de Computación del CINVESTAV-IPN y cuyas aportaciones se incluyeron en el mismo.

En una etapa posterior el modelo de la base de datos se reestructuró para expresarlo mediante el paradigma orientado a objetos, aprovechando que un objeto se puede definir como una tabla que representa una relación.

La implantación de la base de datos se realizó en dos partes, la primera fue en el modelo relacional y l asegunda consistió en definir objetos tipo para actualización de datos nominales y despliegue de datos pictográficos.

II OBTENCIÓN DEL MODELO CONCEPTUAL Y LÓGICO DE LA BASE DE DATOS

En este capítulo se presentan los datos recopilados del área en que se desarrolla el modelo de la base de datos, las actividades que caracterizan a una colección de microorganismos, sus requerimientos de información y el modelo de la vista global de la base de datos propuesta.

II.1 Recopilación de datos

Una de las primeras preguntas a responder fue ¿qué es una colección de microorganismos? Ahora sabemos que es un centro que preserva, custodia, estudia, investiga un conjunto de cepas (microorganismos en un determinado medio de cultivo) de las cuales garantiza su pureza, viabilidad y confidencialidad. Además brinda un conjunto de servicios relacionados con su acervo [9].

Para brindar los servicios y garantizar la pureza y viabilidad de las cepas, el personal encargado de la colección está en un proceso de actualización continua, recabando datos de distintas fuentes, por ejemplo: usuarios depositantes, otras colecciones, ceparios, centros de investigación, documentos bibliográficos, asistencia a foros, resultados de estudios e investigaciones que ellos mismos llevan a cabo, etc. Cada persona dentro de la misma tiene un papel especifico, por lo que hay al menos un especialista para cada grupo taxonómico existente en la colección.

La colección CDBB-500, se inició en 1974, desde entonces se planteó como de importancia industrial y para la investigación tanto a nivel nacional como internacional. En 1977 se registró en el centro mundial de datos microbianos (WDC) con el acrónimo CDBB-500 como actualmente se conoce en el mundo. En 1981 se afilió a la Federación Mundial de Colecciones Microbianas (World Federation Culture Collections, WFCC). La edición de su primer catálogo se llevó a cabo en 1981. Luego de un acrecentamiento en su acervo microbiano y diversificación de sus servicios a nivel internacional, en 1992 la WFCC la postuló para formar parte del comité de educación internacional sobre colecciones de microorganismos.

Cada una de las cepas que constituye la colección representa un proceso de interés biotecnológico. A pesar de la gran cantidad de microorganismos que se conocen en el mundo entero, que alcanza varios miles de especies, aun existen otros tantos por conocer e investigar, para aprovechar los múltiples beneficios que pueden brindarle a la especie humana y a los seres vivos sobre la tierra.

A los usuarios nacionales e internacionales que se les brinda servicios son:
a) Industrias,
b) Otras Colecciones,
c) Ceparios,
d) Centros de investigación,
e) Instituciones académicas,
f) Personas fisicas,

A los proveedores nacionales e internacionales a quienes colección solicita servicios son:
a) Otras colecciones,
b) Ceparios,
c) Centros de investigación,

II.2 Actividades de una colección de microorganismos

El carácter de una colección de microorganismos está dado tanto por el acervo microbiano como por el conjunto de servicios que brinda a sus usuarios.

En esta sección se resumen las actividades que caracterizan a una colección microbiana [9,10]:

a) Distribución de cepas (figura II.1).- se consulta el catálogo de la colección para verificar si se tiene la cepa solicitada, en caso afirmativo se puede otorgar al usuario. Si no, se consulta el catálogo mundial anotándole al usuario los requisitos a cubrir, al tener lista la cepa, se le otorga. Si la cepa en ningún caso se tiene se le puede proponer al usuario un aislamiento de una fuente adecuada.

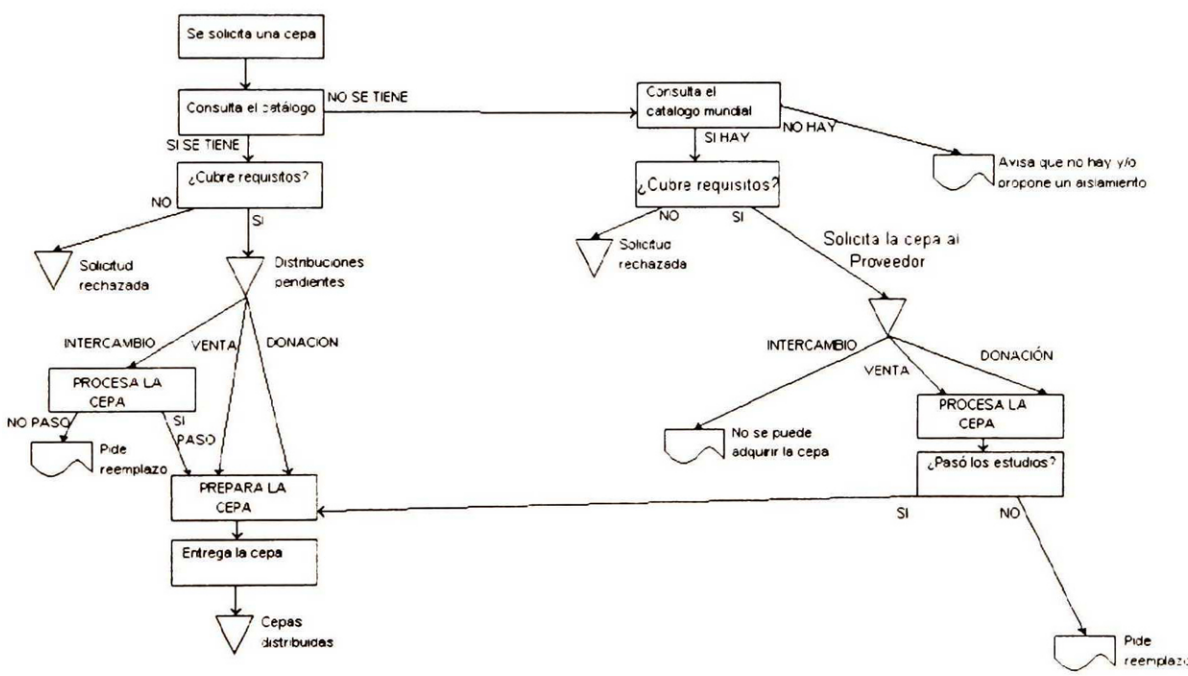


Figura II.1 Distribución de cepas

b) Recepción y deposito de cepas (figura II.2).- el usuario debe presentar la cepa junto con la información que tenga acerca de su cuidado y conservación, reservándose todos aquellos datos privados que le conciernan respecto a su producción y utilidades. Los datos se introducen al sistema computacional siempre que el usuario autorice por escrito, de otra forma se tendrán en un archivo especial. La cepa se procesa para su conservación y se le informa al usuario si ha pasado las pruebas, de resultar positivo y si se cubrieron los requisitos establecidos, se le informa que su cepa se aceptó en la colección, de otra forma se le notifica el rechazo.

Figura II.2 Recepción y depósito de cepas

c) Procesamiento de cepas (figura II.3).- este servicio se proporciona en algunos casos a usuarios interesados en conocer algún aspecto de interés acerca de una cepa, que puede ser alguna existente en la colección o traerse de otro lugar. El usuario debe llenar los requisitos establecidos en la colección previo a la realización del procesamiento de la cepa.

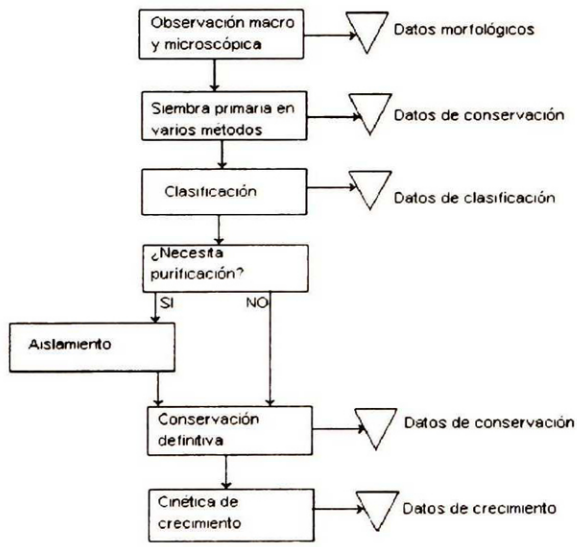


Figura II.3 Procesamiento cepas

d) Aislamiento de cepas (figura II.4).- se da cuando se tienen una muestra de una fuente de aislamiento donde está la cepa de interés. La muestra puede ser un tejido animal, algún órgano, una planta o parte de ella, agua, tierra, algún mineral, etc. Para realizar el aislamiento se deben cubrir los requisitos que la colección tiene establecidos.

Figura II.4 Aislamiento de cepas

e) Capacitación (figura II.5).- este servicio se ofrece de acuerdo con las especialidades que maneja el personal de la colección y/o experiencias en el manejo de cepas. Se requiere del establecimiento de un plan de actividades indicando el periodo y los puntos a cubrir. Esto puede implicar una estancia del interesado en las instalaciones de la colección o en las propias. La capacitación puede ser mediante un curso, asesoría, entrenamiento o una conferencia.

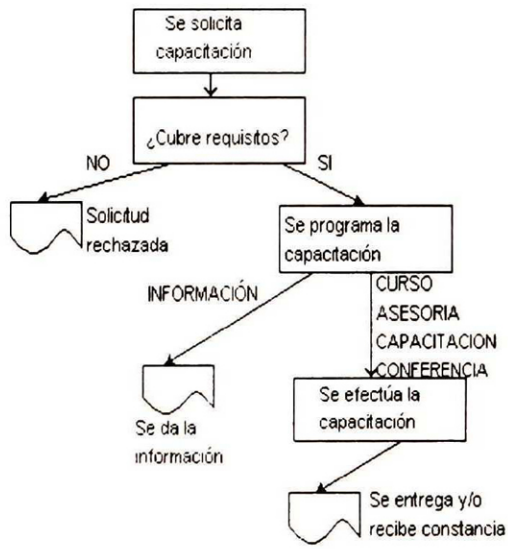


Figura II.5 Capacitación

f) Estudios (figura II.6).- los estudios se realizan tanto en cepas que proporciona el usuario, en alguna de la colección como en obtenidas de otro lugar. Los estudios pueden ser de: microscopía, morfología, conservación, crecimiento, clasificación, identificación. El usuario previo a la realización del servicio debe cubrir los requerimientos que la colección tenga establecidos.

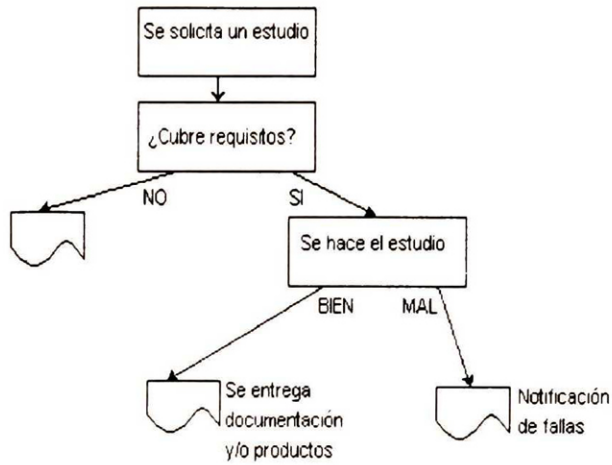


Figura II.6 Estudios a cepas

g) Otros servicios (figura II.7).- esto se refiere a servicios especiales que la colección puede realizar y que aun no se tienen establecidos de manera fija como lo es la aplicación de patentes, al igual que en los servicios anteriores, se deben de cubrir los requerimientos que la misma tenga establecidos.

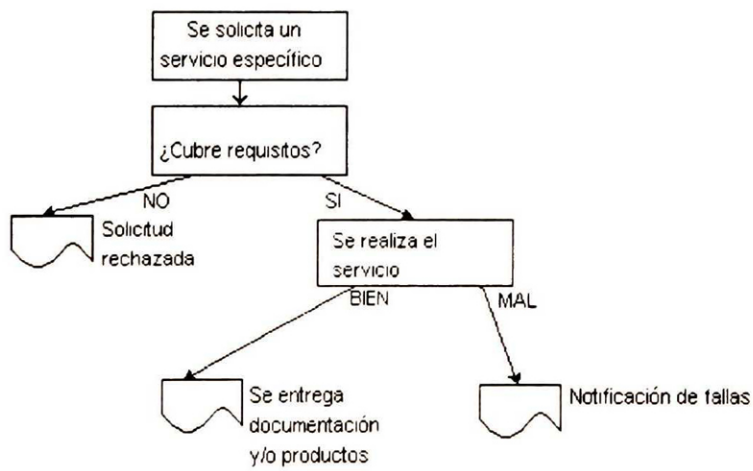


Figura II.7 Otros servicios

II.3 Requerimientos de información

En una colección de microorganismos se manejan miles de cepas, cada una caracterizada por cientos de atributos que le son propios. Esto hace que la urgencia y necesidad de un sistema computarizado que coadyuve al soporte de datos para la toma de decisiones y control sea algo prioritario. Es posible también contar con sistemas automatizados de apoyo a las actividades diarias de verificación de condiciones ambientales de las cepas y sus requerimientos, por lo que el uso de robots es una alternativa a considerar en futuros desarrollos.

Cada cepa dentro de la colección representa un proceso de importancia industrial, del que puede obtenerse uno o varios productos como: yoghurt, ácido cítrico, queso, vino, tequila, cognac, champagne, papel, licores, medicinas, drogas; o bien puede ser de utilidad para asimilar nitrógeno u otros compuestos en el suelo; pueden ser útiles para reciclamiento de la materia orgánica, o la limpieza del ambiente (agua, tierra, aire). Otro aspecto menos deseable pero igualmente importante es debido a la patogenicidad de algunos microorganismos tanto para el ser humano como para animales, plantas e incluso otros de ellos.

Actualmente los microorganismos representan una alternativa de solución a los grandes problemas de la humanidad como: contaminación, alimentación masiva y prevención de plagas, entre otros.

Cada cepa debe conservarse bajo ciertas condiciones de presión, temperatura, humedad, nutrientes y compuestos químicos. Esto se debe hacer de tal manera que se mantenga su pureza y viabilidad evitando la contaminación con otros microorganismos.

Los datos en el sistema computarizado deben estar "En-Línea", tanto para los usuarios externos como para el personal de la colección. Los datos pueden ser: nominales, imágenes o videos. En caso de tener sistemas de monitoreo y mantenimiento de cepas, estos deben ser sistemas en "Tiempo Real"

La información que se mantiene por el personal de la colección es crucial para garantizar las condiciones optimas de las cepas. Son necesarios cientos de atributos por cepa. El volumen de cepas alcanza los miles, lo cual da una idea del gran esfuerzo que se requiere para el registro y almacenamiento de los datos.

Los atributos que se manejan son de diferentes valores y naturaleza, como: patogenicidad, nivel de riesgo, fecha del estudio, temperatura, especie, subespecie, componentes, color, si tiene conidias, aislador, fuente de aislamiento, etc

La información se utiliza para:

a) Soporte a las decisiones, en el caso de determinar si se otorga una cepa a un usuario, si se debe hacer la resiembra de una cepa, si se debe utilizar determinadas condiciones para una cepa, si se debe aplicar un determinado método de conservación, etc.

b) Efectuar el control de las condiciones en que se encuentra una cepa como: temperatura, presión, humedad relativa, cantidad de compuestos químicos que se requieren, etc.

c) Propósitos estadísticos, los cuales le brindan a la colección índices que puede aprovechar en diversas modalidades. Como ejemplos se tienen: cuanto vive en promedio una determinada especie, cuales condiciones requiere para una mejor reproducción, que tipo de servicios se le demandan más a la colección y por quienes, etc.

d) Distribuirse bajo una cuota u otorgamiento, con base en las características del usuario y del compromiso que se tenga para su utilización.

Con el propósito de que la información sea útil y sirva para lo indicado anteriormente, debe estar disponible en todo momento incluyendo las provisiones de seguridad adecuadas.

El personal de la colección utiliza como fuentes de información: fotos, videos, observaciones macro y microscópicas, referencias bibliográficas (libros, revistas, resúmenes, etc.), anotaciones personales, consejos de otros especialistas, datos proporcionados por los usuarios, consultas telefónicas y por correo, resultados de experimentos y estudios.

