26
21.

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN SISTEMA DE DISTRIBUCION Y ACTUALIZACION DE SOFTWARE PARA LAS INSTITUCIONES FINANCIERAS QUE REALIZAN OPERACIONES CON EL BANCO DE MEXICO

T E S I S

QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACION
P R E S E N T A N :

ALEJANDRO JESUS COVARRUBIAS GARCIA
A L F R E D O V E G A D O M I N G U E Z
JOSE ARMANDO PEREZ GONZALEZ
MARIA DE LAS NIEVES CONTRERAS DAVILA

DIRECTOR DE TESIS: M. EN I. LAURO SANTIAGO CRUZ

MEXICO, D. F. 1997.

Dedico este trabajo, con mucho cariño a

Dios, por permitirme cumplir una meta más
en mi vida

Mi Padre Jorge Covarrubias a quien le agradezco
su dedicación, su fortaleza y el cuidado que me dió,
con todo mi amor para ti

Mi Madre Dolores García por ser un ejemplo de
perseverancia, esmero y amor, el cual agradeceré
toda la vida, te quiero mucho

Mi familia Hermanos, cuñados, sobrinos por ser un ejemplo de
unión, trabajo en equipo y de respeto a nuestros
principios, gracias

Mis amigos en general, gracias por permitirme
ser parte de sus vidas

Nieves, Alfredo y Armando por ser
compañeros y amigos, gracias

Ing. Lauro Santiago y todo el grupo del PAT por el apoyo
en la realización de este trabajo

Mis Maestros en general, a **La Facultad de Ingeniería**,
y todos los que la integran, gracias

Todos mil gracias

Alejandro Jesús

DEDICATORIA

Este trabajo y lo que representa quiero dedicarlos a mis padres Luis Vegas y Victoria Dominguez por haberme enseñado el valor del trabajo y acompañarme a recorrer el camino; a mi hermano Ricardo y de manera muy especial para Judith.

AGRADECIMIENTOS

El trabajo de tesis es siempre el resultado del esfuerzo de muchas personas e instituciones, por esto, quisiera agradecer a la UNAM y a la Facultad de Ingenieria por haberme dado una educación profesional. De la misma forma deseo expresar mi agradecimiento al M en I. Lauro Santiago por el tiempo dedicado durante la realización de este trabajo.

A mis amigos de la Facultad de Ingenieria y Coordinacion de Automatización del Instituto de Ingenieria.

A los buenos amigos del Banco de México por las facilidades otorgadas durante la realización de este trabajo.

Finalmente quisiera agradecer a mis padres por el apoyo y la paciencia que siempre tuvieron conmigo. Lamentablemente él no pudo ver terminado este trabajo...

ALFREDO

A mis padres, Javier y Josefina , por enseñarme a trabajar y demostrarme como se triunfa en esta vida.

A mis hermanos, Francisco, Dolores, Nely, Claudia y Mariana , por que siempre puedo contar con ellos.

A Jorge, al cual considero otro de mis hermanos.

A Dafné por su gran amor y por darme un motivo de lucha en esta vida.

A mis sobrinos José e Israel.

Y a todas las personas que han influido en mi vida, que por falta de espacio no menciono.

GRACIAS

ARMANDO

Gracias a Dios, por dejarme andar y disfrutar de la vida.

A mis padres, porque por ellos y para ellos existo, y siempre han sido el cimiento de mis metas. Gracias papis.

A mis hermanos Lily y Arturo, por su cariño y paciencia. Los quiero mucho.

A mi abuelita Nieves, por la ternura que me ha brindado durante toda la vida.

A mis tíos y primos, por su apoyo incondicional.

A Carlos, por su amor y comprensión. Te amo.

A mis amigos, Alfredo, Armando, Arturo, Biliana, Cristina, Gerardo, Iliana, Juan, Manuel, Marcos, Maru, Moisés, Nora, Olaiz, Raúl, por estar siempre conmigo.

A Alejandro, Alfredo y Armando por su entusiasmo y dedicación para lograr juntos una meta más.

A mi Universidad y profesores, por grabar en mi ser sus enseñanzas y hacer de mi un profesionista con la misión de representarla con el corazón en la mano.

Al Banco de México y a la Dirección de Sistemas, por ser parte fundamental en mi desarrollo profesional.

Nieves

ÍNDICE

INTRODUCCIÓN

En la actualidad, la computación se ha convertido en una herramienta indispensable, puesto que hacia cualquier actividad que dirijamos nuestra atención, sea en el ambiente administrativo, industrial, científico o tecnológico, está presente de alguna manera. Definitivamente sin su apoyo sería extremadamente difícil lograr algún avance.

La computación ha hecho posible que en los últimos cuarenta años, la humanidad haya alcanzado logros realmente sorprendentes. Si comparamos este corto período con respecto a cualquier otra época, podemos afirmar que en ninguna se ha progresado tan significativamente como en ésta.

Con la constante evolución de la computación y la electrónica se han desarrollado nuevos dispositivos y metodologías en el desarrollo de software, tales como: fax, CD-ROM, scanners, multimedia, realidad virtual, programación orientada a objetos. etc., y en cuanto a comunicaciones se han obtenido redes de alto desempeño, comunicación vía satélite, transmisión de voz, datos e imágenes sin importar la distancia.

Si bien, en un principio todos estos adelantos tecnológicos se han caracterizado por un elevado costo, hoy en día éste se ha reducido considerablemente y de igual manera los dispositivos se han vuelto significativamente más eficientes de lo que eran hace algunos años.

Por otra parte, en nuestro país, el desarrollo futuro presenta múltiples retos y oportunidades para las empresas e instituciones, las cuales han ido incorporando los avances que en materia tecnológica están a su disposición. El crecimiento con estabilidad y la globalización de nuestra economía, está generando mayor demanda de servicios financieros, que requieren elevar los niveles de productividad y calidad de servicio para competir en una economía abierta. Como resultado, las instituciones financieras se están transformando para orientarse mejor hacia el mercado y adaptarse a las diferentes tendencias del sector financiero mexicano.

La banca comercial busca brindar el mejor servicio, en función de los requerimientos de sus clientes. Para lograr esto se apoyan en los avances tecnológicos que el mercado les brinda como son: el banco por teléfono, las terminales punto de venta y la transferencia de fondos electrónicos entre otros.

Dentro del ámbito bancario, el Banco de Mexico, órgano regulador de la banca comercial, evoluciona constantemente e incorpora la tecnologia necesaria para ofrecer a sus clientes servicios más confiables y eficaces, además desarrolla sistemas de software que permite a las instituciones financieras interactuar con él y compartir información en línea sobre una plataforma uniforme.

Estos sistemas de software están construidos bajo la arquitectura Cliente/Servidor. La parte del servidor se encuentra localizada en el centro de cómputo del Banco de México y la parte del cliente en las terminales de las instituciones.

La parte del cliente se modifica constantemente por aspectos tales como seguridad, confiabilidad, mejor manejo de errores, etc., y esto da origen a distintas versiones de los sistemas utilizados.

El principal problema es proporcionar a las instituciones financieras las nuevas versiones de sistemas en forma oportuna.

Actualmente se necesita disponer de recursos humanos y materiales, tanto por parte del Banco de México como de las instituciones afines, para poder distribuir e instalar las aplicaciones. Esta distribución se hace por medio de discos flexibles, lo cual es muy poco eficiente, ya que retarda la actualización y provoca un alto costo de recursos.

En este trabajo se analizará e implementará un sistema para optimizar la distribución y actualización de software en el Banco de México, estableciendo esquemas de seguridad que permitan la entrega confiable del software a los clientes indicados. Además se establecerán mecanismos para asegurar la instalación correcta y de forma automática de las diferentes versiones del software. Por lo tanto, estableceremos como objetivo principal:

"Proporcionar a las instituciones financieras que realizan operaciones con el Banco de México, un sistema de distribución y actualización de software que permita la transferencia e instalación de aplicaciones en sus terminales de forma práctica y oportuna, minimizando el uso de recursos humanos y materiales".

Descrito el objetivo principal, a continuación se presenta una breve descripción de los capítulos que comprende este trabajo.

Capítulo 1. Se explicarán las bases teóricas necesarias para el desarrollo del sistema.

Capítulo 2. Se analizarán las necesidades, capacidades y restricciones, para establecer los objetivos y delimitar el entorno de trabajo, planteándose estrategias de solución del problema, seleccionando la mejor opción y planeando su desarrollo.

Capítulo 3. Se efectuará un análisis detallado a través de diagramas de flujo de datos, estableciéndose los orígenes y destinos de la información, así como los procesos que la

modifican. Se aplicarán distintas técnicas y principios con el propósito de definir un sistema con los suficientes detalles, que permitan su realización física.

Capítulo 4. Se realizará un estudio para la selección del software y hardware. Se analizarán los módulos a incluir en el sistema. Se realizará el diseño de pantallas y se programará la implementación.

Capítulo 5. Se efectuará la implementación del sistema basándonos en un plan de pruebas y entrenamiento previamente establecidos. Se tomará en cuenta el mantenimiento necesario para añadir mejoras y llevar a cabo optimizaciones, además de mostrar un ejemplo de la ejecución del sistema.

Capítulo 6. Se presentarán los resultados y conclusiones analizando el cumplimiento total del objetivo planteado. Así como las características generales del sistema en el sentido de costo y funcionalidad, dentro del entorno para el cual fue desarrollado, determinándose la calidad real del producto.

Finalmente se presentan la bibliografía consultada y los apéndices.

Capítulo
UNO

ANTECEDENTES

Para el desarrollo del sistema se requiere del conocimiento de algunos aspectos relevantes.

En este capítulo se explicarán las bases teóricas necesarias para la implementación del sistema que cubrirá las necesidades de distribución y actualización de software del Banco de México.

1.1 Esquema Cliente/Servidor

Cada vez es más importante el poder hacer que la información esté disponible en donde se necesita. Para lograr esto, tanto la información como los sistemas para procesarla deben ser distribuidos a una extensa audiencia.

Las tecnologías computacionales modernas buscan responder a las necesidades de las empresas de los 90's y para ello plantean nuevas formas de hacer las cosas. Entre ellas una de las más importantes es el llamado esquema Cliente/Servidor, cuyos principios más importantes se definen a continuación.

El esquema Cliente/Servidor "es un modelo de computación en el que el procesamiento requerido para ejecutar una aplicación o conjunto de aplicaciones relacionadas se divide entre dos ó más procesos que cooperan entre sí".[1] Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el (los) proceso(s) cliente(s) sólo se ocupa de la interacción con el usuario (aunque esto puede variar).

Los principales componentes del esquema Cliente/Servidor son entonces los Clientes, los Servidores y la infraestructura de comunicaciones.

[1] Datapro Client/Server Analyst, client- server computing: emerging trends, solutions and strategies, 1994. http://hidra.uniandes.edu.co/articulos/clser.html

Los clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar una petición, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Los servidores proporcionan un servicio al cliente y devuelven los resultados. Los servicios proporcionados pueden entrar en diferentes categorías como son: funciones del sistema operativo, recursos compartidos y aplicaciones.

Dentro de las principales funciones del sistema operativo se encuentran la de autenticación y asignación de permisos sobre los dispositivos del sistema.

Para la función de compartir recursos encontramos los ejemplos de servidor de archivos y servidor de impresiones.

En cuanto a las aplicaciones, se cuenta con los servidores de correo electrónico, bases de datos, procesadores gráficos y numéricos, entre otros de uso especializado.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas Cliente/Servidor actuales se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

La arquitectura Cliente/Servidor tiene varias ventajas sobre las arquitecturas tradicionales, entre ellas se pueden mencionar las siguientes:

- El tamaño y la complejidad de la aplicación son significativamente reducidas debido a que los servicios comunes están en una sola localidad, el servidor. Esto simplifica las aplicaciones cliente, reduce la duplicación de código, y facilita el mantenimiento de la aplicación.

- Facilita la comunicación entre aplicaciones variadas. Las aplicaciones cliente que utilizan distintos protocolos de comunicación no se pueden comunicar directamente, pero pueden hacerlo a través de un servidor que "habla" ambos protocolos, conocido como *gateway*.

- Esta arquitectura permite que las aplicaciones se puedan desarrollar con distintos componentes. Estos componentes pueden ser modificados o remplazados sin afectar otras partes de la aplicación, favoreciendo la adaptación a cambios en la tecnología, pues facilita la migración de las aplicaciones a otras plataformas y, al aislar claramente las diferentes funciones de una aplicación hace más fácil incorporar nuevas tecnologías en ésta.

- Los procesos se hacen más confiables, ya que los recursos de cómputo pueden distribuirse en varios servidores y los clientes pueden configurarse de modo tal que cualquier aplicación o archivo de datos pueda ser accesado desde cualquier parte. Por lo tanto el aislamiento a fallas que se puede incluir en este ambiente, reduce el tiempo perdido.

1.2 Bases de Datos Relacionales

Un sistema de manejo de base de datos, consiste en un conjunto de datos relacionados entre si y un grupo de programas para tener acceso a esos datos. El conjunto de datos se conoce comúnmente como base de datos. El objetivo primordial de un manejador de base de datos es crear un ambiente en que sea posible guardar y recuperar información de la base de datos en forma conveniente y eficiente.

Los sistemas de base de datos se diseñan para manejar grandes cantidades de información. El manejo de los datos incluye tanto la definición de las estructuras para el almacenamiento de la información, así como los mecanismos para el manejo de la misma. Además, el sistema de base de datos debe cuidar la seguridad de la información almacenada en está, tanto contra las caídas del sistema como contra los intentos de acceso no autorizados. Si los datos van a ser compartidos por varios usuarios, el sistema debe evitar la posibilidad de obtener resultados anómalos.

Existen tres modelos básicos de bases de datos, los cuales se describen brevemente:

Modelo de red. Los datos se representan por medio de conjuntos de registros y las relaciones entre los datos se representan con ligas, que pueden considerarse como apuntadores. Los registros de la base de datos se organizan en forma de conjuntos de gráficas arbitrarias.

Modelo jerárquico. El modelo jerárquico es similar al modelo de red en cuanto a la representación por medio de registros y ligas, pero difiere del de red, en que los registros están organizados como conjuntos de árboles.

Moldeo Relacional. En el modelo Relacional, los datos y las relaciones entre los datos se representan por medio de una serie de tablas, cada una de las cuales tienen varias columnas con nombres únicos.

La mayoría de las bases de datos actuales (Informix, Sybase, Oracle, etc.), están basadas en el Modelo Relacional, por lo cual nos enfocaremos al estudio del mismo.