Los datos colectados se refieren a diversos aspectos de interés que tiene la colección en cuanto a su acervo de microorganismos. Los datos se organizan principalmente en tablas, párrafos, gráficas, formatos predefinidos, fichas, libros y manuales.

Dada la cantidad de atributos y datos manejados en la colección, es obligatorio el uso de la tecnología de computación para almacenar, recuperar, procesar y manipular los datos.

Los datos manejados son textos, campos, fotografías, videos. Todos susceptibles de estar en la misma base de datos. De esto que sea importante contar con una base de datos orientada a objetos, en la cual las imágenes, videos, atributos nominales se almacenan como un objeto junto con sus métodos que permiten su recuperación y almacenamiento.

El "sistema de información" debe proporcionar facilidades para almacenamiento y actualizacion de los datos mediante pantallas apropiadas. Proporcionar los datos de dominio público mediante una interfaz amigable al usuario, teniendo las provisiones de seguridad apropiadas para evitar que puedan acceder a datos privados y valiosos. El sistema debe proporcionar en la misma pantalla, todos los tipos de información que requiera el usuario (nominales, gráficas, imágenes, videos, etc.).

Los reportes que deben emitirse con base al contenido de la base de datos son:

a) El catálogo de la colección [10], en donde cada cepa de dominio público, debe publicarse con una ficha estructurada con los datos de mayor importancia al usuario,
b) Localización de cada cepa,
c) Aplicaciones y cuales cepas la brindan,
d) Servicios pendientes y en curso,

El sistema debe proporcionar "En-Línea" los datos de dominio publico acerca del:
a) Catálogo de la colección actualizado, e
b) Información para usuarios de Internet, como cursos, congresos, etc.

Se debe tener un administrador de la base de datos quien coordine y supervise el mantenimiento de la consistencia y confiabilidad de las vistas en la base de datos.

Es posible utilizar robots diseñados ad-hoc que apoyen en los procesos de cuidado y mantenimiento de las cepas, los cuales utilicen la base de datos para su acción y toma de decisiones.

II.4 Modelo relacional de la base de datos

Las entidades de las diferentes vistas fueron expresadas en términos del modelo relacional [1] mediante un diagrama entidad-vinculo-extendido (EVE) (figura II.8) el cual se validó por un grupo de estudiantes de maestría de ciencias de la computación del CINVESTAV-IPN quienes hicieron contribuciones importantes en la depuración del modelo. Posteriormente un empleado del CINVESTAV-IPN, implantó el modelo bajo el manejador de base de datos ACCESS(TM) de Microsoft a quien se le apoyó para incluir los datos existentes en el sistema de información que se tenía.

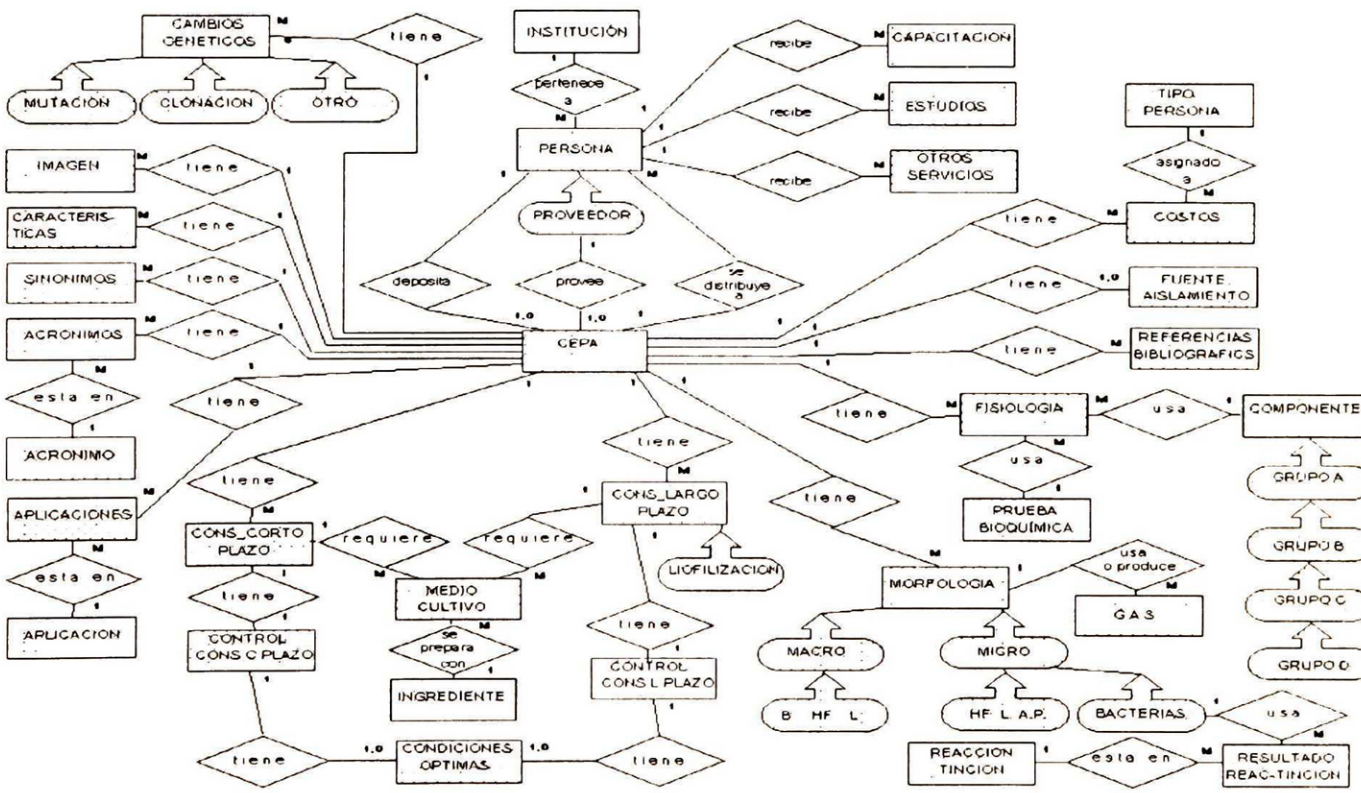


Figura II.8 Diagrama entidad-vínculo-extendido de la vista global de la base de datos

Para cada entidad se estableció una llave, la cual consiste de uno, dos, tres o hasta cuatro atributos. En cada tabla pueden encontrarse llaves foráneas que referencian otras entidades.

Se aplicaron las primeras tres formas normales a las entidades para minimizar la redundancia y facilitar la integridad y correctitud de los datos. En [13] se presenta una explicación amplia sobre el modelo relacional, incluyendo normalización, integridad de dominio, referencial e intrarelacional y la forma en que han evolucionado las bases de datos desde los sistemas de archivo, pasando por el relacional, semántico, el funcional, hasta el modelo orientado a objetos [10].

El enfoque relacional utiliza los diagramas entidad-vinculo-extendido propuestos por Chen en donde:

a) Los rectángulos representan a las entidades,
b) Los rombos representan vínculos entre entidades,
c) Las líneas con punta de flecha o sin ella, representan mapeos que se indican sobre ellas mediante un indicador (1 o M, según sea uno o muchos).

Los atributos se agrupan en entidades (tablas) en las cuales se define una llave única que identifica a cada tupla de la entidad.

Para manipular las tablas se usa el álgebra relacional con las operaciones: selección, proyección, junta natural y producto cartesiano, un ejemplo de ellos es el lenguaje SQL para consulta de bases de datos relacionales.

III VISTAS DEL USUARIO

Los atributos colectados son de naturaleza y valores diversos. Los atributos se se organizaron en un modelo lógico de base de datos, consistente de cinco vistas, mismas que se integran en la vista global de la base de datos mostrada en la figura II.8.

Las vistas de datos (figura III.1) a) catálogo, b) conservación, c) morfología, d) fisiología, e) servicios; se discuten en las secciones de este capítulo, presentando para cada una su diagrama EVE y las pantallas con las cuales se consultan y actualizan.

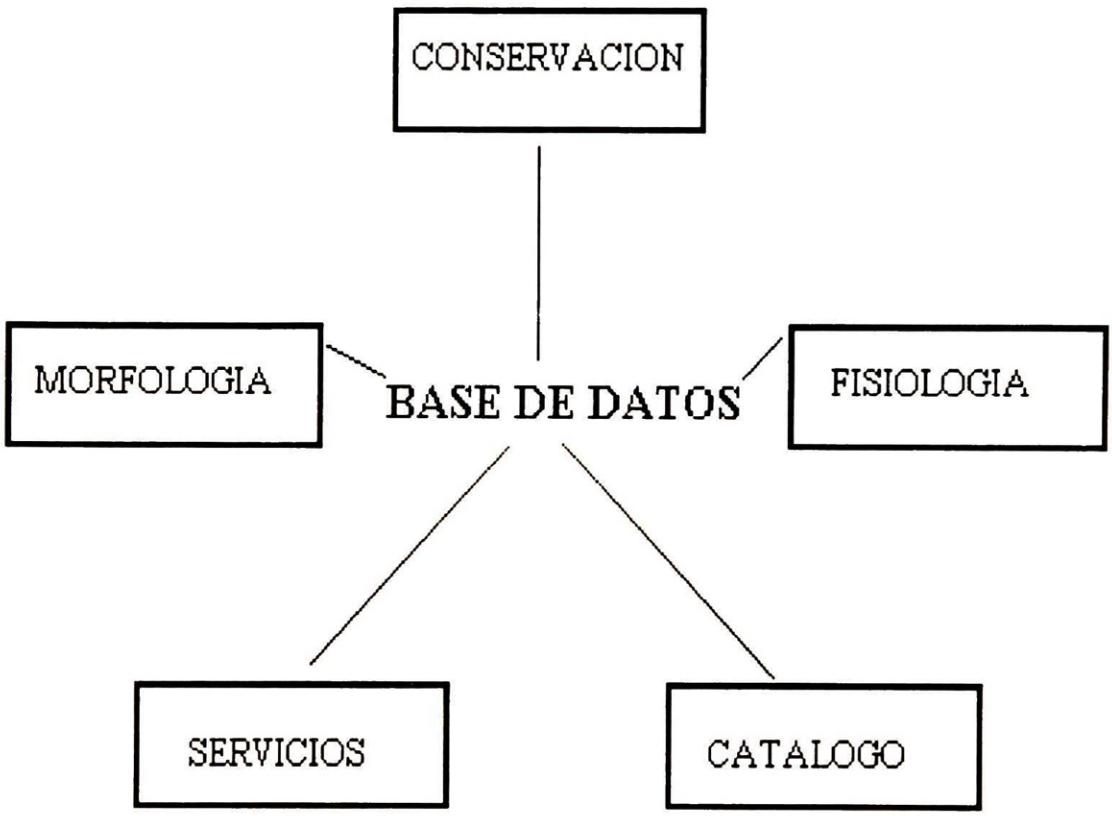
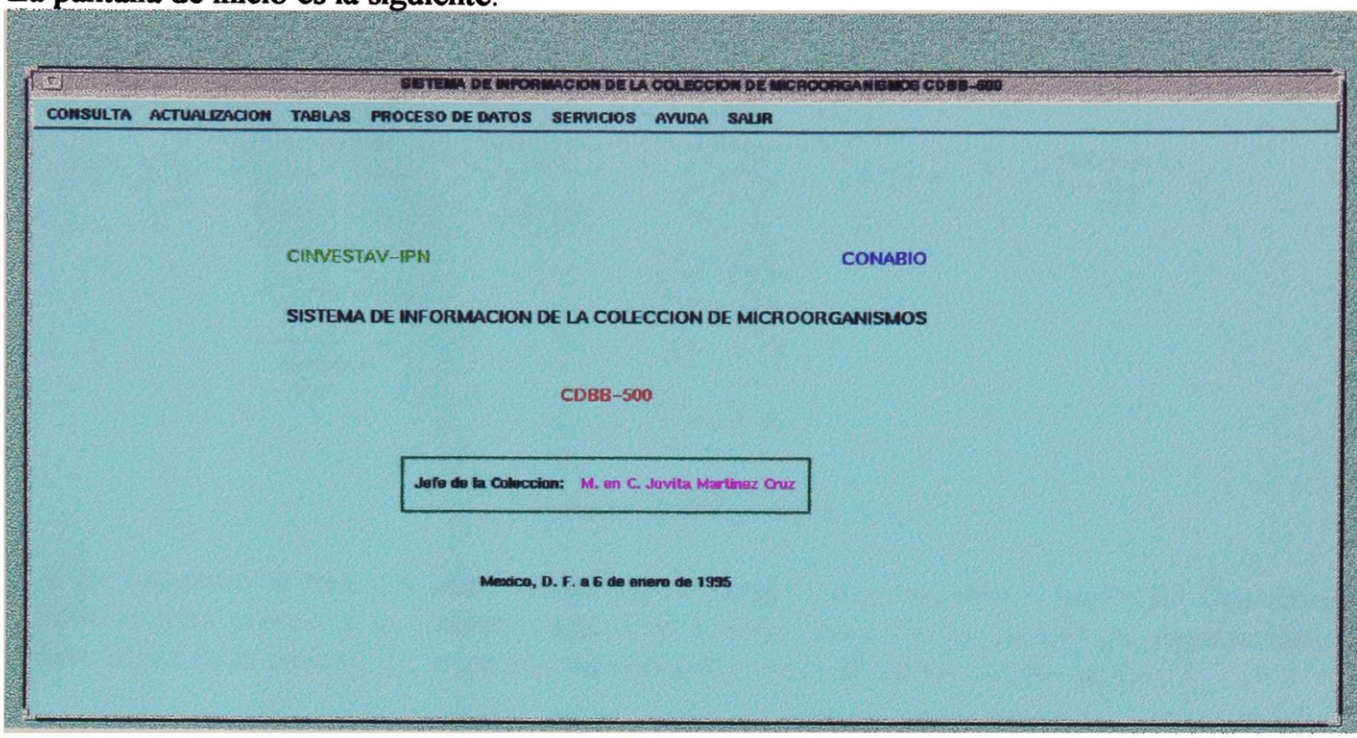


Figura III.1 Vistas de la base de datos

La pantalla inicial del sistema contiene el menú de opciones que le permiten al usuario acceder a los diferentes facilidades que brinda el sistema:

a) Consultas de datos;
b) Actualización de datos;
c) Tablas, captura y modificación de las tablas de datos utilizadas en el sistema;
d) Procesos, acceden y almacenan datos procesados:
e) Servicios, incluye los datos completos de los servicios.

La pantalla de inicio es la siguiente.



III.1 Catálogo

La vista del catálogo (figura III.2) comprende datos de dominio público. Las entidades que la forman son: datos taxonómicos (cepa), sinónimos, acrónimos, características, aplicaciones, imágenes de dominio público, referencias bibliográficas, y fuentes de aislamiento.

Cada cepa tiene un número único que la identifica dentro de la base de datos, un registro de datos taxonómicos, cero o más sinónimos, cero o más acrónimos descritos en un catálogo de acuerdo a la nomenclatura internacional, cero o más aplicaciones que se encuentran normalizadas y registradas en un catálogo, cero o más características normalizadas y registradas en un catálogo, cero o varias referencias bibliográficas tanto para características como para aplicaciones, cero o una fuente de aislamiento.

A partir de esta vista se constituye el catálogo de la colección el cual se imprime anualmente y se encontrará disponible "En-Línea" vía Internet.

El personal de la colección es el único que puede adicionar y actualizar los datos de esta vista. Los usuarios externos solo pueden visualizarlos.

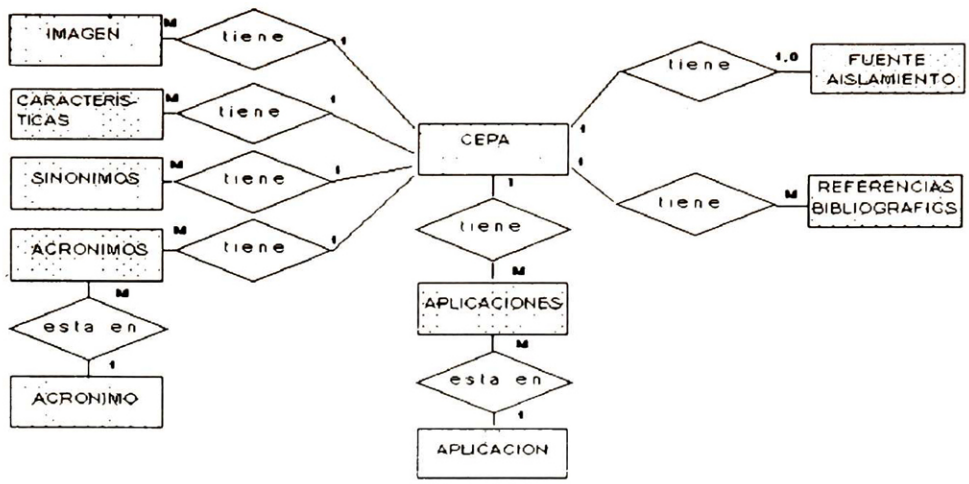


Figura III.2 Vista del catálogo

A continuación se muestra una instancia de las tablas que componen a esta vista (los datos son ficticios para preservar la confidencialidad de la colección). Las columnas en negritas forman la llave única de la entidad. El esquema completo de la vista global se encuentra en [1].