1.2.1 Modelo Relacional

Se podría establecer que el modelo relacional es el resultado de los anteriores modelos de manejadores de bases de datos (de red, jerarquicos, etc.)

El modelo de datos entidad-relación está basado en una percepción de un mundo real que consta de una serie de objetos reales llamados entidades, y de las relaciones entre ellos.

Una entidad es un objeto que existe y puede distinguirse de otros objetos. La distinción se logra relacionando cada objeto con una serie de atributos que lo describen. Una relación es una asociación entre varias entidades. Los conjuntos de todas las entidades y de todas las relaciones del mismo tipo se denominan conjunto de entidades y conjunto de relaciones, respectivamente.

Además de las entidades y las relaciones, el Modelo Relacional cuenta con los siguientes elementos:

- **Diagrama Entidad-Relación:** Diagrama mediante el cual la estructura lógica general de una base de datos puede expresarse en forma gráfica. Para su representación se usan los siguientes componentes:

  Rectángulo, que representa a las entidades.

  Heptágono, que representa atributos.

  Elipse, que representa las relaciones entre conjuntos de entidades.

  Líneas, que conectan los atributos a los conjuntos de entidades y los conjuntos de entidades a las relaciones.

- **Cardinalidad de Mapeo:** El esquema de entidad-relación representa las cardinalidades de mapeo que expresan el número de entidades con las que pude asociarse otra entidad mediante una relación.

  Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de mapeo debe ser una de las siguientes:

  Una a una: Una entidad en A está asociada únicamente con una entidad en B, y una entidad en B está asociada sólo con una entidad en A.

  Una a muchas: Una entidad en A está relacionada con cualquier número en entidades en B, pero una entidad en B puede asociarse únicamente con una entidad en A.

  Muchas a una: Una entidad en A está vinculada únicamente con una entidad en B, pero una entidad en B está relacionada con cualquier número de entidades en A

  Muchas a muchas: Una entidad en A está asociada con cualquier número de entidades en B, y una entidad en B está vinculada con cualquier número de entidades en A.

- **Llave primaria:** Es una columna o combinación de columnas dentro de una tabla cuyo(s) valor(es) identifica(n) unívocamente a cada fila de la tabla. Una tabla tiene una única llave primaria.

- **Llave foránea:** Es una columna o combinación de columnas en una tabla, y puede(n) ser llave primaria para alguna otra tabla. Una tabla puede contener más de una llave foránea.

- **Índices:** Un índice es una estructura que proporciona un acceso rápido a las filas de una tabla en base a los valores de una o más columnas. Existen los índices sobre la llave primaria, las cuales se denominan *clustered*, y los índices sobre los demás campos de la tabla, que se denominan *nonclustered*.

- **Vista:** Una vista es una "tabla virtual" cuyo contenido está definidos por una consulta. Para el usuario de la base de datos, la vista aparece igual que una tabla real, una vista no existe en la base de datos como conjunto almacenado de valores. En su lugar, las filas y columnas de datos visibles a través de la vista son los resultados producidos por la consulta que define la vista.

1.2.2 Álgebra Relacional

Un lenguaje de consulta sirve para que el usuario solicite información de la base de datos, suelen clasificarse en lenguajes de procedimientos o sin procedimientos. En un lenguaje de procedimientos el usuario le ordena al sistema que realice una serie de operaciones con la base de datos para obtener el resultado deseado. En un lenguaje sin procedimientos, el usuario describe la información que desea sin indicar un procedimiento específico para obtenerla.

El álgebra relacional es un lenguaje de consulta de procedimientos. Existen cinco operaciones fundamentales en el álgebra relacional, que son: elegir, proyectar, producto-cartesiano, union y diferencia-conjuntos. Todas ellas producen como resultado una nueva relación.

Las operaciones elegir y proyectar se denominan operaciones unarias, ya que actúan sobre una sola relación. Las otras tres relaciones operan sobre parejas de relaciones, por lo que se llaman operaciones binarias.

- **Operación Elegir:** La Operación Elegir opta por tuplas que satisfagan cierto predicado. Se utiliza la letra griega sigma minúscula (σ) para señalar la selección. El predicado aparece como subíndice de sigma. La relación que constituye el argumento se da entre paréntesis después de la sigma.

$\sigma p(E1)$, donde p es un predicado con atributos de E1

- **Operación Proyección:** La Proyección se señala con la letra griega pi mayúscula(Π). Como subíndice de pi se pone una lista de todos los atributos que se desea que aparezcan en el resultado. La relación argumento se escribe después de Π entre paréntesis.

$\Pi s(E1)$, donde s es una lista que consiste en algunos de los atributos que aparecen en E1

- **Producto Cartesiano:** Esta operación es binaria, se representa por una cruz (X). Se utiliza notación infija para las operaciones binarias. En general, si se tienen las relaciones r1(R1) y r2(R2), entonces r1Xr2 es una relación cuyo esquema es la concatenación de R1 y R2. La relación R contiene todas las tuplas t para las cuales existe una t1 en r1, y t2 en r2, para las que t[R1]=t2[R1] y t[R2]= t2[R2].

- **Unión:** Se representa con el símbolo U, y funciona igual que en teoría de conjuntos. Para que una operación r unión con s sea legal, es necesario que se cumplan dos condiciones:

1. - Las relaciones r y s deben tener el mismo número de atributos.
2. - Los dominios de los atributos i-ésimo de r y de s deben ser los mismos.

- **Diferencia de Conjuntos:** El operador de Diferencia de Conjuntos, representado por el símbolo -, permite encontrar las tuplas que estén en una relación, pero no en otra. La expresión r - s resulta en una relación que contiene las tuplas que estén en r pero no en s.

- **Operadores Adicionales**

  Intersección de Conjuntos(∩): El operador Intersección nos permite encontrar las tuplas que estén en ambas relaciones. La expresión r ∩ s resulta en una relación que contiene las tuplas que están en r y en s.

  No se incluyó la Intersección de conjuntos entre las operaciones fundamentales por que se puede simular la operación intersección con operaciones de diferencia de conjuntos (r ∩ s = r - (r - s)).

  Producto Theta(Θ): El Producto Theta es una operación binaria que permite combinar la elección y el producto cartesiano en una sola operación; se indica con [X]Θ, donde [X] es el símbolo de "unión" y el subíndice Θ (la letra griega theta) se sustituye con el predicado de elección. El operador producto theta forma el producto cartesiano de sus dos argumentos y después lleva a cabo una elección mediante el predicado Θ.

  En general, Θ puede ser un predicado arbitrario. Dadas dos relaciones, r y s, y un predicado Θ

  $$r\ [X]_{\Theta}\ s = \text{beta}_{\Theta}\ (r \times s)$$

  Producto Natural: Piénsese en dos relaciones r(R) y s(S). El Producto Natural e r y s, expresado por r [X] s, es una relación del esquema R U S. Es la proyección sobre R U S de un producto theta donde el predicado requiere que r.A = s.A para cada atributo A en R U S. Formalmente,

  $$r \times s = \prod_{R \cup S}^{([X]\ s)} \quad r.A1 = s.A1 \wedge, \ldots, \wedge\ r.An = s.An$$

  Donde R U S = {A1,...,An}

  Operación División (÷): El operador División es apropiada para consultas que incluyen la frase "para todos" denotada por ∀.

  $$\forall\ t(Q(t))$$

  Significa " Q se cumple para todas las tuplas t.

1.2.3 Leyes de Codd

El Dr. Codd sentó las bases de lo que posteriormente sería el modelo Relacional, a través de su artículo " Relational Model of Data for Large Shared Data Banks".[2] En éste se especifican algunas características técnicas que deberían cubrir un sistema de Base de Datos que pretendiera ser Relacional, posteriormente publicó un resumen de puntos técnicos conocidos en la actualidad como las 12 reglas de Codd, mismas que presentamos a continuación.

Regla 1 (Regla de la Información)

Toda la información en un Sistema Relacional de Base de Datos se representa explícitamente y de una sola manera en el nivel lógico a través de tablas relacionales. En terminología relacional una tabla recibe el nombre de atributos y representan las características de una entidad. El renglón recibe el nombre de tupla.

Regla 2 (Acceso Garantizado)

Un sistema de Base de Datos Relacional debe garantizar el acceso a todos y cada uno de los datos almacenados, a través de la combinación de 3 valores que son: Nombre de la Columna, Nombre de la Tabla y Valor del la Llave Primaria.

Regla 3 (Tratamiento Sistemático de Información Faltante)

Un sistema completamente relacional debe prever indicadores (diferentes de cero como valor o una variable alfanumérica rellena de caracteres en blanco) para representar de una manera sistemática, en el nivel lógico, el hecho de que cierta información dentro de la base de datos se desconozca. Además de la representación lógica, el sistema relacional debe prever funciones de manipulación para dichos indicadores y estos a su vez, deben ser independientes del tipo de información faltante.

Regla 4 (Catálogo Relacional dinámico en Línea)

La descripción de la Base se representa, en el nivel lógico, como cualquier otra información, de tal manera que los usuarios autorizados para ello puedan llevar acabo operaciones sobre esta información en la misma forma en que lo haría con la información normal. Esto quiere decir que la definición de la Base de Datos se almacena en un Catálogo Dinámico, dentro del cual podemos accesar dicha información, ejecutando los mismos comandos SQL que utilizaríamos para explotar información almacenada en la Base de Datos, de tal manera que el Administrador de la Base de Datos utiliza el mismo lenguaje para administrar ésta que el programador para explotar información.

Regla 5 (Sub_lenguaje de Datos)

Un sistema relacional debe soportar por lo menos, un lenguaje cuyos comandos puedan expresarse, a través de una sintaxis formalmente definida, como un conjunto de caracteres alfanuméricos y cuya utilización nos permite llevar a cabo por lo menos las operaciones básicas.

[2] Codd, E.F. 'A Relational Model for Large Shared Data Banks', Communications of the ACM, Volumen 13, Number 6,June 1970.

Regla 6 (Actualización de Vistas)

El sistema relacional debe incluir un algoritmo, para determinar si en la vista en cuestión se puede llevar a cabo cierto tipo de acciones como: insertar una tupla, borrar una tupla o actualizar cualquiera de sus columnas. Este algoritmo deberá almacenar los resultados de dicho análisis en el catálogo dinámico de datos. El objetivo de este algoritmo es preservar la integridad de la información.

Regla 7 (Inserción, Actualización y Borrado Masivo)

La capacidad de manejar una relación base o derivada a través de un solo operador, debe ser característica no solo de los comandos de la lectura de información, sino también de los comandos que se utilizan para actualizar, insertar y borrar información en la base de datos.

Regla 8 (Independencia Física de los Datos)

Los programas de aplicación y cualquier otra actividad que se efectúa sobre la información de la base de datos, no deben sufrir modificación alguna cuando alteramos las características lógicas. Esto implica que los programas no deberán modificarse cuando se lleve a cabo alguna modificación en la parte física de la base de datos.

Regla 9 (Independencia Lógica de los Datos)

Los programas de aplicación y cualquier otra actividad que se efectúa sobre la información de la base de datos, no deben sufrir modificación alguna cuando alteramos las características lógicas. Esto implica que los programas no deberán modificarse cuando se lleve acabo alguna modificación en la parte lógica de la base de datos.

Regla 10 (Independencia de Integridad)

Las reglas de integridad pertenecientes a una base de datos en particular deben poder definirse en el sublenguaje de base de datos y almacenarse en el catálogo dinámico de datos. Las reglas de integridad controlan que los datos que se insertan en la base de datos cumplan con ciertos requisitos o que al modificar información se lleven a cabo ciertas operaciones de mantenimiento en la información restante.

Regla 11 (Independencia de Distribución)

Un sistema relacional de base de datos puede considerarse distribuido, sólo cuando el Manejador de Base de Datos posee un sublenguaje que permite a los programas de aplicación permanecer sin modificaciones cuando:

a) Se utilicen diversos protocolos de comunicaciones

b) Los datos son redistribuidos.

Regla 12 (No-Subversión)

Si un sistema relacional tiene un lenguaje de bajo nivel. Dicho lenguaje no pude utilizarse para ignorar o subvertir las reglas de integridad o características específicas en el lenguaje relacional de alto nivel.

1.3 Modelo OSI y TCP/IP

Modelo OSI

Para que dos computadoras en la red puedan comunicarse, se deben considerar aspectos tan diversos que generalmente se aplica la técnica "divide y vencerás", partiendo el problema en varias tareas, cada una encargada de resolver un punto específico de la comunicación.[3]

De esta manera la conexión lógica entre las computadoras se divide en una serie de "capas" que, en conjunto, forman lo que se llama la arquitectura de la red, o bien la familia de protocolos de la red.

Con el fin de permitir que distintas arquitecturas diseñadas por distintos fabricantes puedan interactuar entre sí, la Organización de Estándares Internacionales (ISO, por sus siglas en inglés) inició en 1977 la definición de un modelo de referencia llamado el Modelo de Interconexión de Sistemas Abiertos (el modelo OSI), que divide la arquitectura de la red en siete capas, cada una con funciones específicas e independientes de las demás, como se muestra en la figura 1.1.

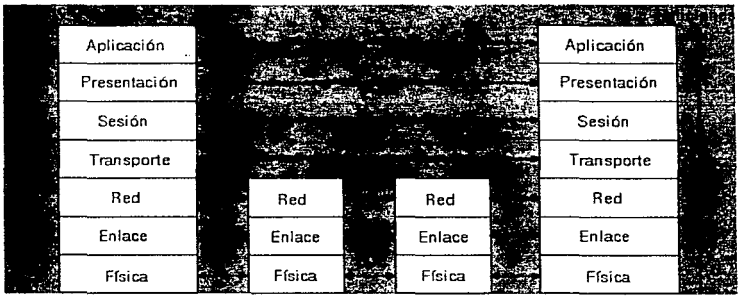


Fig. 1.1 Capas del modelo OSI.

[1] http://www.itam.mx/Info/Extuni/sat/comp/redes/conceptos.html

En este modelo, la capa N en una computadora, realiza sus funciones comunicándose con la capa del mismo nivel (llamada entidad par) en otra computadora. La comunicación entre entidades pares se lleva a cabo a través de reglas bien definidas - los protocolos - para una arquitectura de red en particular. Las funciones de la capa N sirven para que ésta pueda ofrecer un servicio a la capa inmediatamente superior en la misma computadora. De esta manera se van agregando servicios conforme se asciende por las capas de la arquitectura hasta llegar a la capa de aplicación, de donde los procesos de los usuarios de la red, obtienen sus servicios.