CEPA

númeroCDBB	genero	especie	. . .
D-1	animalus	pruebus	.
E-56	animalis	subspecis	. .

CARACTERISTICAS

numeroCDBB	**elemento**	descripcion
D-1	**1**	32
E-25	**1**	25

SINONIMOS

númeroCDBB	**elemento**	genero	especie
D-1	**1**	animalis	sinonis
D-1	**2**	animalis	glonis

APLICACIONES

numeroCDBB	**elemento**	descripcion
D-1	**1**	987
D-1	**2**	34

ACRONIMOS

númeroCDBB	**elemento**	acronimo
D-1	**1**	DAC
D-1	**2**	CATP

FUENTE AISLAMIENTO

numeroCDBB	origen	.
D-1	tierra	
D-3	agua	. . .

ACRONIMO

acronimo	descripcion
DAC	Dowell Alternative Collection
DPM	Dinamark Pollution Microbian

APLICACION

aplicacion	descripcion
1	produce oxígeno
2	asimila CO_2

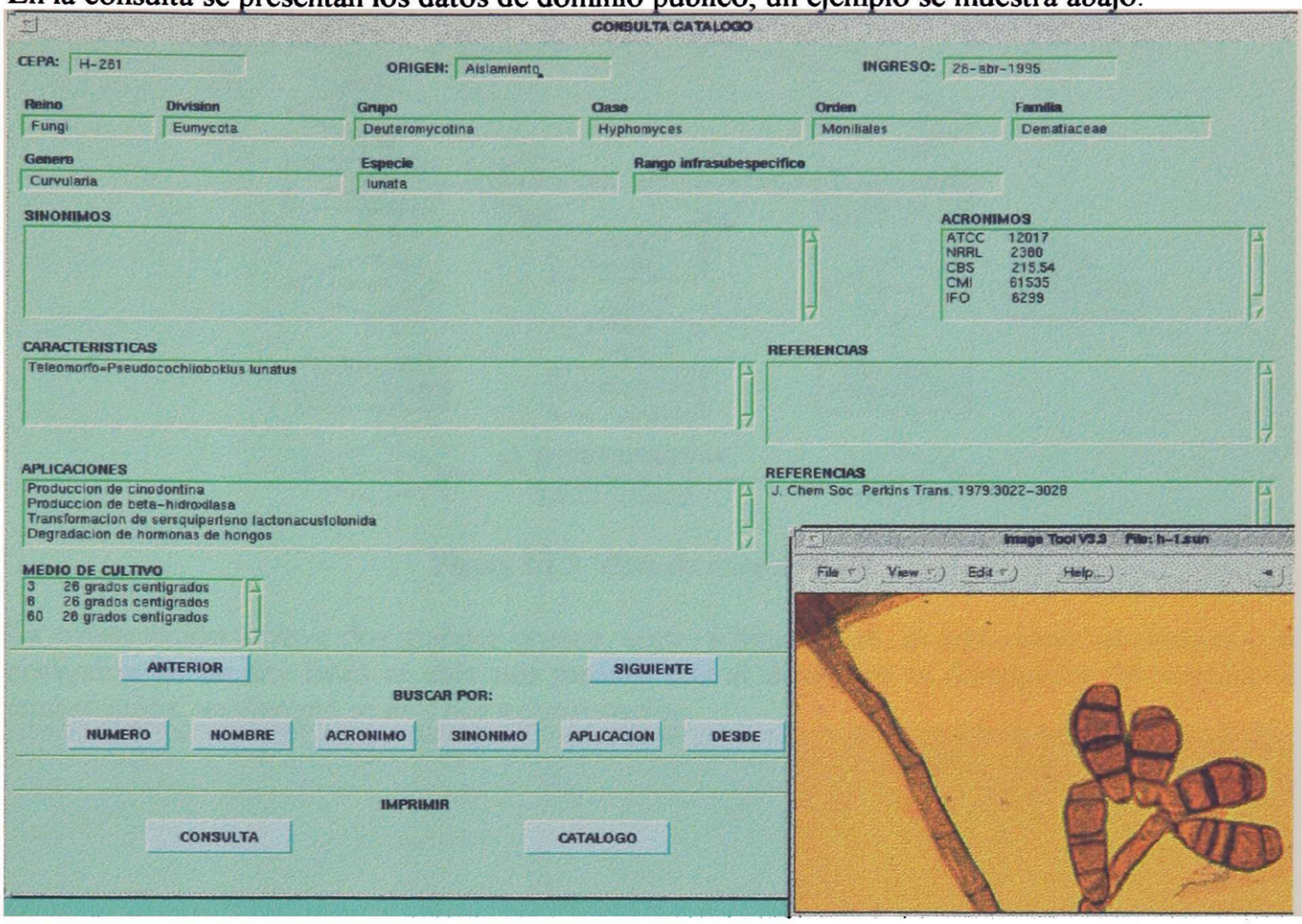
IMAGEN

númeroCDBB	**elemento**	imágen
D-1	**1**	010110101
D-1	**2**	001110101

REFERENCIAS BIBLIOGRAFICAS

numeroCDBB	**elemento**	referencia
D-1	**1**	Acta Microb p 80 v 2
D-1	**2**	Proceed. Microb. v. 18

En la consulta se presentan los datos de dominio público, un ejemplo se muestra abajo.



III.2 Conservación

Esta vista almacena la información relativa a los métodos y técnicas utilizados en la preservación de las cepas.

Hay diferentes métodos divididos en dos grupos:

a) Corto plazo,
b) Largo plazo.

Cada cepa puede tener varios registros para cada método de conservación, en donde se indican los nutrientes, compuestos químicos, productos, y otros aspectos de interés que deben considerarse. Si algún registro incluye condiciones optimas, esto se indica generando otro en la entidad de condiciones optimas, que pueden ser tanto de corto como de largo plazo.

Esta vista contiene datos privados para el personal de la colección, excepto aquellos referentes a los atributos temperatura y medio de cultivo de la tabla de condiciones optimas, los cuales son de dominio público y aparecen en el catálogo de la colección.

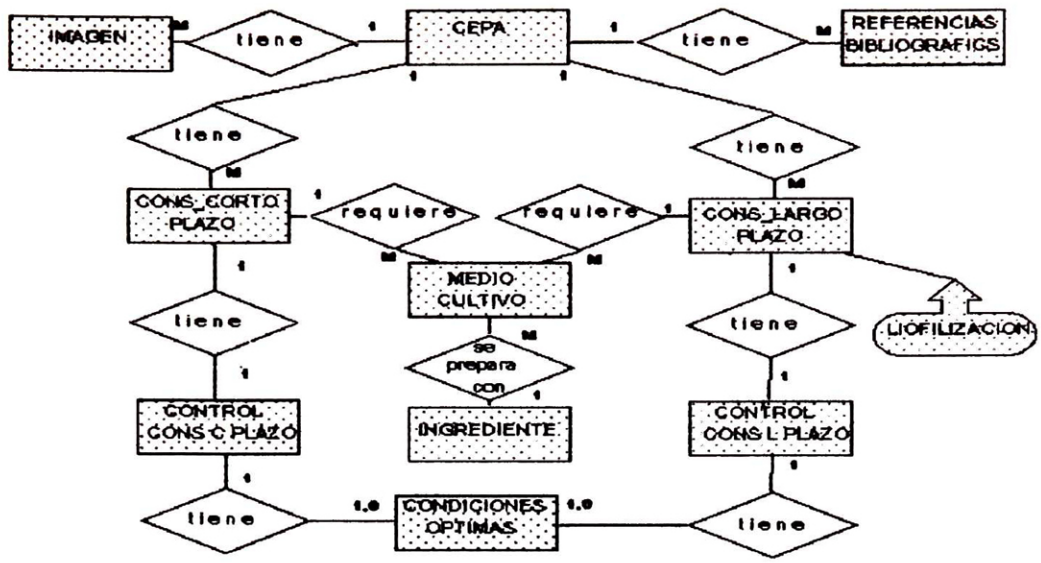
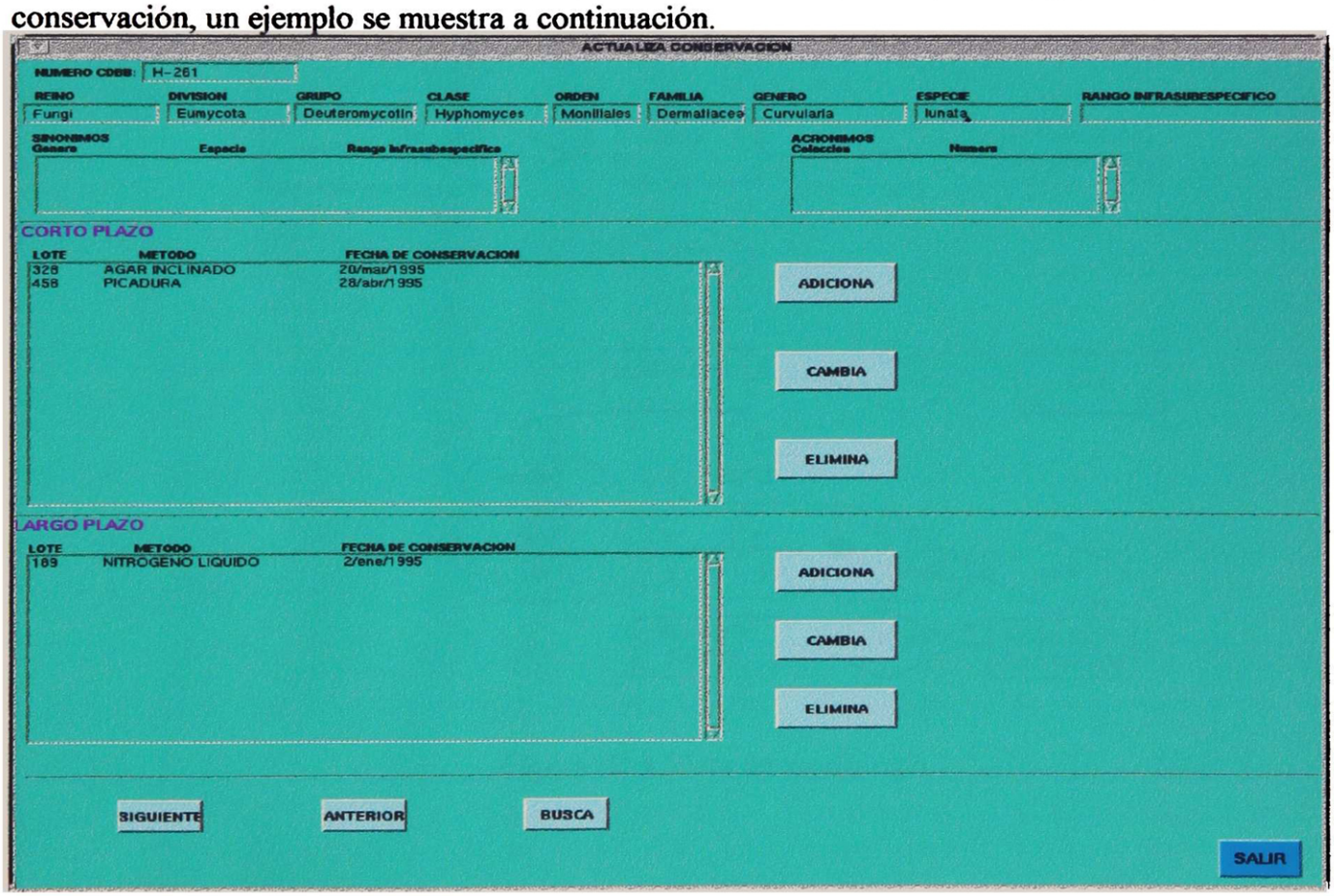


Figura III.3 Vista de conservación

La conservación incluye dos grandes grupos, corto y largo plazo, al seleccionar un elemento de cualquiera de las dos listas se abre una pantalla con el detalle de la descripción del método de conservación, un ejemplo se muestra a continuación.



III.3 Morfología

Los datos almacenados en esta vista se refieren a la forma de los microorganismos en sus diferentes fases de vida: crecimiento, reproducción, productiva, y muerte. Dado que el tipo de datos son imágenes, resulta conveniente que el sistema de información brinde los resultados en términos de este tipo de datos, lo cual ayuda al especialista del área a captar la información.

Los atributos almacenados son diferentes para cada tipo de microorganismo.

La división general se hace en cuanto a morfología:
a) Macroscópica,
b) Microscópica.

Las bacterias, hongos filamentosos y levaduras tienen morfología macro y microscópica.

Las microalgas y los protozoarios solo tienen micromorfología.

Se almacena un registro de morfología para cada estudio de cada cepa, con atributos característicos de cada uno.

Los datos de esta vista son privados y de uso exclusivo por el personal de la colección.

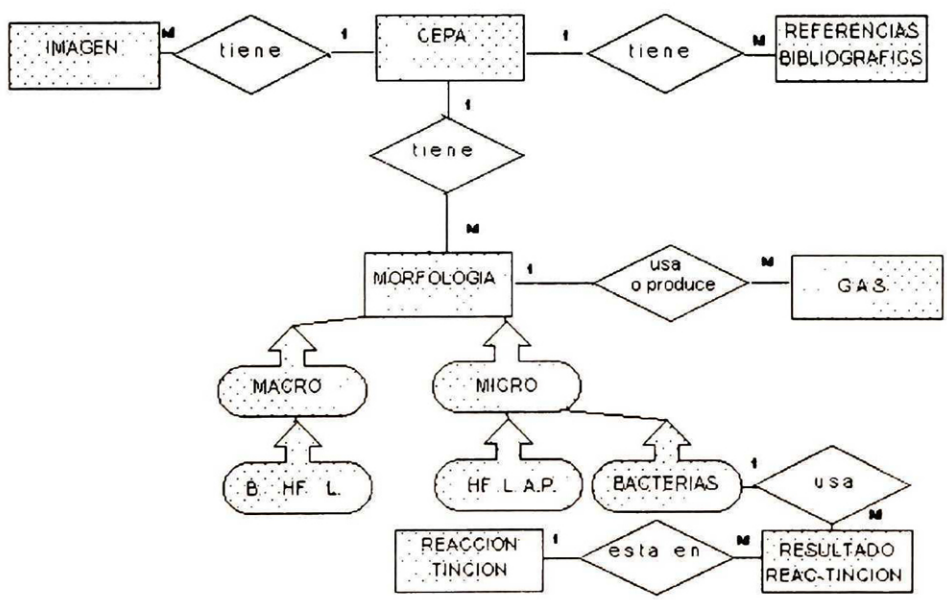
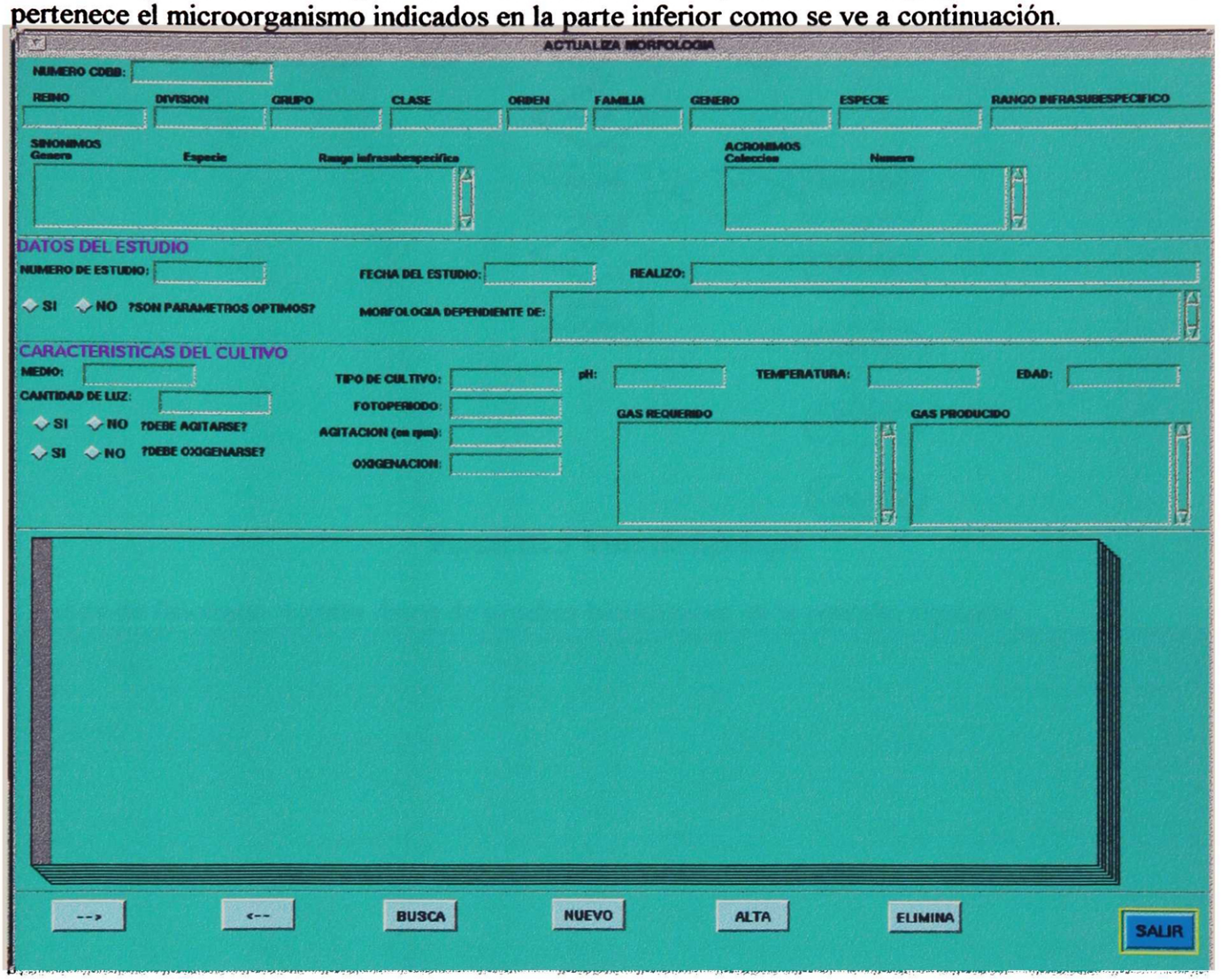