El modelo OSI no define protocolos ni servicios, éstos dependen de la implementación específica a cada arquitectura de red. Lo que el modelo define es la función que debe de realizar cada capa.

Sobre la capa de aplicación interactúan los usuarios en la red. En ella residen los procesos que otorgan los servicios de red, tales como el correo electrónico y transferencia de archivos. Para ofrecer estos servicios, los procesos de la capa de aplicación también se conocen entre sí, siguiendo un protocolo de aplicación.

A continuación se describen las diferentes capas que constituyen el modelo OSI:

Capa 1: Medio Físico

La capa física se ocupa de la transmisión de bits a lo largo de un canal de comunicación. Esta capa debe asegurar que cuando un nodo envía un bit con valor de 0, éste se reciba exactamente como un bit con ese valor en el otro extremo, y no como un bit con valor de 1.

En esta capa se define:

- Qué tipo de medio de comunicación se va a utilizar (cobre, fibra óptica, microondas, satélite, radio, etc.).
- La forma de realizar transmisiones bidireccionales en forma simultánea.
- La forma de establecer la conexión inicial y cómo interrumpirla.
- Cómo se definen los extremos de las conexiones en el medio físico; el tipo de conector, cable, etc.

Los problemas de diseño a considerar aquí son el aspecto mecánico, eléctrico, de procedimiento de interfase y el medio de transmisión física.

Capa 2: Capa de Enlace

La capa de enlace se encarga de que a partir de un medio de transmisión, éste se convierta en una línea sin errores de comunicación para la capa de red. Esta tarea la realiza a partir de transformar el flujo de datos entre las computadoras en tramas de datos.

Las tramas de datos se constituyen por cientos de bytes, que se deben de transmitir en forma secuencial. Estas tramas se deben procesar desde el nodo que origina

el mensaje, y una vez recibidas se procesarán las tramas de asentimiento, las cuales informan al nodo que está transmitiendo información que ésta ha llegado a su destino.

Como la capa física sólo se encarga de transmitir un flujo de bits sin tener en cuenta su significado y estructura, recae sobre la capa de enlace la creación y el reconocimiento de los límites de la trama.

Las tramas pueden ser susceptibles a errores debido a ruido en la línea, en cuyo caso la máquina que inició la transmisión de la trama, deberá volver a enviarla.

Capa 3: Capa de Red

La capa de red se ocupa del control de la operación de la subred. Su función principal consiste en cómo encaminar los paquetes del origen al destino. Las rutas entre los dos nodos podrían ser estáticas, en las cuales un dispositivo de conmutación conoce las direcciones de las máquinas en los puertos de entrada y salida del mismo; o también puede ser por medio de tablas dinámicas que se pueden determinar al inicio de la conversación entre dos nodos.

En las redes de difusión el problema de encaminamiento es simple, por lo cual la capa de red es normalmente muy pequeña o casi inexistente.

Capa 4: Capa de transporte

La función principal de la capa de transporte consiste en aceptar datos de la capa de sesión, dividirlos, siempre que sea necesario, en unidades tan pequeñas como sea necesario para pasarlos a la capa de red y asegurar que todos ellos lleguen correctamente al otro extremo.

Bajo condiciones normales, la capa de transporte crea una conexión de red distinta para cada conexión de transporte solicitada por la capa de sesión. El tipo más importante de conexión de transporte corresponde al canal punto a punto sin error, o por medio del cual se entregan los mensajes en el mismo orden en que fueron enviados.

La capa de transporte es una capa del tipo origen - destino o de extremo a extremo, formando entre los dos nodos un canal virtual.

Capa 5: Capa de Sesión

La capa de sesión permite que usuarios en distintas máquinas puedan establecer sesiones entre ellos. A través de una sesión se puede llevar un transporte de datos ordinario. Una sesión permitirá a un usuario acceder a un sistema de tiempo compartido a distancia, o transferir un archivo entre dos máquinas.

Uno de los servicios de la capa de sesión consiste en gestionar el control de diálogo. Las sesiones permiten que el tráfico vaya en ambas direcciones al mismo tiempo, o bien, en una dirección en un instante dado.

La administración del testigo es otro de los servicios relacionados con esta capa, para el caso de algunos protocolos resulta esencial que ambos lados no traten de realizar la misma operación en el mismo instante. Para manejar estas actividades, la capa de

sesión proporciona testigos que pueden ser intercambiados. Solamente el extremo con el testigo puede realizar la operación crítica.

Capa 6: Capa de Presentación

La capa de presentación se ocupa de los aspectos de sintaxis y semántica de la información que se transmite.

Los programas de usuario, en realidad no intercambian tramas entre sí, sino códigos, nombres, llamadas a funciones, etc. En el caso de una red de cómputo, se pueden estar realizando consultas a máquinas con distinto sistema operativo, las cuales pueden almacenar internamente los datos en distintos formatos (ASCII, EBCDIC, etc.). Para permitir la comunicación entre distintos nodos con diferentes representaciones de la información, la estructura de los datos a intercambiarse puede definirse en forma abstracta, junto con una norma de codificación para ambas máquinas.

Capa 7: Capa de Aplicación

La capa de aplicación contiene una variedad de protocolos que se necesitan frecuentemente.

Por ejemplo, hay cientos de terminales incompatibles en el mundo. Una forma de resolver esto es con una aplicación de terminal virtual, con el que editores y otros programas pueden ser escritos para trabajar con él.

Otra función de la capa de aplicación es la de transferir archivos. La transferencia de archivos entre dos sistemas diferentes requiere de la resolución de éstas y otras incompatibilidades.

Este servicio, así como el correo electrónico, la entrada de datos remota, el servicio de directorio y otros servicios de propósito general y específico, también corresponden a la capa de aplicación.

TCP/IP[4]

El modelo OSI describe un protocolo de comunicación idealizado, y aunque TCP/IP no corresponde directamente a este modelo se puede hacer una referencia de semejanza entre ambos, la cual se describe a continuación.

Un protocolo de comunicación de datos se compone de una serie de reglas que describen como el software y el hardware deben de interactuar en una red. Para que una red funcione correctamente la información debe de ir de un punto a otro de una forma estandarizada, es por eso que los diseñadores definieron el concepto de protocolo de comunicación.

La unidad básica que viaja a través de una red TCP/IP generalmente es denominada paquete. Cada paquete es muy parecido (en contenido) a una carta, la cual

[4] Solaris 2.4 TCP/IP Network Administration Guide, pag 1-24, Sunsoft, USA, 1994. Pag. 13

lleva en un sobre la dirección del destinatario y del remitente, que para el caso del paquete corresponde al encabezado, así como una hoja con datos en su interior, que en el paquete corresponde a un mensaje, que a su vez se conforma de un determinado número de bytes, dependiendo del medio de transmisión de datos.

La dirección de estos paquetes se compone de una dirección lógica y una física:

- Dirección Lógica: Llamada también dirección IP, es la identificación de una máquina en una red, esta dirección es la que ayuda a determinar dónde está localizada una máquina en la red. Esta dirección es única y para garantizar lo anterior es asignada por una organización denominada interNIC.

- Dirección Física: Generalmente denominada dirección Ethernet, cada máquina tiene una dirección física, esto es que se contiene en el hardware (tarjeta de red). Aquí es importante subrayar que cada dirección física es única.

La mayoría de los protocolos de comunicación están estructurados en capas, algunas veces referenciadas como pila del protocolo. Cada capa está diseñada para una actividad en específico y existen tanto en el transmisor como en el receptor, y estas actividades actúan de manera independiente a lo que sucede en las capas de arriba o de abajo.

En la figura 1.2 se muestra la correspondencia entre las capas del modelo OSI y las capas de TCP/IP, así como ejemplos del tipo de aplicaciones y estándares que corresponden a cada una de estas capas.

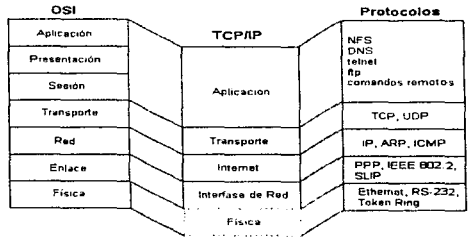


Fig. 1.2 Correspondencia entre las capas ISO-OSI y TCP/IP.

Capas de TCP/IP

Ahora se dará una breve descripción de las capas de TCP/IP, incluyendo una breve descripción de algunas de las aplicaciones y estándares que se pueden localizar en cada capa:

1) **Capa física de red.** Especifica el tipo de hardware que se utilizará en la red. Algunos ejemplos son los estándares de Ethernet, IEEE 802.3, y RS-232.

2) **Capa Interfase de Red.** Especifica el tipo de paquete que se utilizará para la transmisión, en este caso TCP/IP. Ejemplos para este caso son Ethernet, IEEE 802.2, PPP y SLIP.

3) **Capa Internet.** Es la correspondiente en el modelo OSI de la capa de red y se encarga de recibir y entregar paquetes a través de la red. Como ejemplos de protocolos a este nivel podríamos mencionar:

   - Protocolo IP. Se encarga de manejar las direcciones IP, formateo de paquetes (conocidos como datagramas IP) y de fragmentar los paquetes muy grandes en paquetes más pequeños.

   - Protocolo ARP. Este protocolo conceptualmente existe entre las capas de ligado de datos e Internet, se encarga de hacer el mapeo entre dirección física (48 bits de largo) y dirección lógica (32 bits de largo).

   - Protocolo ICMP. Es el protocolo responsable de detectar errores en la red y reportarlos, reporta principalmente cuando los paquetes llegan demasiado rápido para procesarlos, cuando una máquina no contesta a las peticiones realizadas hacia la misma y enrutando los paquetes hacia otra máquina que posiblemente sabrá resolver la dirección IP.

4) **Capa de transporte.** Se encarga de que los paquetes sean recibidos en la misma secuencia en que fueron generados y sin error. Como ejemplos de protocolos en esta capa se podrían mencionar:

   - Protocolo TCP. Da la capacidad a las aplicaciones de poderse comunicar entre ellas, aún a través de circuitos, de manera que las aplicaciones sólo ven una transmisión carácter y no por paquetes. Proporciona un proceso de confirmación de bytes enviados y recibidos.

   - Protocolo UDP. Provee un servicio de envío y entrega de datagramas. No contiene el servicio de verificación de conexión y desconexión entre un proceso y otro, se usa para enviar cantidades pequeñas de datos en lugar de utilizar TCP.

5) **Capa de aplicación.** Define las aplicaciones y servicios de red de alto nivel, tales como: telnet, ftp, tftp, comandos remotos de UNIX, servicios de archivos como NFS, servicios de nombres de servidores como DNS.

1.4 Interfase de sockets de Berkeley e implementación Winsock

El término sockets se refiere al API (Application Programming Interfaces) que está implementada con UNIX BSD. La interfase de sockets es usada por UNIX para permitir a los procesos que se comuniquen entre sí. Los procesos pueden estar en el mismo servidor o distribuidos a través de la red. Estos procesos pueden también ser aplicaciones corriendo en otras plataformas, con la única restricción de que la aplicación soporte el API de sockets.

La Interfase de sockets fue desarrollada por la Universidad de California en Berkeley y ha sido incluida en muchas versiones de UNIX, debido a que el UNIX BSD ha sido portado para correr en una gran variedad de equipos desde PC's de escritorio hasta mainframes. El uso de sockets ha sido ampliamente adoptado como un mecanismo de comunicación. Adicionalmente, debido a que hay una gran variedad de compiladores de C disponibles, la mayoría de estos compiladores actuales han incluido soporte para rutinas de sockets.

Un socket es simplemente un término cuyo origen se encuentra en el sistema operativo UNIX para un extremo de comunicación. Para que los procesos puedan comunicarse entre sí, un canal de comunicaciones debe ser abierto entre ellos. Un socket es un extremo de este canal de comunicaciones.

Un socket contiene una dirección IP y un número de puerto de cada máquina en cada extremo del canal de comunicaciones. Como previamente se mencionó, los procesos que se comunican mediante este canal, pueden estar en la misma máquina, o en diferentes máquinas distribuidas en la red. El número de puerto es simplemente un número lógico entre 1 y 65,535.

Las aplicaciones de sockets involucran 2 procesos: un proceso servidor y un proceso cliente. El proceso servidor recibe conexiones desde una aplicación cliente y procesa los datos recibidos. El proceso cliente realiza conexiones hacia el proceso servidor y usualmente transmite datos. En algunas ocasiones los procesos servidores pueden actuar como procesos cliente y viceversa, esto depende totalmente de la aplicación. Adicionalmente los procesos servidores pueden ser de un sólo hilo (single-threaded) o multihilo (multi-threaded). Si un proceso servidor es de un solo hilo, aceptará y procesará la conexión de un cliente a la vez. Si el proceso servidor es multihilo, múltiples conexiones de varios clientes, pueden ser manejadas a la vez. La técnica consiste en esperar a recibir una conexión del cliente e iniciar un proceso para manejar la transferencia de datos. En este punto, el proceso servidor regresa al punto inicial para esperar una nueva conexión del cliente.

1.4.1 Interfase de sockets de Berkeley

Una Interfase de Programa de Aplicación (API, Application Program Interface) es, simplemente, un grupo de funciones que los programadores utilizan a fin de desarrollar programas de aplicación para un ambiente de cómputo específico. Por ejemplo, para los desarrolladores de software que desean crear aplicaciones de Windows, Microsoft proporciona una extensa colección de rutinas de programación: la API de Windows.

En la década de los ochenta, la Agencia de Investigación de Proyectos Avanzados (ARPA, Advanced Research Projects Agency) del gobierno de Estados Unidos

proporcionó fondos a la Universidad de California en Berkeley para implementar los protocolos TCP/IP (Transport Control Protocol/Internet Protocol) bajo el sistema operativo UNIX. Durante este proyecto, un grupo de investigadores desarrollo un API de comunicaciones en redes TCP/IP, esta API recibió el nombre de Interfase de sockets[5].

Al principio, los desarrolladores modelaron la Interfase de sockets a partir de las llamadas del sistema UNIX ya existentes. De hecho, originalmente la diseñaron para que utilizara las llamadas a funciones de UNIX, es decir, la construyeron dentro del entorno del sistema operativo UNIX. En este sistema operativo, las llamadas de E/S tienen la forma de procesos *abrir-leer-escribir-cerrar*. Para utilizar un archivo, el programador de UNIX primero lo abre, después ejecuta las operaciones de lectura y escritura y, por último, lo cierra. Los programadores de UNIX siguen los mismos pasos para tener acceso a dispositivos de hardware. Dicho de otro modo, en UNIX se utilizan las mismas llamadas al sistema para acceder a impresoras, unidades de cinta y archivos. UNIX mapea los dispositivos de hardware y archivos dentro del sistema de archivos del sistema operativo. Para abrir un archivo o para acceder a un dispositivo como una unidad de cinta, los programadores llaman a las mismas funciones del sistema. En respuesta a tales llamadas, la función devuelve un apuntador o manejador, el cual se llama, en UNIX descriptor de archivo, que apunta a un registro en una tabla que describe al archivo o dispositivo. Para un archivo, el registro del descriptor de archivo en la tabla contiene información como el nombre, tamaño y fecha del archivo. Un descriptor de archivo de UNIX puede apuntar a un archivo, a un dispositivo de hardware o cualquier otro objeto que cumpla funciones del sistema de E/S.