Figura III.4 Vista de morfología

Esta vista puede tener una gran variedad de atributos dependiendo del grupo taxonómico al que pertenece el microorganismo indicados en la parte inferior como se ve a continuación.



III.4 Fisiología

Los datos de esta vista tienen los resultados de las pruebas bioquímicas aplicadas a las cepas para clasificarlas e identificarlas.

Para cada prueba bioquímica se utiliza un compuesto de una familia de ellos y se obtiene un resultado que puede ser: positivo, muy positivo, negativo, muy negativo o nulo.

Los compuestos se encuentran agrupados en una jerarquía, el grupo de mayor nivel tiene un nombre genérico o puede ser un compuesto especifico. Los nombres genéricos a su vez pueden dividirse en otros nombres genéricos o específicos, así sucesivamente hasta un total de cinco niveles. El usuario puede elegir un nombre de cualquier nivel e indicar el resultado.

Los datos de esta vista son únicamente para uso del personal de la colección.

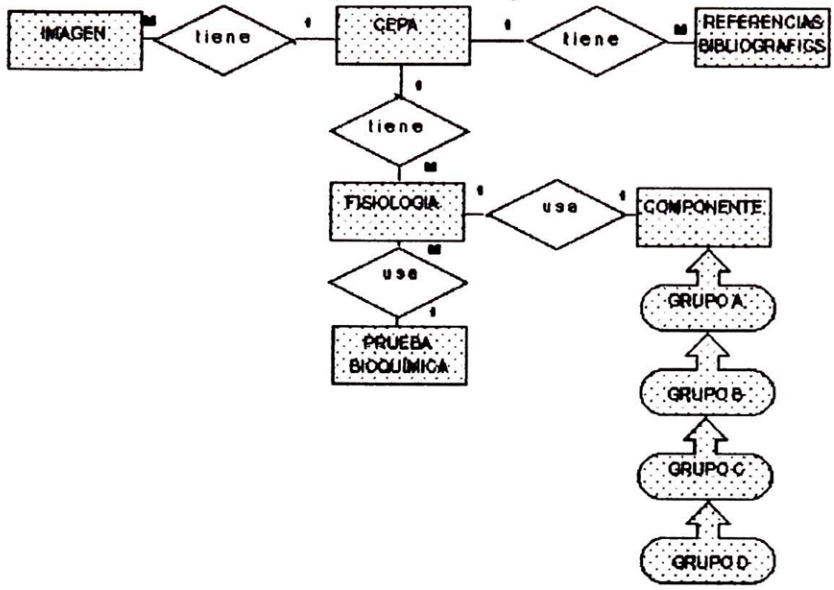
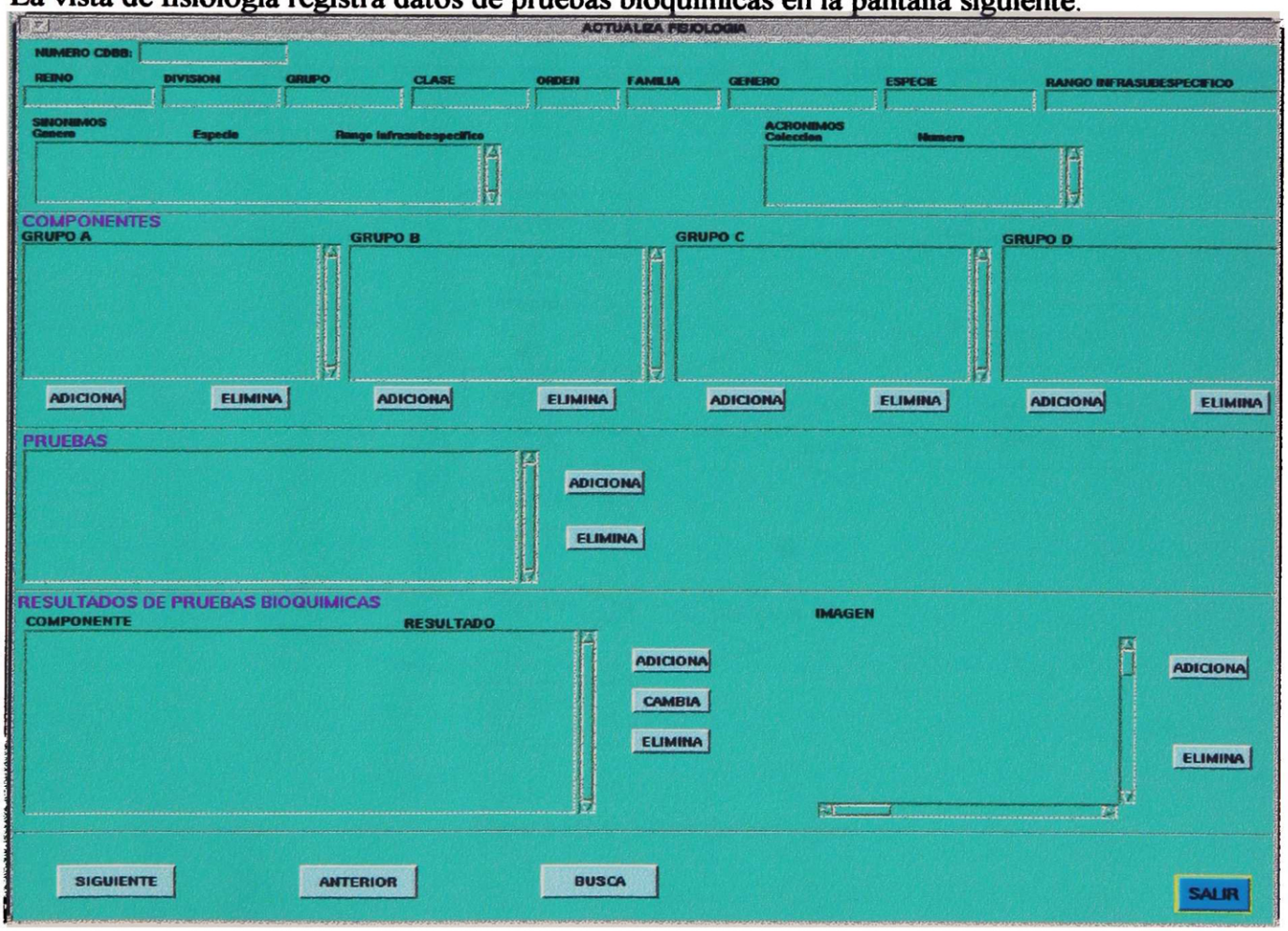


Figura III.5 Vista de fisiología

La vista de fisiología registra datos de pruebas bioquímicas en la pantalla siguiente.



III.5 Servicios

Esta vista contiene los datos de los servicios que brinda la colección usuarios externos. Dada la naturaleza diferente de cada servicio se tiene una entidad para cada uno de ellos y son:

a) Estudios,
b) Capacitación,
c) Depósito,
d) Distribución,
e) Proveedores,
f) Otros estudios,
g) Persona,
h) Institución.

En la entidad persona se almacenan las referencias a personas que se tienen en el sistema de información, tanto si es depositante, aislador, mutante, clonador, usuario de servicios, etc. Más de una persona puede pertenecer a una misma institución.

Cada usuario puede tener diferentes registros dentro de esta vista de acuerdo a los servicios que se le proporcionen. El solicitante de servicios siempre es una persona, en algunos casos puede representar a una institución, de tal forma que el contacto con instituciones es a través de una persona física.

El personal de la colección es el único que puede almacenar o actualizar datos de esta vista. Los datos contenidos aquí permiten conocer si es posible brindar un servicio y al mismo tiempo darle seguimiento.

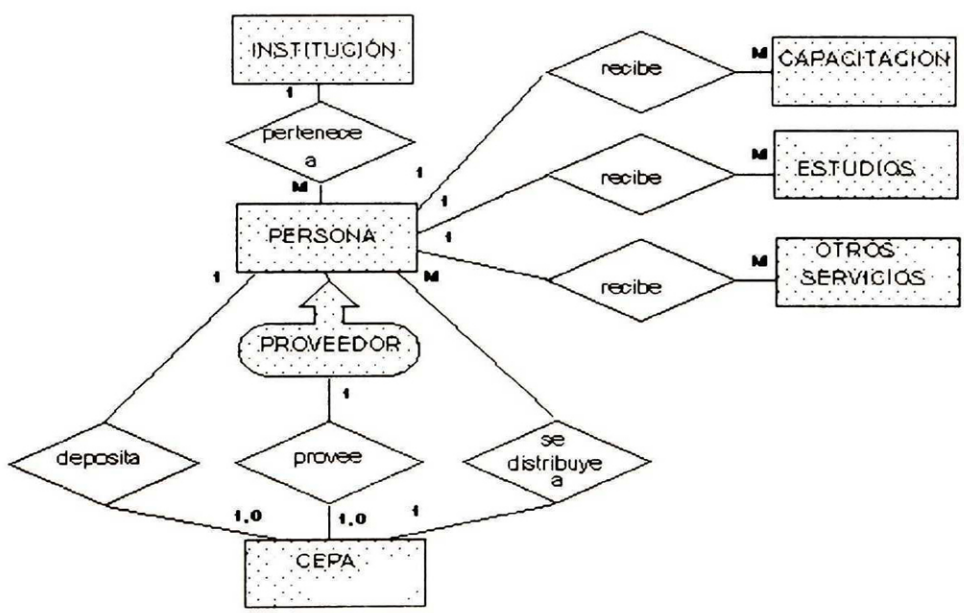
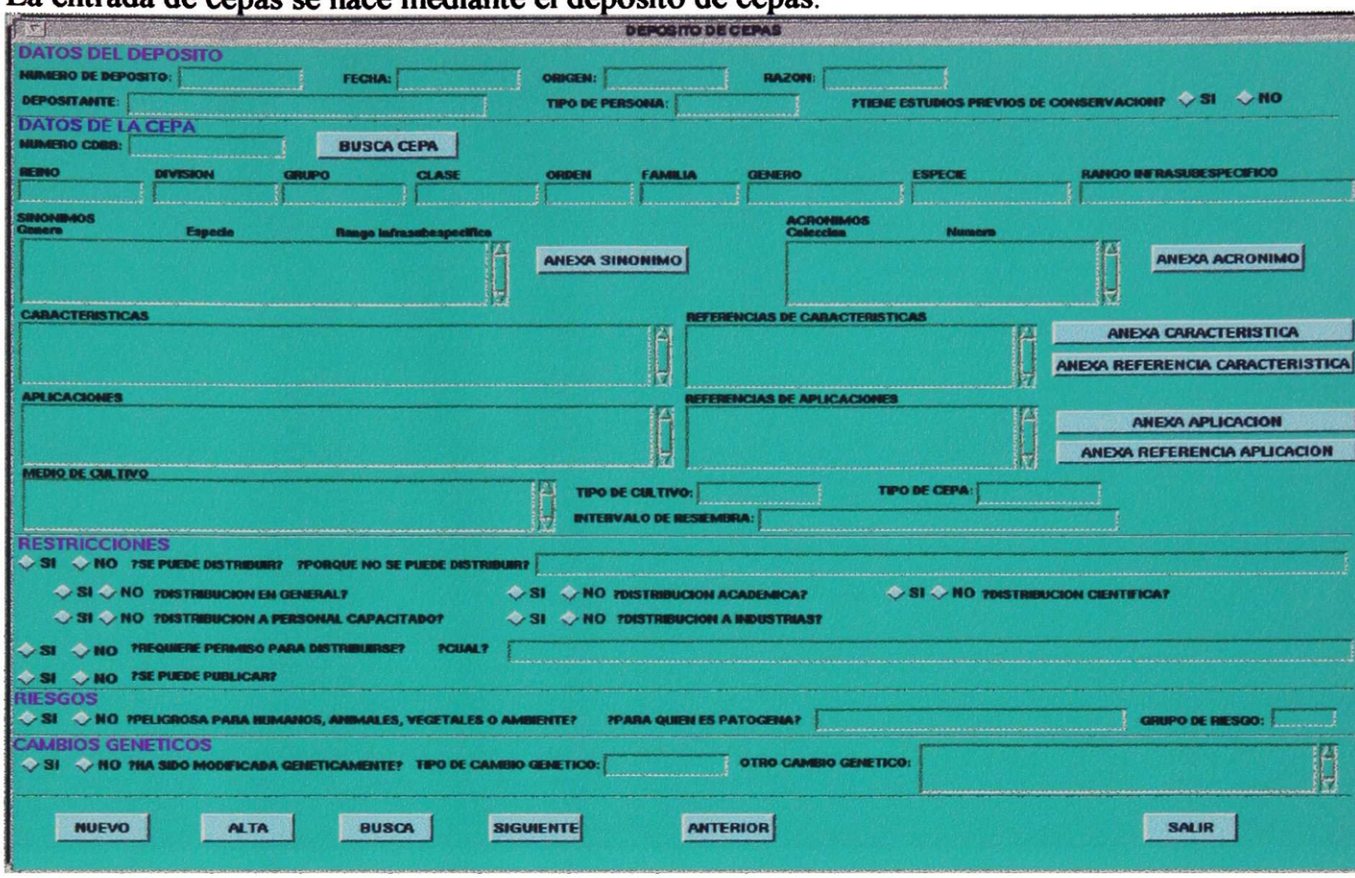
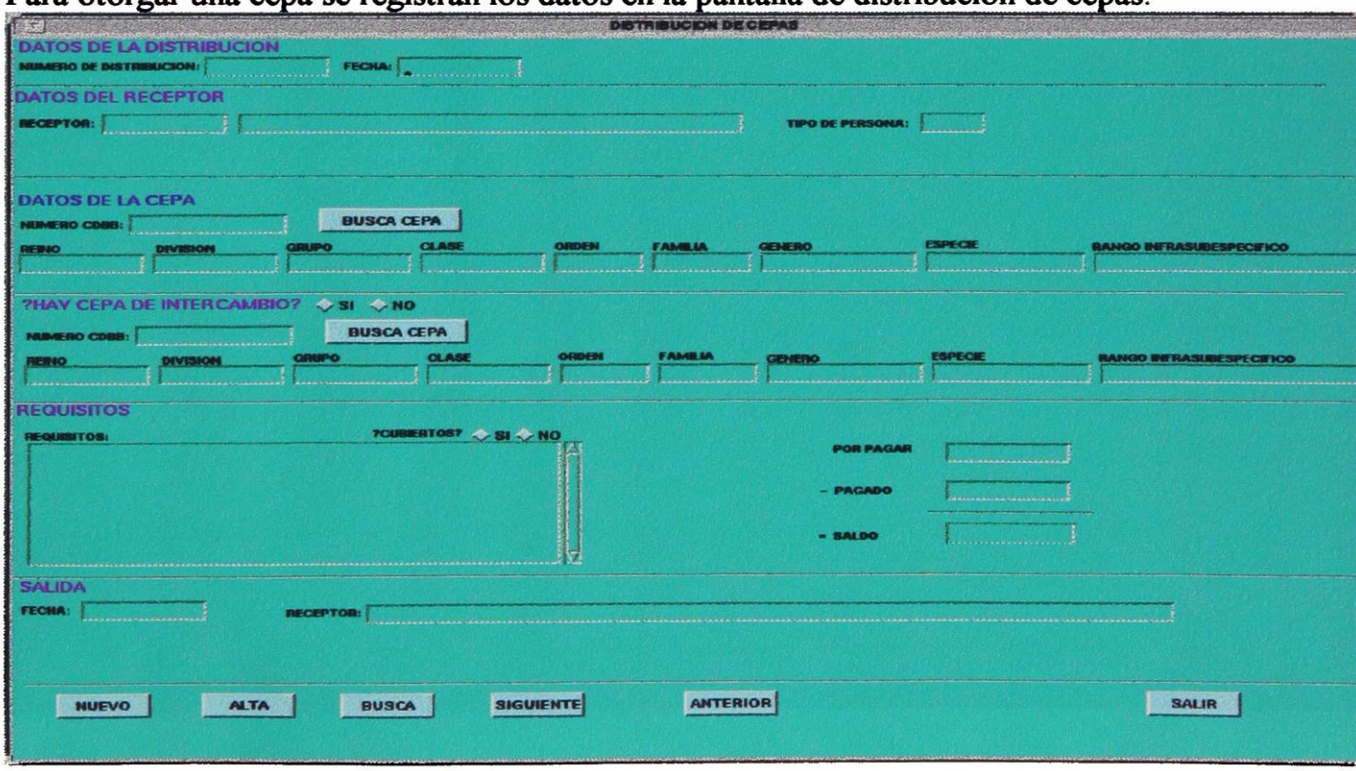


Figura III.6 Vista de Servicios

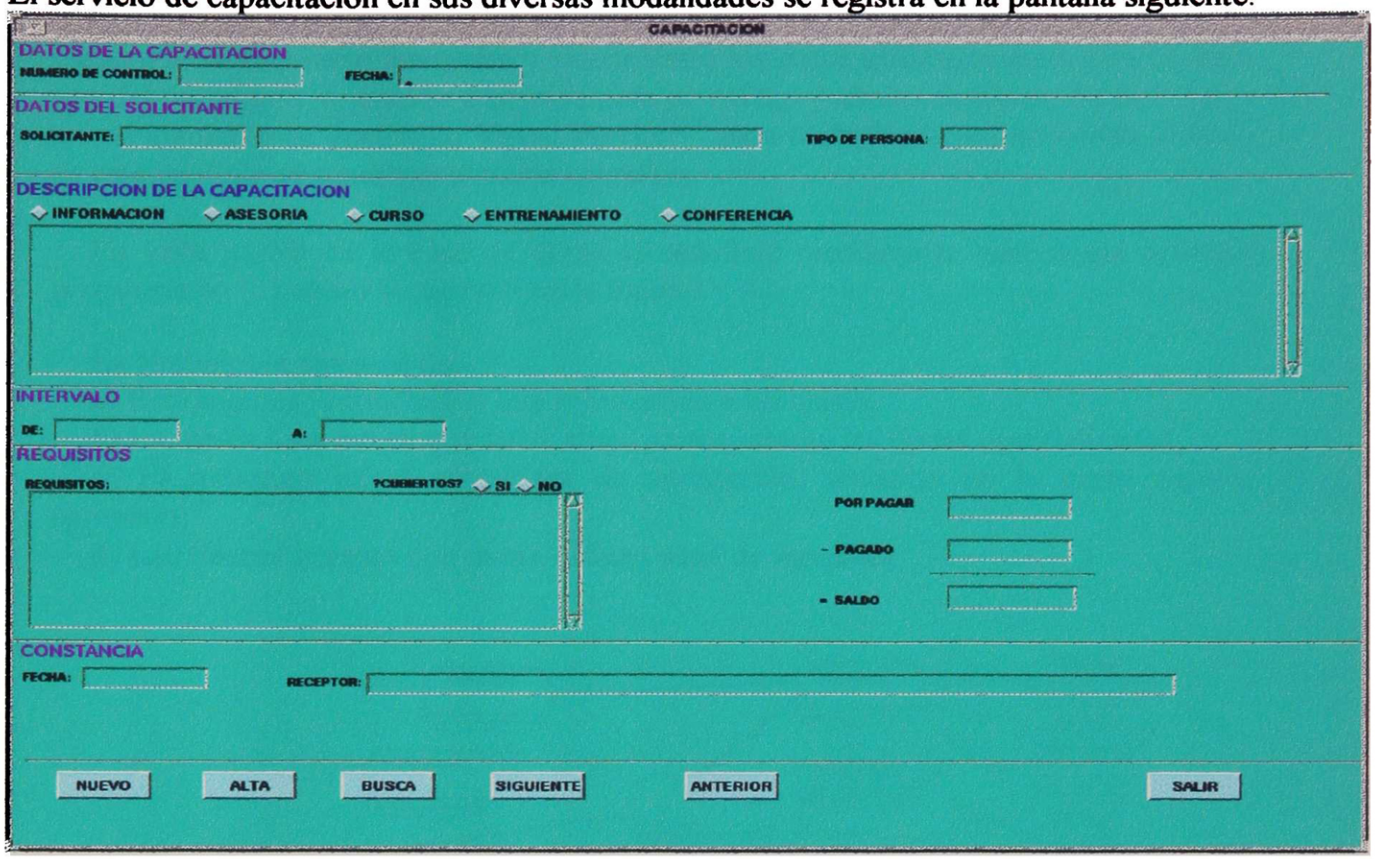
La entrada de cepas se hace mediante el depósito de cepas.



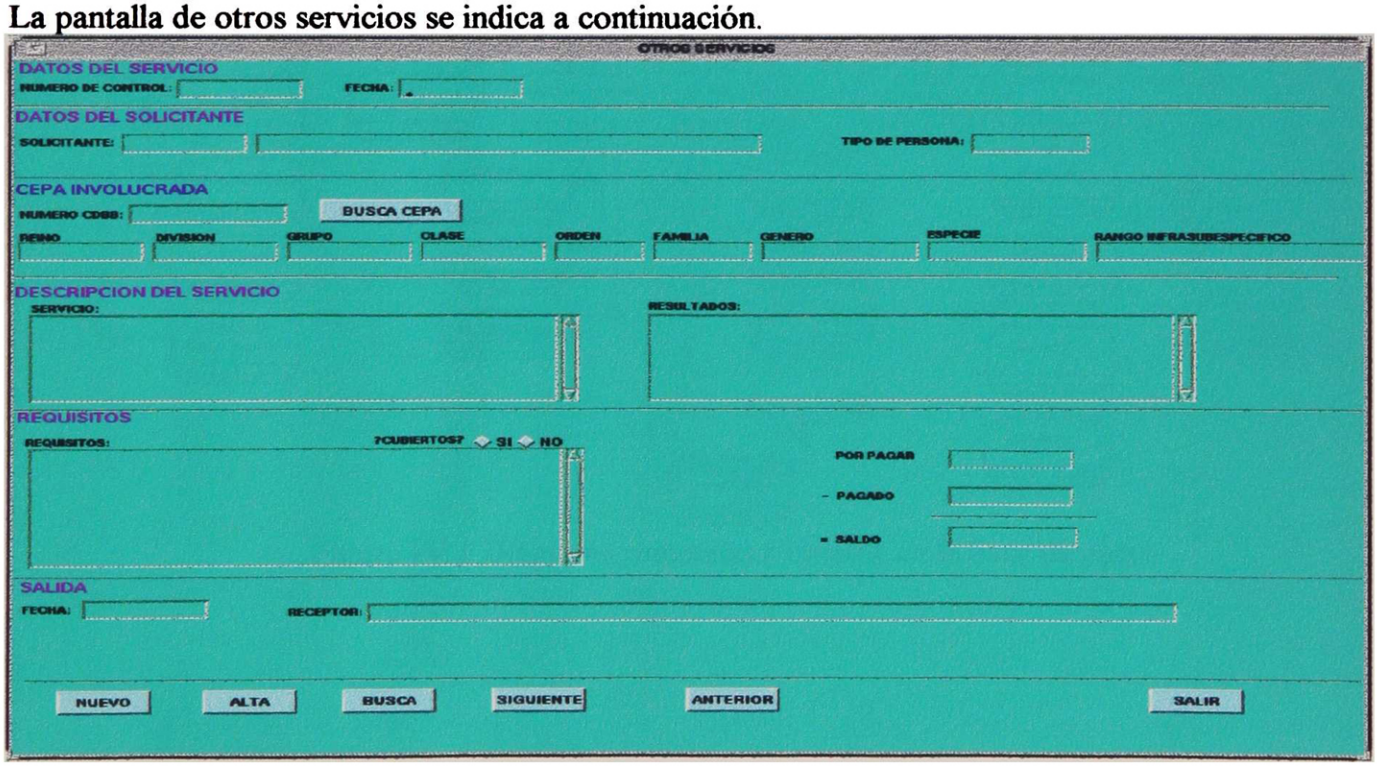
Para otorgar una cepa se registran los datos en la pantalla de distribución de cepas.



El servicio de capacitación en sus diversas modalidades se registra en la pantalla siguiente.



La pantalla de otros servicios se indica a continuación.



IV MODELO ORIENTADO A OBJETOS DE LA BASE DE DATOS

Dado que se requieren imágenes y videos entre los datos utilizados en la colección se propuso el uso del paradigma orientado a objetos para el manejo y almacenamiento de este tipo de datos. Se uso el ambiente de desarrollo Visual Works(TM) para la implantación orientada a objetos de la que en este capítulo se muestran algunos ejemplos.

La vista global de la base de datos se redefinió mediante la simbología orientada a objetos propuesta en [3] como se muestra en la figura IV.1.

La simbología consiste de:

a) Rectángulos redondeados que representan a las clases,

b) La media luna indica que hacia abajo se encuentran las clases especializadas,

c) El triángulo indica relaciones de agregación, teniendo en la parte superior al objeto agregado,

d) Las flechas gruesas con punta indican paso de mensajes.

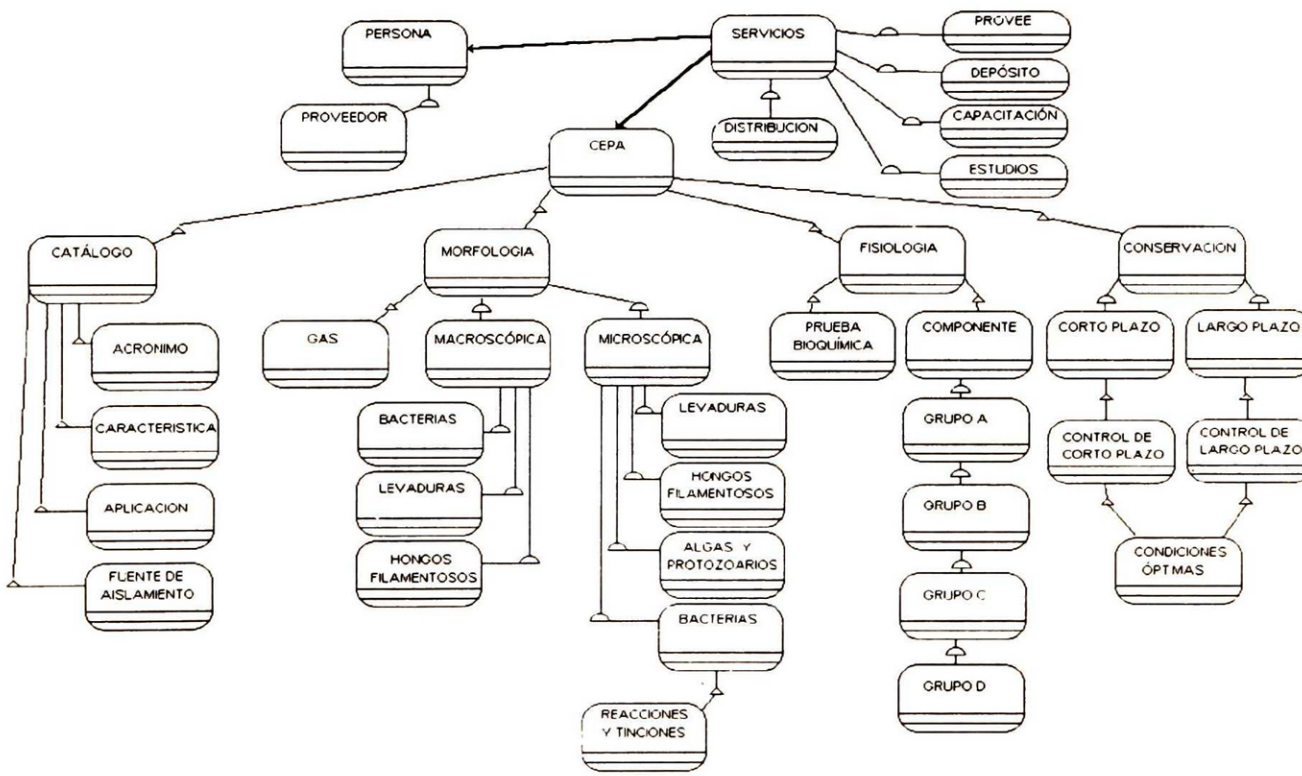


Figura IV.1 Diagrama orientado a objetos de la base de datos

IV.1 Componentes del sistema de información

En el sistema de información para la colección CDBB-500 la base de datos en el enfoque orientado a objetos, es el depósito de todos los datos que se manejan sin importar su tipo: nominales, pictográficos, reglas, señales, etc. Las imágenes se captan mediante un microscopio con una cámara de video incorporada, o mediante fotografías e imágenes digitalizadas.

Está propuesto que la interacción con el usuario y la base de datos sea mediante VisualWorks(TM), utilizando a GemStone (TM) como depósito de los objetos persistentes (figura IV.2). Las pantallas con las que interactúa el usuario se presentan en el capítulo III en este trabajo y se encuentran más detalladas en [1].