En las primeras etapas de diseño de la Interfase de sockets, los investigadores de Berkeley intentaron hacer funciones de E/S de red como cualquier otra función del sistema de E/S de UNIX. Querían diseñarla para que usara el proceso de *abrir-leer-escribir-cerrar*. Por lo tanto, para comunicarse con una red TCP/IP, el programa primero abre una conexión con la red, después lee y escribe datos a través de ella y, por último, cierra la conexion. Desde la perspectiva UNIX, este enfoque de diseño hace una comunicación en red como cualquier otra función del sistema de E/S y por tanto más fácil de integrar al sistema operativo.

Conforme progresó el proyecto de Berkeley, los desarrolladores descubrieron que la E/S en red era mucho más complicada que cualquier otro tipo de E/S. Para implementarla tuvieron que superar diferentes obstáculos. Casi cualquier comunicación en red sigue el modelo Cliente/Servidor. Por tanto, los diseñadores de la Interfase de sockets pudieron fácilmente modificar la API del sistema de E/S de UNIX para permitir que los desarrolladores crearan programas cliente que hicieran una búsqueda activa para establecer una conexión de red. Sin embargo una API de red también debe permitir a los desarrolladores crear programas servidores que aguarden pasivamente a que un programa cliente haga contacto. Como el sistema de E/S común de UNIX no incorpora vasta capacidad pasiva, los diseñadores de sockets tuvieron que crear nuevas funciones del sistema para manejar operaciones pasivas de E/S.

Asimismo, encontraron que las funciones de comunicación del sistema de E/S de UNIX eran inadecuadas para satisfacer los requerimientos de una red. En general, el sistema de E/S de UNIX utiliza direcciones fijas para comunicarse con archivos y

[5] Debido a que la Universidad de California en Berkeley estuvo involucrada en el desarrollo de la Interfase de sockets, con frecuencia la gente se refiere a ella como Interfase de sockets de Berkeley o sockets de Berkeley.

dispositivos, es decir, la ubicación (dirección) de un archivo o dispositivo en una computadora no cambia. Además, la conexión (o ruta de datos) hacia un archivo o dispositivo está disponible durante todo el ciclo de *lectura-escritura*, hasta que el programa cierra el archivo o dispositivo.

Las direcciones fijas funcionan bien para comunicaciones de red orientadas a conexión, pero para comunicaciones sin conexión, presentan problemas. Por ejemplo, cuando un programa de red transmite un datagrama especifica la dirección destino (la dirección IP del anfitrión destino), pero no establece una conexión punto a punto con la computadora destino. El sistema de E/S de UNIX no ofrecía facilidades para este tipo de conexión. En otras palabras, al principio, el sistema de E/S de UNIX sólo podía hacer operaciones de lectura-escritura por flujo de bytes. De acuerdo con los problemas descritos y otros obstáculos similares, los diseñadores de sockets abandonaron el enfoque de E/S de UNIX, y en lugar de modificar el código de la API existente, agregaron nuevas funciones a UNIX.

1.4.2 Implementación Winsock

El software llamado Windows Sockets (comúnmente conocido como Winsock) es una API basada en el modelo de sockets. De hecho, sus desarrolladores derivaron esta API de la Interfase de sockets de Berkeley. Sin embargo, mientras la interfase de Berkeley es una API que pueden usar los programadores con múltiples sistemas operativos, la API Winsock se dirige específicamente a la familia de sistemas operativos de Microsoft Windows.

Winsock incluye muchas de las funciones de Berkeley desarrolladas para UNIX, así como extensiones específicas de Windows que permiten a los programadores aprovechar el ambiente manejado con mensajes de ese entorno operativo.

La especificación de Windows Sockets define un estándar aceptado para el desarrollo de programas en ambiente de Windows en redes TCP/IP. Así, la API de Windows Sockets ofrece enormes oportunidades para los programadores que desean escribir aplicaciones para redes. El esfuerzo de definir la especificación de Winsock incluye a un gran número de personas de diversas instituciones. La meta de este esfuerzo es definir una sola API que puedan emplear los programadores y proveedores de software para red como estándar al desarrollar aplicaciones para redes basadas en Windows. Windows Sockets implementa la Interfase de sockets como una biblioteca de enlace dinámico[6].

La API Winsock proporciona una colección (o biblioteca) de funciones que pueden usar sus programas para lograr tareas específicas. La especificación de Winsock organiza la biblioteca en tres grupos:

[6] En Windows, una biblioteca de enlace dinámico (DLL) proporciona un método estándar para agregar nuevas funcionalidades que pueden emplear los programas durante su ejecución. Una biblioteca de enlace dinámico es un módulo de código ejecutable que Windows puede cargar cuando se le requiera. Windows puede también descargar una DLL cuando él y otros programas ya no necesiten los módulos de código que almacena la biblioteca. El diseño de la mayoría de las DLL de Windows permite que múltiples programas las usen al mismo tiempo.

- Las funciones de los sockets de Berkeley incluidas en la API original.
- Funciones de base de datos que permiten que sus programas obtengan información de Internet acerca de los nombres de dominios, servicios de comunicaciones y protocolos.
- Las extensiones específicas de Windows a las rutinas de los sockets de Berkeley.

Cada una de estas funciones se identifican como de bloque o de no bloqueo. Una función de bloqueo evita que el programa llame a cualquier otra función de Winsock hasta que la propia función de bloqueo termine sus operaciones en la red. Una función de no bloqueo termina de inmediato su operación, o regresa con un mensaje de error. En otras palabras, una función de no bloqueo no espera hasta que la operación concluye.

Funciones de conexión

Una operación de bloqueo evita que los programas ejecuten alguna otra de las funciones de Winsock hasta que esa operación concluya. Es común que las operaciones de bloqueo sean funciones estilo Berkeley que realizan la E/S de información a la red, es decir, a una operación de bloqueo habitualmente corresponde un retraso en la recepción o envío de información a través de la red. La tabla 1.1 muestra las rutinas de sockets estilo Berkeley que pueden bloquear la API de Winsock.

De la descripción mostrada en la tabla 1.1 puede observarse que cada función realiza algún tipo de E/S en la red, o espera a que ocurra antes de terminar su operación. Puede notarse también que cualquier función que ejecute o dependa de E/S en la red puede bloquear, casi siempre otras funciones de Winsock.

Función	Descripción
accept	Confirma una conexión de entrada. Crea un socket nuevo y lo conecta al anfitrión remoto que pidió la conexión. Devuelve el socket original a su estado de atención.
Closesocket	Cierra un extremo de una conexión por sockets.
Connect	Inicia una conexión en el socket especificado
Recv	Recibe información de un socket conectado.
Recvfrom	Recibe información de un socket conectado o de uno no conectado.
Select	Ejecuta multiplexajo síncrono de E/S al monitorear el estado de múltiples sockets.
Send	Envía información a un socket conectado.
Sendto	Envía información a un socket conectado o a uno no conectado

Tabla 1.1 Funciones de bloqueo de la API Winsock.