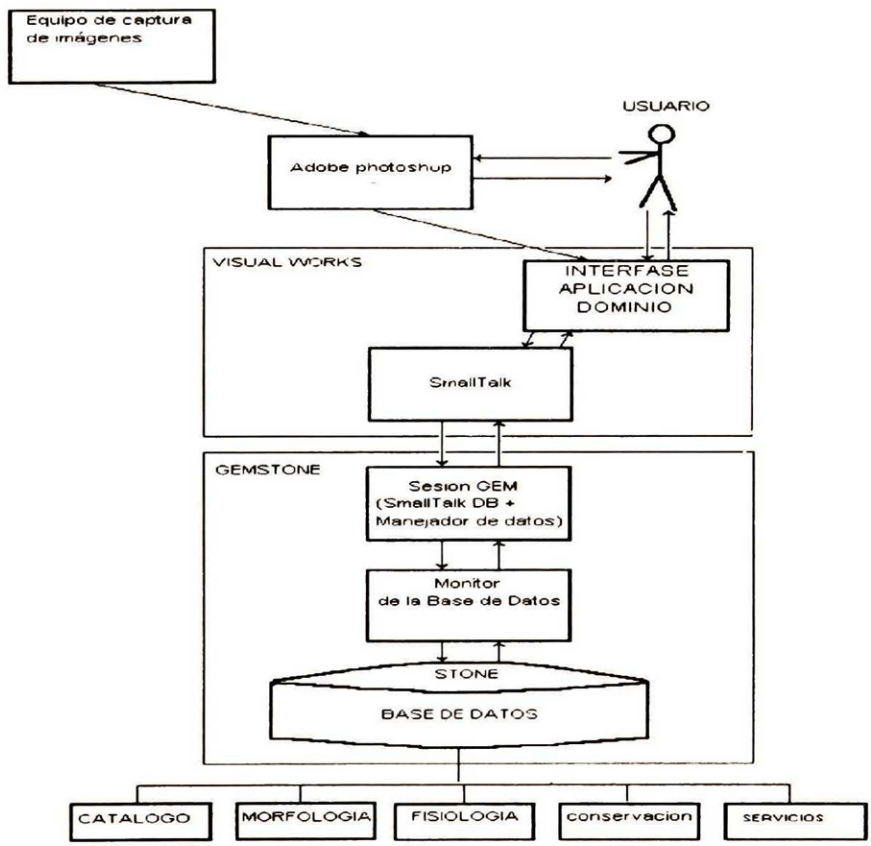


Figura IV.2 Arquitectura del sistema de información

IV.2 Métodos para actualización de datos nominales

En esta sección se muestran los objetos y métodos que implementan altas, bajas y cambios en objetos de tipo Dictionary (equivalentes a una tabla del modelo relacional) las cuales son las operaciones típicas en la actualización de datos.

El objeto tupla es ObjetoDeposito del listado IV.1. TablaDeposito del listado IV.2 representa el objeto tabla mediante el objeto interconstruido de smalltalk: Dictionary. El listado IV.3 lista al objeto que contiene los elementos de la interfase con el usuario para aplicar interactivamente las operaciones de actualización, indicadas en los métodos del mismo.

Listado IV.1 Objeto tupla

```
Autor: Jesús Manuel Olivares Ceja
Model subclass: #ObjetoDeposito
   "VARIABLES DEL OBJETO. numero ES LA LLAVE"
        instanceVariableNames: 'numero nombre descripcion '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'CDBB'!
!ObjetoDeposito methodsFor: 'ACCESO'!
descripcion
   ^descripcion!
descripcion: unaDescripcion
   descripcion := unaDescripcion!
nombre
   ^nombre!
nombre: unNombre
   nombre := unNombre!
numero
   ^numero!
numero: unNumero
   numero := unNumero!
printOn: unStream
   unStream nextPutAll: numero displayString, nombre displayString , descripcion displayString.!!
```

Listado IV.2 Objeto Tabla

```
Autor: Jesús Manuel Olivares Ceja
Model subclass: #TablaDeposito
        instanceVariableNames: 'tablaDeposito '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'CDBB'!
```

```
!TablaDeposito methodsFor: 'MÉTODOS'!
borraClave: unaClave
   tablaDeposito removeKey: unaClave ifAbsent: [nil].!
claveRepetida: unDeposito
   tablaDeposito at: (unDeposito numero) ifAbsent: [^false].
   ^true!
creaDeposito
|unDeposito|
   unDeposito := ObjetoDeposito new.
   unDeposito numero: ''
   unDeposito nombre: ''.
   unDeposito descripcion: ''.
   ^unDeposito!
daDeposito: unaClave
   ^tablaDeposito at: unaClave ifAbsent: [nil].!
daTablaDeposito
   ^tablaDeposito!
estaClave: unaClave
   tablaDeposito at: unaClave ifAbsent: [^false].
   ^true!
exporta
   tablaDeposito do:
    [:registro|
      Transcript show: registro;cr.
   ]!
guardaDeposito: unDeposito
   self daTablaDeposito at: (unDeposito numero) put: unDeposito.!
initialize
   tablaDeposito := Dictionary new.! !
TablaDeposito class
        instanceVariableNames: ''!
!TablaDeposito class methodsFor: 'INICIALIZA'!
new
   ^super new initialize! !
```

Listado IV.3 Interfase para las operaciones de actualización en un objeto Dictionary

```
Autor: Jesús Manuel Olivares Ceja
ApplicationModel subclass: #CDBBServicioDeposito
        instanceVariableNames: 'dato datoTablaDeposito clave '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'CDBB'!
!CDBBServicioDeposito methodsFor: 'ACCIONES'!
```

hazAltaServicioDeposito

```
    "REALIZA LA ALTA DEL SERVICIO DE DEPOSITO"
|unDeposito elDeposito|
   self dato value: datoTablaDeposito creaDeposito.
   unDeposito := self openDialogInterface: #cdbbAltaDeposito.
   elDeposito := (self dato value) copy.
   unDeposito ifTrue:
   [
      (datoTablaDeposito claveRepetida: elDeposito) ifTrue:
      [
         Dialog warn: 'NO APLICA LA ALTA, CLAVE REPETIDA'.
      ]
      ifFalse:
      [
         datoTablaDeposito guardaDeposito: self dato value.
      ]
   ]
   ifFalse:
   [
      Dialog warn: 'NO se realizo la alta'.
   ]!
```

hazBajaServicioDeposito

```
    "BORRA UN DEPOSITO DEL SERVICIO DE DEPOSITO"
| claveBien confirmaBaja |
   claveBien := self openDialogInterface: #leeClave.
   claveBien ifTrue:
   [
      (datoTablaDeposito estaClave: clave value) ifTrue:
      [
         self dato value: (datoTablaDeposito daDeposito: clave value) copy.
         "DESPLIEGA LOS DATOS ACTUALES"
         confirmaBaja := self openDialogInterface: #confirmaBaja.
         "CONFIRMA LA  BAJA"
         confirmaBaja ifTrue:
         [
            datoTablaDeposito borraClave: clave value.
            Dialog warn: 'BAJA REALIZADA'.
         ]
         ifFalse:
         [
            Dialog warn: 'NO SE REALIZO LA BAJA'.
         ]
      ]
      ifFalse:
      [
```

```
        Dialog warn: 'NO EXISTE LA CLAVE ', clave value.
      ]
    ]!
hazCambioServicioDeposito
    "CAMBIA LOS DATOS DE UN DEPOSITO"
| claveBien confirmaCambio  depositoCambiado|
    claveBien := self openDialogInterface: #leeClave.
    claveBien ifTrue:
    [
      (datoTablaDeposito estaClave: clave value) ifTrue:
      [
        self dato value: (datoTablaDeposito daDeposito: clave value) copy.
        "DESPLIEGA LOS DATOS ACTUALES"
        confirmaCambio := self openDialogInterface: #cdbbAltaDeposito.
        "CONFIRMA LOS CAMBIOS"
        depositoCambiado := (self dato value) copy.
        confirmaCambio ifTrue:
        [
          datoTablaDeposito borraClave: clave value.
          datoTablaDeposito guardaDeposito: depositoCambiado.
          Dialog warn: 'CAMBIOS REALIZADOS'.
        ]
        ifFalse:
        [
          Dialog warn: 'NO SE REALIZO LOS CAMBIOS'.
        ]
      ]
      ifFalse:
      [
        Dialog warn: 'NO EXISTE LA CLAVE ', clave value.
      ]
    ]!
hazSalirServicioDeposito
    "CIERRA EL OBJETO DE LA INTERFASE"
    self closeRequest.!
```

IV.3 Métodos para desplegado de datos pictográficos

La visualizacion de imágenes se implanto mediante un programa desarrollado en una plataforma PC bajo el sistema operativos DOS el cual decodifica los archivos PCX (listado IV 4) y deja en forma de un archivo de texto los valores correspondientes a cada pixel. El archivo de valores se transfiere a la estacion de trabajo y se importa dentro de Visual Works(TM), ahi se despliega mediante el programa de importacion de imágenes en smalltalk mostrado en el listado IV.6.

Listado IV.4 Programa decodificador de archivos de imágenes PCX.

Basado en el artículo A C++ PCX File Viewer for Windows 3 en Dr Dobb's Journal, July 1991, Autor: Paul Chiu. Adaptó: Jesús Manuel Olivares Ceja noviembre 1994

```
#include <graphics.h>
#include <stdio.h>
#include <process.h>
#include <conio.h>

#define NO_HAY -1
#define SI 1
#define NO 0
#define BITS_POR_BYTE 8

struct PcxRGB {
    unsigned char r ,g ,b ;
} ;

struct PcxHeader {
   unsigned char fabricante ;
   unsigned char version ;
   unsigned char codificacion ;
   unsigned char bits_pixel ;
   int x_min ,y_min ;
   int x_max ,y_max ;
   int res_horiz ,res_vert ;
   struct PcxRGB paleta [16] ;
   unsigned char reservado ;
   unsigned char planos ;
   int bytes_linea ;
   int inf_paleta ;
   unsigned char relleno[58] ;
};
// VARIABLES
struct PcxHeader Encabezado ;
int XPant ,YPant ,NumCols ;
int Plano ; // PLANO DE COLOR QUE SE DECODIFICA
unsigned int LineaVideo[1000] ;
```

```
int ULineaVideo ,ILineaVideo ;
FILE *Archivo;
void _ModoGrafico(int cual_modo)
/* ASIGNA UN MODO GRAFICO EN LA PANTALLA DE VIDEO */
{
int tarjeta
,  modo_graf  ;

   switch( cual_modo )
   {
   case 1:
      tarjeta = VGA;
      modo_graf = VGAHI;
      break;
   case 2:
      tarjeta = VGA;
      modo_graf = VGAMED;
      break;
   default:
      tarjeta = DETECT;
      break;
   }
   initgraph( &tarjeta, &modo_graf, "" );
}
void leepcx_PintaByte          (unsigned char byte)
/* DESPLEGA EL CONTENIDO DE UN BYTE */
{
int color ,num_bit ,valor_bit ;

   for( num_bit = 0; num_bit < 8; num_bit++ )
   {
      valor_bit = 0;
      color = byte & 0x80;
      if( color == 0x80 )
      valor_bit = 1;
     LineaVideo[ILineaVideo + num_bit] += valor_bit << Plano;
      byte = byte << 1;
   }
   ILineaVideo += 8;
}
void main(void)
{
FILE *arch_pcx ;
long u_reg_pcx ,reg_pcx ;
unsigned char byte ;
```

```
int repite ,itera ,num_byte ,columna ;
char area[50] ;

   /* LEE EL NOMBRE DEL ARCHIVO */
   printf("ARCHIVO PCX 16 COLORES: ");
   gets(area);
   /* ABRE EL ARCHIVO DE ENTRADA */
   arch_pcx = fopen(area ,"rb");
   if( arch_pcx == NULL )
   {
      printf("ERROR No existe el archivo %s\n\n" ,area);
      exit(0);
   }
   /* TOMA LA LONGITUD DEL ARCHIVO EN BYTES */
   fseek(arch_pcx ,0L ,SEEK_END);
   u_reg_pcx = ftell(arch_pcx);
   /* LECTURA DEL ENCABEZADO DEL ARCHIVO PCX */
   fseek(arch_pcx ,0L ,SEEK_SET);
   fread(&Encabezado ,128 ,1 ,arch_pcx);
   if( Encabezado.planos != 0x04 || Encabezado.bits_pixel != 0x01 )
   {
      printf("ERROR Sólo puedo leer 16 colores\n\n");
      exit(0);
   }
  _ModoGrafico(1);
  Archivo = fopen("puntos.dat" ,"wt");   // ARCHIVO DE SALIDA
 XPant = 0;
 YPant = 0;
 /* DECODIFICA CADA LINEA DE LA IMAGEN */
 for( reg_pcx = 128L, reg_pcx < u_reg_pcx; )
 {
    for( ILineaVideo = 0; ILineaVideo < 1000; ILineaVideo++ )
       LineaVideo[ILineaVideo] = 0;
   for( Plano = 0; Plano < Encabezado.planos; Plano++ )
   {
       ILineaVideo = 0;
       for( columna = 0; columna < Encabezado.bytes_linea; )
       {
          /* LEE UN BYTE */
          fseek(arch_pcx ,reg_pcx ,SEEK_SET);
          fread(&byte ,1 ,1 ,arch_pcx);
          reg_pcx++;
          /* SI ES REPETICION LEE EL BYTE SIGUIENTE Y DUPLICALO */
          if( byte >= 0xC0 )
          {
```

```
            repite = byte & 0x3F;
            fseek(arch_pcx ,reg_pcx ,SEEK_SET);
            fread(&byte ,1 ,1 ,arch_pcx);
            reg_pcx++;
           for( itera = 0; itera < repite; itera++ )
           {
              leepcx_PintaByte(byte);
             columna++;
           }
         }
         else
         /* SOLO PINTA EL BYTE */
         {
            leepcx_PintaByte(byte);
            columna++;
         }
      }
    }
    for( ILineaVideo = 0; ILineaVideo < Encabezado.bytes_linea * 8; ILineaVideo++ )
    {
      fprintf(Archivo ,"%d\n" ,LineaVideo[ILineaVideo]);
      if( LineaVideo[ILineaVideo] == 0 )
         putpixel(XPant ,YPant ,BLACK);
     else
         putpixel(XPant ,YPant ,LineaVideo[ILineaVideo]);
      XPant++;
      if( XPant > (BITS_POR_BYTE * Encabezado.bytes_linea - 1) )
      {
         YPant++;
         XPant = 0;
      }
    }
}
 fclose(Archivo);
 fclose(arch_pcx);
 restorecrtmode();
} // fin del programa
```

Listado IV.6 Desplegador de imagenes en SmallTalk

```
leeImagenDeposito
ventana contexto x y i color max_x      "VARIABLES PARA DESPLEGADO GRAFICO"
archivo entero contenido simbolo valor unsimbolo     "VARIABLES PARA ARCHIVO" |
ventana := ExamplesBrowser prepareScratchWindow.
ventana label: 'Imagen de la CEPA'.
ventana openWithExtent: 1280 @ 480.
```

```
contexto := ventana graphicsContext.
x := 1.   y := 1.   max_x := 640 * 2.
archivo := 'puntos.dat' asFilename.    "ARCHIVO DE ENTRADA"
contenido := archivo contentsOfEntireFile.
entero := archivo fileSize.
simbolo := '       '    unsimbolo := ' '    color := 0.
Transcript show: entero displayString;cr.
i := 1.
1 to: entero do:
 [ :indice|
      (contenido at: indice) = Character cr
      ifTrue: [
         color == 0 ifTrue: [contexto paint: ColorValue black].
         color == 1 ifTrue: [contexto paint: ColorValue blue].
         color == 2 ifTrue: [contexto paint: ColorValue green].
         color == 3 ifTrue: [contexto paint: ColorValue cyan].
         color == 4 ifTrue: [contexto paint: ColorValue darkRed].
         color == 5 ifTrue: [contexto paint: ColorValue darkMagenta].
         color == 6 ifTrue: [contexto paint: ColorValue salmon].
         color == 7 ifTrue: [contexto paint: ColorValue lightGray].
         color == 8 ifTrue: [contexto paint: ColorValue darkGray].
         color == 9 ifTrue: [contexto paint: ColorValue royalBlue].
         color == 10 ifTrue: [contexto paint: ColorValue springGreen].
         color == 11 ifTrue: [contexto paint: ColorValue lightCyan].
         color == 12 ifTrue: [contexto paint: ColorValue red].
         color == 13 ifTrue: [contexto paint: ColorValue magenta].
         color == 14 ifTrue: [contexto paint: ColorValue yellow].
         color == 15 ifTrue: [contexto paint: ColorValue white].
         contexto displayLineFrom: x @ y to: x @ y.
         x >= max_x
         ifTrue: [ y := y + 1. x := 1.]
         ifFalse: [ x:= x + 1.].
         1 to: i do: [:k| simbolo at: k put: (Character space)].
         i := 1.
         color := 0.
         x :=  x + 1.          ]
      ifFalse: [     simbolo at: i put: (contenido at: indice).
         unsimbolo := (contenido at: indice).
         valor := (unsimbolo asInteger) - ($0 asInteger).
         color := color * 10.
         color := color + valor.
         i := i + 1        ]. ].
ventana display.  "DESPLEGA LA IMAGEN"
(Delay forSeconds: 5) wait.
```

V BASES DE DATOS ORIENTADAS A OBJETOS

Debido a las dificultades en la creación y reutilización de código en la década de los 60s los lenguajes de programación se rediseñaron simplificando su estructura, modularizandolos, y produciendo código reusable. Asimismo surge la disciplina de Ingeniería de Software. Los programas ya no se consideraron líneas de código sino módulos interrelacionados. Esto dió origen al surgimiento de metodologías estructuradas para el diseño de sistemas, las cuales perduran hasta nuestros dias. En estas un sistema de información se considera estructurado como una colección de módulos cada uno de los cuales desarrolla una función específica dentro del mismo.

Con los lenguajes orientados a objetos, se tienen objetos como una característica del lenguaje, donde cada uno pertenece a una clase y con la herencia se forma una jerarquía. Cada objeto representa una entidad la cual tiene un conjunto de operaciones que determinan su comportamiento. El estado de un objeto, correspondiente a sus estructuras de datos, es privado y se manipula mediante los mensajes (métodos) que se le envían. Esto se hace considerando los principios de abstracción de datos, modularidad y "ocultamiento de información"[13].

V.1 Características comunes

Un sistema manejador de base de datos orientado a objetos (SMBDOO) debe satisfacer dos criterios[14]:

a) Ser un sistema manejador de base de datos (SMBD) y
b) Ser un Sistema orientado a objetos (OO).

Por el primer criterio se tienen cinco características: persistencia, administración del almacenamiento secundario, concurrencia, recuperabilidad y facilidades para consultas no planeadas.

El segundo criterio implica ocho características: objetos complejos, identidad de objetos, encapsulación, tipos o clases, herencia, polimorfismo con encadenamiento dinámico, extensibilidad y completitud computacional.

a) Objetos complejos.- Son aquellos que se forman mediante otros más simples aplicándoles constructores. Los objetos simples son enteros, caracteres, cadenas de bytes, booleanos y flotantes. Los constructores son: tuples, conjuntos, bolsas (bags), listas y arreglos. Los constructores mínimos que debe proporcionar un sistema son conjuntos, tuples y listas. Los constructores deben ser ortogonales a los objetos, esto es, cualquier constructor puede aplicarse a cualquier objeto. Los constructores de modelo relacional no satisfacen este requerimiento porque el constructor conjunto solo puede aplicarse a tuples y el constructor tuple solo puede aplicarse a valores atómicos.

b) Identidad en objetos.- Es un concepto utilizado desde hace mucho en los lenguajes de programación pero es reciente en bases de datos. La idea es que un objeto es independiente de su valor, así dos objetos pueden ser idénticos (son el mismo objeto) o iguales (tienen el mismo valor). La identidad en los lenguajes de programación se da asignando un nombre a cada variable a la cual

le corresponde una dirección física en memoria, pero el concepto es nuevo para los sistemas relaciónales los cuales son basados en valor.

c) Encapsulación.- Esta característica proviene de la necesidad de distinguir entre la especificación y la implementación de una operación y de la necesidad de modularizar. La modularización se necesita para estructurar aplicaciones complejas implementadas por un grupo de desarrolladores.

La idea de encapsulación proviene de los tipos de datos abstractos. En este caso, un objeto tiene una parte de interfase y una parte de implementación. La interfase es el conjunto de operaciones que puede realizarse en el objeto, es la única parte visible del objeto. La parte de implementación tiene una parte de datos y una parte de procedimientos. La parte de datos describe la representación del estado del objeto. La parte de procedimientos describe en algún lenguaje de programación a la implementación de cada operación. Al extenderlo a bases de datos se tiene que los objetos encapsula programas y datos.

La encapsulación proporciona una forma de "independencia lógica de datos", ya que se puede cambiar la implementación de un tipo sin cambiar a los programas que usan el tipo.

La abstracción es un proceso cognoscitivo que permite describir cada objeto del mundo real mediante sus propiedades relevantes, ignorando las propiedades no relevantes para un propósito especifico; las propiedades no consideradas pueden adicionarse en cualquier momento en el modelo.

d) Tipos y clases.- Hay dos categorías de sistemas orientados a objetos, aportan la noción de clases y aquellos que soportan la noción de tipo. En la primera categoría se tienen sistemas como SmallTalk, GemStone, Visión, y en general los sistemas de la familia smalltalk; también Orión, Flavors, G-Base, Laure y en general los sistemas derivados de Lisp. En la segunda categoría se tienen sistemas como C++, Simula, Trellis/Owl, Vbase y O2.

Un tipo, en un sistema orientado a objetos, resume las características comunes de un conjunto de objetos. Corresponde a la noción de tipo de datos abstracto. Tiene dos partes la interfase y la implementación. Únicamente la interfase es visible a los usuarios del tipo; la implementación del objeto es visible sólo a los diseñadores. La interfase consiste de una lista de operaciones junto con sus parámetros (cada uno de ellos tiene un tipo). En los lenguajes de programación, los tipo son herramientas para incrementar la productividad asegurando la correctitud de los programas, forzando a que el usuario declare los tipos de datos que se deben manejar. Esto se utiliza a tiempo de compilación que no puede modificarse a tiempo de ejecución (run time).

La noción de clases es diferente de la de tipo La especificación de una clase es similar a la de tipo, pero varia en el sentido de que es una noción de tiempo de ejecución. Contiene dos aspectos. una fábrica de objetos y un ejecutor de objetos. La fábrica se utiliza para crear objetos nuevos, mediante la aplicación de la operación "new" El ejecutor permite que las instancias de la clase las pueda manipular el usuario aplicando las operaciones que le pertenecen vía los mensajes.

Las clases no se usan para verificar la correctitud sino para crear y manipular objetos.

La elección entre tipos y clases se deja al implantador.

e) Herencia.- Esta característica tiene dos ventajas, la primera es que brinda una descripción del mundo adecuada y segundo, ayuda en la implementación de de aplicaciones. Por ejemplo si se tiene una aplicación que tiene empleados y estudiantes. Las entidades Empleado y Estudiante pueden compartir atributos comunes, pero en vez de implementar a cada uno como entidades diferentes, se puede definir una entidad común Persona de la que se derivan los Estudiantes y los Empleados, y definiendo los métodos exclusivos de cada uno en sus definiciones derivadas de Persona.

Hay al menos cuatro tipos de herencia: sustitutiva, inclusiva, restrictiva y especializativa. La herencia sustitutiva, se da cuando en un tipo o clase t heredada de un tipo t' se pueden realizar más operaciones en el tipo t' La herencia inclusiva corresponde a la noción de clasificación, establece que un tipo t es un supertipo de un tipo t' si cada objeto de tipo t también es un objeto de tipo t' La herencia restrictiva es un subcaso de la herencia inclusiva, un tipo t es un subtipo de un tipo t' si cada objeto de t no sólo es un objeto de tipo t' per también satisface la restricción dada. Por ejemplo Joven es un subtipo de Persona, Joven no tiene más atributos que Persona pero obedece a la restricción edad menor a 30. La herencia de especialización es cuando un tipo t es un subtipo de t' si los objetos de tipo t son objetos de tipo t' pero con información más especifica Por ejemplo un Empleado es una Persona pero con atributos extra como salario.

Se pueden entremezclar los diferentes tipos de herencia mencionados.

La jerarquía es la forma natural en que se organizan los objetos dentro de un sistema dado Los objetos se agrupan en clases que son las que contienen el esquema de las instancias de esa clase, también llamados objetos. Puede haber clases que heredan las propiedades de otras clases llamadas heredadoras. A la clase que hereda se le nombre subclase y a la heredadora se le nombra superclase.

f) Polimorfismo y encadenamiento dinámico.- Hay casos en los que se quiere usar un mismo nombre para diferentes operaciones. Considere por ejemplo, la operación despliega, la cual toma como entrada un objeto y lo despliega en la pantalla. Dependiendo del tipo de objeto, se puede querer utilizar diferentes mecanismos para desplegar. Si el objeto es un tuple se querrá una forma de desplegado de tuples, si el objeto es una gráfica, se querrá su representación gráfica, etc En una aplicación convencional se tendrían las operaciones despliega_tuple, despliega_grafica, despliega_video. En un sistema orientado a objetos se define la operación despliega en el nivel mas alto en la jerarquía e implementar para cada objeto la operación particular que requiera, redefiniendo en cada caso el método despliega. A esto se le conoce como polimorfismo, un mismo mensaje se comporta diferente dependiendo del objeto al que se invoque y que tiene implementado su método de distinta forma.

Para proporcionar esta funcionalidad, el sistema no encadena los nombres a tiempo de compilación. Sino que las operaciones se resuelven (se traducen a direcciones de programa) a tiempo de ejecución. Esta traducción retrasada se le llama Encadenamiento Dinámico Esto hace que la verificación de tipo sea más dificil (en algunos casos imposible).

g) Extensibilidad.- Esta característica se refiere a tener la posibilidad de extender los tipos predefinidos en el sistema para definir tipos nuevos sin hacer distinción entre los tipos definidos en el sistema y los tipos definidos por el usuario que igualmente se usan para crear otros.

h) Completitud computacional.- Se refiere a que se pueda expresar cualquier función computable utilizando el lenguaje de manejo de datos del sistema, en base de datos esto es hasta cierto punto novedoso ya que el lenguaje SQL no es completo. La completitud computacional es diferente de la "completitud de recursos" , esta última se refiere a la posibilidad de utilización de todos los recursos del sistema (monitor, impresoras, comunicaciones remotas, etc.).

i) Persistencia.- Es un concepto nuevo en los lenguajes de programación pero común en las base de datos. Es la propiedad de los datos de sobrevivir a través de ejecuciones múltiples de un proceso, de tal forma que es posible el uso del dato persistente por otro proceso. El usuario no debe indicar explícitamente que el objeto se haga persistente, esto debe ser automático.

j) Administración del almacenamiento secundario.- Es una característica clásica de los SMBD, se soporta a través de mecanismos como manejo de índices, grupos de datos, buffers, optimización de consultas. Ninguno de estos debe estar visible al usuario. El programa de aplicación tampoco debe incluir código para mantener alguna de las facilidades mencionadas.

k) Concurrencia.- Consiste en que el SMBDOO proporcione servicios armoniosamente a más de un usuario simultaneamente.

l) Recuperabilidad.- El sistema debe proporcionar el mismo servicio de recuperabilidad como los actuales sistemas de base de datos, tanto si hay una falla en hardware como si la hay en software.

m) Facilidades para consultas no planeadas (Query Languages).- El sistema debe permitir hacer consultas sencillas a la base de datos, sin requerir que esté programada la consulta, esto mediante un lenguaje de consultas. La medida actual son los Sistemas Manejadores de Base de Datos Relacional. Varios manejadores de base de datos orientada a objetos, como Vbase, Eiffel, O2, etc. proporcionan un lenguaje de consulta de alto nivel llamado Object SQL (u OSQL). Utiliza la forma sintáctica tradicional de SQL (SELECT ... FROM ... WHERE ...), pero utiliza nombres de objetos en la cláusula FROM y propiedades en la cláusula SELECT. Utilizando la notación tradicional de 'punto' es posible recuperar información de objetos relacionados. Las diferencias principales entre Object SQL y el SQL relacional son:

1) Los objetos se referencian directamente en vez de utilizar llaves. Las variables se asignan a objetos en la creación o recuperación y pueden utilizarse para referenciar a los objetos en instrucciones siguientes.
2) Se pueden tener operaciones en las clausulas SELECT y WHERE.

Algunos ejemplos de OSQL se dan a continuación indicando primeramente lo que se quiere realizar y abajo las instrucciones correspondientes:

1) Recuperar los nombres de todos los empleados del departamento de Planeación.

```
SELECT e.nombre
FROM e IN Empleado
WHERE e.departmento.nombre = 'Planeación'
```

2) Recupera los nombres y departamentos de los empleados que trabajan en el proyecto 121 junto con el puesto que tienen.

```
SELECT e.nombre, e.departamento.nombre, e.puesto (e,p)
FROM e IN Empleado, p IN Proyecto
WHERE p.numero = 121 AND p IN e.trabaja_en
```

3) Ejemplo de 02SQL, recuperar el nombre de los empleados del cuarto piso que trabajan en un proyecto con un presupuesto mayor a 10000 dolares.

```
SELECT TUPLE (Empleado: e.nombre)
FROM e IN Empleado, p IN Proyecto
WHERE e.piso = 4 AND e IN p.equipo AND p.presupuesto > 10000.
```

V.2 Características opcionales

Las características opcionales de los SMBDOO se refieren a aquellas que son poco comunes en los sistemas comerciales, algunas de ellas son: herencia múltiple, verificación de tipos e inferencia de tipos, distribución, transacciones, versiones.

a) Herencia múltiple.- Consiste en que sea posible generar objetos como resultado de herencia múltiple, existen muchas formas posibles de solución para esto.

b) Verificación de tipos e inferencia de tipos.- Esto consiste en que un programa que haya sido validado a tiempo de compilación no produzca errores de tipo a tiempo de ejecución. La cantidad de inferencia de tipos se tiene abierta a lo que disponga el diseñador. La situación ideal es donde el usuario declara sus tipos base y el sistema infiere los tipos temporales.

c) Distribución. Es una característica ortogonal a los objetos, esto es que el sistema de base de datos puede estar distribuido o no. Esto es los objetos pueden encontrarse fisicamente localizados en diferentes lugares pero formando una sola base de datos.

d) Transacciones.- Esto consiste en manejar un conjunto de operaciones como una unidad, la cual sino se termina satisfactoriamente, se restablece la base de datos a como estaba antes de iniciar la transacción. El problema en los SMBDOO es que las transacciones tienden a ser muy largas, y es un problema a resolver. Una transacción [13] es una unidad de trabajo la cual

corresponde directamente a una actividad de la empresa la cual se modela con la base de datos. Un sistema dado de base de datos puede tener diferentes clases de transacciones desde las actualizaciones interactivas simples hasta programas grandes y complejos que involucran tal vez miles de operaciones de la base de datos. Las propiedades que todas las transacciones tienen en común es que deben preservar la consistencia y correctitud de los datos almacenados. Esto es que las operaciones realizadas por una transacción de actualización deben transformar la base de datos de un estado consistente a otro también consistente. Los estados intermedios en caso de existir, pueden se inconsistentes. Por lo tanto para garantizar la consistencia de la base de datos, se requiere que las transacciones de actualización se procesen completamente o no, pero completas (esto es, las transacciones son atómicas). Una transacción que no se complete debido a cualquier falla, debe rehacer los cambios hasta su estado inicial y cualquier cambio realizado en la base de datos debe deshacerse. Cuando varias transacciones se ejecutan concurrentemente en una base de datos compartida, debe sincronizarse su ejecución. Esto es que el efecto en la base de datos de una transacción debe ser el mismo que se obtendría si solo se tuviera una transacción. El efecto de ejecutar varias transacciones concurrentemente, por lo tanto, debe ser el mismo como si se hubieran ejecutado en secuencia en cualquier orden. Si la secuencia de operaciones de un conjunto de transacciones ejecutándose concurrentemente es tal que se satisface esta condición, se dice que la secuencia es serializable.

e) Versiones.- Esto es para las aplicaciones como las de CAD/CAM y CASE en donde durante la actividad de diseño se requiere del manejo automático de diferentes versiones. Actualmente varios SMBDOO proporcionan este servicio.

V.3 Implantaciones

Existen diversas implementaciones en cuanto a lenguajes de programación y sistemas manejadores de base de datos orientados a objetos.

Smalltalk lo desarrolló Xerox a mediados de los 70s en su Centro de Investigación de Palo Alto, actualmente lo comercializa alrededor del mundo una subsidiaria de Xerox llamada ParcPlace Systems. El manejador de base de datos orientada a objetos GemStone(TM) y su lenguaje asociado OPAL está basado en smalltalk.

La motivación original de los desarrolladores de smalltalk fue producir un ambiente de programación personal avanzado. Aunque smalltalk fue influenciado por Simula, el primero tiene un estilo libre de tipos en el cual el encadenamiento dinámico tiene un papel primordial. No se realiza verificación estática de tipos, de tal forma que el encadenamiento de estructuras se hace a tiempo de ejecución. Por ejemplo, los errores que resultan de invocar una operación en un objeto que no contiene esa operación solo se detectan a tiempo de ejecución. En un lenguaje que maneja tipos de manera estática dichos errores se detectan a tiempo de compilación.

Todo en smalltalk es un objeto, incluyendo a cada clase. Esto es, cada clase se ve como una instancia de una clase de mas alto nivel llamada una metaclase. Así que un programa en smalltalk puede describirse como un árbol de objetos donde la raíz es una superclase interconstruida llamada

Object. La raíz del subarbol que contiene únicamente clases es una metaclase interconstruida llamada Class.

Observando una clase como una instancia de una metaclase mas abstracta hace posible definir métodos de clase los cuales se aplican a la clase en si misma mas que a sus instancias. Dichos métodos de clase pueden usarse por ejemplo, para implementar casos especiales de operaciones interconstruidas, tal como la operación new la cual genera instancias de una clase. Generar clases como parte del ambiente de tiempo de ejecución facilita el desarrollo de herramientas, tal como como rastreadores simbólicos, los cuales requieren accesar a la clase texto a tiempo de ejecución.

En smalltalk, aun una simple estructura de datos como un entero se considera un objeto, con métodos asociados. Por ejemplo la operación:

2 + 4

se interpreta como enviar el mensaje '+' con el argumento '4' al objeto entero '2'. La clase 'Integer' tiene un método cuyo selector es '+' y así es heredado por la instancia '2' la cual realiza la suma y regresa el resultado al enviador del mensaje. La clase Integer es una Subclase de Number la cual también tiene subclases Float y Fraction. La clase Integer también tiene subclase para enteros grandes o pequeños. Cada una de las subclases de Number hereda el método '+' aunque la implementacion del método puede diferir de clase a clase.

GemStone(TM) [16] es un producto que proporciona un ambiente de ejecución de smalltalk basado en la arquitectura cliente-servidor, en ambiente multiusuario. La Interfase (GSI) de smalltalk de GemStone(TM) proporciona herramientas para manejar clases y métodos. La GSI mantiene la replicación sincronizada. GemStone(TM) tiene sus objetos a disposición de interfases tanto en lenguaje SmallTalk, C, C++ en productos como VisualWorks(TM), Visual Age(TM), Visual SmallTalk Enterprise(TM) GemStone(TM) proporciona facilidades para definición, manipulación y consulta de datos en un solo lenguaje. El lenguaje smalltalk ofrece clases (tipos) interconstruidos, operadores y estructuras de control que son comparables a las ofrecidas por C o Pascal y aun mas, sirven para operar sobre objetos complejos.

GemStone(TM) proporciona una biblioteca amplia de clases predefinidas utilizables como bloques prefabricados en la creación de aplicaciones especificas del usuario. La jerarquía de clases incluye tipos estandar como: alfanuméricos incluyendo caracteres, strings; números; fechas, colecciones de clases como arreglos y conjuntos; streams de clases para I/O (entrada/salida). Las clases disponibles también incluyen gráficas, sonidos, componentes de interfases tales como formas. El usuario puede derivar estas clases predefinidas incorporándolas a la jerarquía y dejándolas disponibles para otros usuarios o aplicaciones.

GemStone(TM) soporta cientos de usuarios concurrentemente y almacenamiento de hasta 10 terabytes. Cuenta con arquitectura cliente-servidor y soporte de multiplataformas lo que le permite operar en una red heterogénea de estaciones de trabajo, mainframes y servidores de despliegue. Puede ejecutar sus métodos en la plataforma donde más se le requiere para minimizar el trafico en la red y así mejorar el performance (rendimiento). Automáticamente pasa a memoria cache la

información usada frecuentemente para reducir el trafico en disco. Se pueden construir índices para acelerar las búsqùedas de las colecciones de uso frecuente, se pueden agrupar los objetos en algún extent (deposito) para mejorar el performance de lectura/escritura en disco. Permite hacer respaldos y recuperaciones en cualquier momento, aun cuando se están procesando transacciones (En-Linea). Se tienen recuperadores de datos después de una falla en disco o en el sistema, el restablecimiento se puede hacer utilizando algún respaldo y aplicándole respaldos de actualización, hasta que se apliquen las transacciones mas actuales. Proporciona mecanismos de seguridad a diferentes niveles y selectivo para diferentes objetos o usuarios.

La arquitectura de GemStone(TM) consiste de un deposito donde se almacenan los objetos y un conjunto de procesos que cooperan entre si. Los dos procesos clave son Gem y Stone. Los procesos Gem proporcionan servicios de acceso a disco, proceso del lenguaje smalltalk, almacenamiento de objetos, la recuperación de objetos para programas de aplicación se pueden desarrollar con cualquier interfase. El proceso Gem proporciona una vista consistente de los objetos al usuario y maneja la sesión de GemStone, manteniendo una relación de los objetos que se han utilizado y los requerimientos de memoria. El proceso Stone es el coordinador de los recursos del deposito de objetos. Sincroniza y asegura la consistencia así como la completitud de las transacciones. Los recursos que ásigna Stone son identificadores de objetos, paginas de memoria y seguridades de objetos.

Los requerimientos típicos de GemStone son:

a) Para sistemas MS-DOS 6.x/Windows 3.x, OS/2, Power Machintosh/Machintosh se requieren 16 Mb de memoria y hasta 24 o 32 Mb para algunas configuraciones, se recomienda 32 Mb. En sistemas UNIX típicamente se requieren 32Mb de memoria y 3 Mb adicionales por sesión de GemStone(TM)

Los lenguajes de objetos se utilizan para crear aplicaciones sofisticadas de negocios [16] como
a) Sistemas de información financiera que almacenan relaciones complejas,
b) Sistemas de control de procesos que interpretan datos cambiantes en tiempo real,
c) Sistemas de manejo de documentos que almacenan documentos digitalizados,
d) Sistemas basados en conocimiento que usan reglas de inferencia,
e) Sistemas de publicidad, recuperadores de documentos "En-Linea",
f) Sistemas de información geográfica,
g) Sistemas de manejo de información médica,
h) Aplicaciones multimedia que usan imágenes, voz y video,
i) Aplicaciones de recursos humanos incluyendo la nómina.

V.4 Bases de conocimiento orientadas a objetos

Conforme los modelos de datos han evolucionado durante las décadas pasadas ha existido un movimiento lento hacia el desarrollo de base de datos 'inteligentes' Dichas bases de datos manejan información en una forma natural y amigable, haciendo la información fácil de almacenar, accesar y usar [13]. Esto implica el uso de aplicaciones como sistemas basados en conocimiento. Hay un gran número de aplicaciones en las bases de datos inteligentes donde las bases de datos orientadas a objetos pueden ser de utilidad. Una de estas áreas es la de representación y manipulación del conocimiento..

En cuanto a los esquemas de representación del conocimiento se tienen varios de ellos:

a) Representación del conocimiento utilizando lógica. En este esquema el conocimiento se representa mediante expresiones formales de la lógica. Para resolver problemas se utilizan las reglas de inferencia. La forma más utilizada es la representación utilizando cálculo de predicados de primer orden. El lenguaje de programación Prolog es un ejemplo de esto.

b) Representación del conocimiento mediante redes semánticas: En este esquema el conocimiento se almacena con una gráfica dirigida en donde los nodos representan objetos o conceptos del dominio del problema y los arcos representan relaciones entre ellos. Son una de las formas más utilizadas para representación del conocimiento.

c) Representación del conocimiento utilizando procedimientos: Esta es la forma típica en que el conocimiento se representa en forma de conjunto de instrucciones o procedimientos para resolver problemas específicos. Los sistemas basados en este esquema tienden a crear programas grandes y difíciles de entender.

d) Esquemas estructurados de representación del conocimiento [15] y [13]: Consisten en ampliar las redes semánticas para permitir que en los nodos se manejen estructuras de datos complejas, los valores que puede contener pueden ser datos, apuntadores a otras estructuras complejas o posiblemente procedimientos para ejecutar alguna tarea específica.

Las representaciones basadas en lógica son útiles para representar una gran variedad de tipos de conocimiento. Proporcionan mecanismos poderosos de razonamiento, sin embargo, las estructuras que permiten representar son muy limitadas, y tienen problemas para manejar el conocimiento cuando se adiciona o elimina alguno.

En el caso de la información semántica es esencial para un amplio espectro de aplicaciones incluyendo información taxonómica, tal como las relaciones entre los miembros de una familia, la clasificación de especies de plantas y animales, la descripción de objetos complejos en términos de sus componentes o la manipulación de diferentes conocimientos.

Flavors, Clos, Loops son sistemas orientados a objetos que se han utilizado en implantaciones de problemas de IA

El Ambiente de Ingeniería de Conocimiento (Knowledge Engineering Environment(TM), KEE) desarrollado por Intellicorp, Inc., es un sistema híbrido que incorpora una gran gama de técnicas de solución de problemas de IA incluyendo programación funcional, marcos, reglas y explicación

de razonamientos. Las facilidades orientadas a objetos en el sistema están en el uso de paso de mensajes al estilo SmallTalk aunque el sistema esta construido sobre Lisp. Incluye una interfase gráfica sofisticada que ayuda en el proceso de desarrollo de la base de conocimientos.

El bloque base de KEE es la Unidad, la cual es comparable al concepto de clase. Una unidad puede heredar propiedades y métodos.

Similar a los marcos, una unidad consiste de ranuras que almacenan sus atributos. Las ranuras pueden contener datos textuales o numéricos aunque también pueden contener estructuras de datos complejas como tablas o redes. También pueden almacenar reglas y mas importantemente métodos. Cada ranura tiene sus propios atributos llamados facets. Un uso importante de los facets es que proporcionan un mecanismo para establecer las reglas de herencia que gobiernan la manera en la cual una propiedad o un método se deriva de sus ancestros.

Los métodos en las ranuras se escriben en un lenguaje de programación y se inician al recibir un mensaje. La unidades pueden emitir y recibir mensajes. Los mensajes pueden enviarse a través de la interfase dirigida por ratón o desde un programa. Los métodos pueden tener cualquier numero de argumentos.

En [15] se tienen objetos que pueden contener cualquier tipo de dato. El conocimiento se registra en una Red Semántica Ampliada (RSA). Cada objeto esta relacionado semánticamente con otros que establecen su significado final por contexto. La adquisición de conocimiento se hace mediante oraciones declarativas en lenguaje escrito del usuario y la consulta de la RSA se hace mediate oraciones interrogativas o imperativas también en el lenguaje escrito del usuario. La obtención de la RSA se hace mediante un proceso de agregación en donde se generan relaciones de subordinación, supraordinacion y coordinación. Se tiene herencia en la dirección que establezca la semántica de los objetos. Uno de sus módulos interconstruidos permite proponer reglas con base a la estructura que tengan los objetos.

CONCLUSIONES

Los resultados obtenidos de la revisión y modelación de la base de datos para la automatización de la colección de microorganismos CDBB-500, concuerdan con la expectativa inicial acerca de la complejidad del modelo de datos. Se encontró una gran cantidad de atributos necesarios para el adecuado manejo y preservación en condiciones óptimas de los microorganismos.

La metodología aplicada al caso consistió de las etapas: a) recopilación de datos y detección de requerimientos, b) análisis de datos y diseño conceptual de la base de datos, c) depuración del modelo, d) implantación de objetos tipo.

Con la primera etapa se conoció lo que es una colección de microorganismos, sus actividades y requerimientos de información.

La segunda etapa permitió obtener el modelo de la base de datos, primeramente el relacional y posteriormente reestructurado mediante el enfoque orientado a objetos. En la depuración del modelo fueron útiles y se incluyeron en el mismo los comentarios y sugerencias de los alumnos de maestría de ciencias de la computación del CINVESTAV-IPN.

La implantación del modelo de base de datos se hizo en dos partes. En la primera se implantaron objetos tipo en VisualWorks(TM) como base para la interacción con el usuario y GemStone(TM) En la segunda, motivo de otro trabajo, en el manejador de base de datos ACCESS(TM)

Del trabajo realizado surgen varios puntos de interés que pueden tomarse en consideración para constituir el sistema de información de la colección CDBB-500:

a) Realizar estudios del área desde el punto de vista estadístico, para profundizar en cuanto a la obtención, manejo y utilización de este tipo de datos.

b) Considerar la aplicación de las diferentes especialidades de las ciencias de la computación e informática, como el caso de inteligencia artificial, para constituir la base de datos en una base de conocimientos. Procesamiento digital de imágenes para incluir los aspectos necesarios en el manejo, almacenamiento y recuperación de las mismas. Implantar módulos que faciliten y mejoren la interacción con el usuario como lenguaje natural y lenguajes visuales.

c) Toma en cuenta y en lo posible incluir aportaciones de otras áreas como la electrónica lo que automatizaría y mejoraría las tareas repetitivas involucradas en la preservación óptima del acervo de la colección.

REFERENCIAS BIBLIOGRAFICAS

[1] MARTINEZ Cruz, Jovita, et al., SISTEMA DE INFORMACION PICTOGRAFICO DE LA COLECCION DE MICROORGANISMOS DEL CINVESTAV-IPN, CINVESTAV-IPN, México, D.F., 28 abril 1995

[2] MARTINEZ Cruz, Jovita, et al., INFORME PROYECTO P-153 COLECCION DE CULTIVOS MICROBIANOS DEL CINVESTAV-IPN: Fase I Base de Datos, septiembre-diciembre 1984, CINVESTAV-IPN, México, D.F., 6 enero 1995

[3] YOURDON, Edward, ANALISIS Y DISEÑO ORIENTADO A OBJETOS (OOA/OOD), 7-8 septiembre 1994, México, D.F., Technology Training S. de R.L. de C.V.

[4] AZUARA Monter, Iván, et al., TECNOLOGIA Y MANEJO DE INFORMACION GEOGRAFICA EN BIOCONSERVACION en Ciencia y Desarrollo, septiembre-octubre 1994, vol. XX, número 118, pp 58-85, México, D.F.

[5] _____, INSTRUCTIVO PARA LA CONFORMACION DE BASE DE DATOS COMPATIBLES CON EL SISTEMA NACIONAL DE INFORMACION SOBRE BIODIVERSIDAD, Comisión Nacional para el Conocimiento y uso de la Biodiversidad, México, D.F.

[6] CHAPA Vergara, Sergio V., et al., DISEÑO CONCEPTUAL Y LOGICO DE UNA BASE DE DATOS GEOGRAFICA APLICADA A LA EXPLORACION PETROLERA, CINVESTAV-IPN, México, D.F., 10 marzo 1994

[7] QUINTANILLA, Gloria, EL IMPACTO EN LA INGENIERIA DE SOFTWARE DE LA PROGRAMACION ORIENTADA A OBJETOS en SOLUCIONES AVANZADAS, México, D.F., abril-mayo 1993

[8] OKTABA, Hanna, PROGRAMACION ORIENTADA A OBJETOS ¿MODA O REALIDAD? en SOLUCIONES AVANZADAS, México, D.F , abril-mayo 1993

[9] MARTINEZ Cruz, Jovita, COLECCIONES DE MICROORGANISMOS en Avance y Perspectiva, vol. 11, noviembre-diciembre 1992, CINVESTAV-IPN, México, D.F.

[10] VASKEVITCH, David, DOS PASOS HACIA ADELANTE. UN PASO HACIA ATRAS en PC/TIPS BYTE, México, D.F., mayo 1992

[11] ALTAMIRANO Robles, et al, SISTEMAS DE BASE DE DATOS PICTOGRAFICOS (IMAGE DATABASE SYSTEMS), CINVESTAV-IPN, México, D.F., marzo 1992

[12] HERNANDEZ Stefanoni, Raúl, et al., NORMALIZACION DE BASE DE DATOS EN C, CINVESTAV-IPN, México, D.F., octubre 1990

[13] HUGHES Jonh G., OBJECT-ORIENTED DATABASES, Prentice Hall International, USA, 1991

[14] BANCILHON Francois, DELOBEL Claude, KANELLAKIS Paris, BUILDING AN OBJECT-ORIENTED DATABASE SYSTEM: THE STORY OF O2, Morgan Kaufmann Publishers Inc . San Mateo California, USA, 1992

[15] OLIVARES Ceja, Jesús M., SISTEMA EVOLUTIVO PARA REPRESENTACION DEL CONOCIMIENTO, IPN-UPIICSA, México, 1991

[16] Servio, GEMSTONE en Internet

Los abajo firmantes, integrantes de jurado para el examen de grado que sustentará el **Lic. Jesús Manuel Olivares Ceja**, declaramos que hemos revisado la tesis titulada:

CONSTRUCCION DE UNA BASE DE DATOS PICTOGRAFICOS CON APLICACION A LA COLECCION DE MICROORGANISMOS CDBB-500 DEL CINVESTAV-IPN

y consideramos que cumple con los requisitos para obtener el grado de Maestro en Ciencias, con especialidad en Ingeniería Eléctrica.

Atentamente

Dr. Sergio Victor Chapa Vergara

Dr. José Angel Lodegario Ortega Herrera

Dr. Manuel González Hernández