Por otro lado, las funciones que muestra la tabla 1.2 no requieren E/S en la red para terminar sus operaciones, sino que utilizan información almacenada de manera local, o solo trabajan con el extremo de una conexión del socket de la propia computadora. Estas funciones no ejecutan operaciones que requieran comunicación con un anfitrión remoto. Por lo tanto se conocen como funciones de no bloqueo.

Función	Descripción
bind	Asigna un nombre local a un socket sin nombre.
Getpeername	Obtiene el nombre del compañero conectado al socket especificado.
Getsockname	Obtiene el nombre local para el socket especificado
Getsockopt	Obtiene opciones asociadas con el socket especificado
Htonl	Convierte un número de 32 bits de ordenamiento de byte de anfitrión a ordenamiento de byte de red.
Htons	Convierte un número de 16 bits de ordenamiento de byte de anfitrión a ordenamiento de byte de red.
Inet_addr	Convierte una cadena de caracteres que representa una dirección IP en notación decimal al valor binario de 32 bits en ordenamiento de byte de red.
Inet_ntoa	Convierte una dirección IP a notación decimal.
IoctlSocket	Controla varios parámetros relacionados con la forma de operar del socket y el manejo de la E/S de red
listen	Indica a un socket específico que atienda las conexiones entrantes. Es decir coloca al socket en modo de escucha
Ntohl	Convierte un número de 32 bits de ordenamiento de byte de red a ordenamiento de byte de anfitrión.
Ntohs	Convierte un número de 16 bits de ordenamiento de byte de red a ordenamiento de byte de anfitrión.
Setsockopt	Almacena opciones asociadas con el socket especificado.
Shutdown	Cierra parte de una conexión full dúplex (sólo en el anfitrión local).
Socket	Crea un extremo para la comunicación y devuelve un identificador de socket.

Tabla 1.2 Funciones de no-bloqueo de la API Winsock.

Funciones de base de datos

Los programas basados en Internet funcionan con diferentes tipos de direcciones. Por ejemplo, un usuario puede especificar un nombre de dominio, como *microsoft.com*, o como una dirección en notación decimal (168.158.20 102). Sin embargo, antes de que los programas en red basados en Windows puedan procesar tales direcciones, siempre deben convertirlas a una estructura de datos de socket que puedan entender la mayoría de las funciones de Winsock. Asimismo, cuando los programas necesitan mostrar la información de la dirección del anfitrión a un usuario, casi siempre deben convertir la información de la dirección que está en el formato de la estructura de datos de socket a una forma que el usuario pueda entender. Las funciones de base de datos de Winsock mostradas en la tabla 1.3 pueden realizar estas tareas de conversión de direcciones para sus programas. Estas funciones permiten que los programas obtengan información de Internet sobre nombres de dominio, servicios de comunicación y protocolos. Para encontrar tal información, estas funciones pueden tener acceso a una gran variedad de fuentes de bases de datos (locales y remotas). La especificación de Winsock define la interfase, así como la información que las funciones de bases de datos deben devolver. No obstante, la implementación de Winsock determina exactamente dónde almacena y cómo obtiene Winsock la información.

Función	Descripción
gethostbyaddr	Obtiene el nombre de dominio y dirección IP correspondiente a una dirección de red.
Gethostbyname	Obtiene el nombre de dominio y dirección IP correspondiente a un nombre de anfitrión.
Gethostname	Obtiene el nombre de dominio del anfitrión local.
Getprotobyname	Obtiene un protocolo por nombre y devuelve el nombre oficial y el número definido para representar al protocolo.
Getprotobynumber	Obtiene el nombre y número del protocolo representado por un número específico.
Getservbyname	Obtiene un nombre de servicio y el puerto de protocolo correspondiente al nombre del servicio.
Getservbyport	Obtiene el nombre del servicio y el puerto correspondiente a un puerto de protocolo específico.

Tabla 1.3 Funciones de base de datos de la API Winsock.

Varias de las funciones de bases de datos de Winsock devuelven apuntadores (o estructuras) de datos volátiles. La implementación de Winsock (WINSOCK.DLL) puede reusar áreas de datos volátiles con cada llamada a una función de Winsock. De esta forma, si un programa necesita cualquier información contenida en un área de memoria volátil, debe copiarse en un lugar diferente de la memoria, antes de llamar a otra función de Winsock. La información volátil que contienen las áreas del búfer de Winsock sólo es válida hasta la próxima llamada que haga el programa a una función de Winsock. La API Winsock incluye versiones asíncronas de todas las funciones de base de datos, excepto para la función `gethostname`; estas funciones asíncronas son extensiones específicas de Windows a la Interfase de sockets de Berkeley. Los desarrolladores de Winsock diseñaron las funciones asíncronas para aprovechar las capacidades basadas en mensajes inherentes a Windows. La tabla 1.4 enlista siete funciones estilo Berkeley y las funciones de base de datos asíncronas incluidas en la API Winsock.

Función estilo Berkeley	Función asíncrona equivalente
gethostbyaddr	WSAAsyncGetHostByAddr
gethostbyname	WSAAsyncGetHostByName
getprotobyname	WSAAsyncGetProtoByName
getprotobynumber	WSAAsyncGetProtoByNumber
getservbyname	WSAAsyncGetServByName
getservbyport	WSAAsyncGetServByPort

Tabla 1.4 Funciones asíncronas de base de datos de la API Winsock.

Funciones de extensión específicas de Microsoft Windows

Los desarrolladores de Winsock diseñaron versiones asíncronas especiales de ciertas funciones de sockets para que los programadores pudieran aprovechar las capacidades el uso de mensajes dentro de Windows. La tabla 1.5 muestra este tipo de funciones incluidas dentro del API de Winsock.

Función	Descripción
WSAAsyncSelect	Ofrece una versión asíncrona de la función select.
WSACancelAsyncRequest	Cancela una instancia de la función WSAAsyncGetXByY.
WSACancelBlockingCall	Cancela una llamada de bloqueo de la API.
WSACleanup	Termina sesión de los DLL subyacentes de Winsock.
WSAGetLastError	Obtiene detalles sobre el último error de la API Winsock.
WSAIsBlocking	Determina si el DLL de Winsock subyacente está bloqueado.
WSASetBlockingHook	Engancha el método de bloqueo que usa la implementación subyacente de Winsock.
WSASetLastError	Establece el regreso de error para WSAGetLastError subsecuente.
WSAStartup	Inicia el DLL de Winsock subyacente.
WSAUnhookBlockingHook	Restaura la función original de bloqueo.

Tabla 1.5 Funciones de extensión específicas de Windows.

1.5 Trasferencia de Archivos Via FTP

La capacidad de intercambiar archivos con otros sistemas es una de las grandes ventajas de las comunicaciones entre computadoras, y con los protocolos actuales de transferencia de archivos, esta tarea resulta más fácil y rápida que nunca. Podemos transferir directamente archivos de bases de datos, hojas de cálculo, aplicaciones completas o documentos situados en otras computadoras remotas a nuestros discos, listos para ser usados. Podemos transferir archivos directamente de nuestros discos a los de una computadora remota. Este método de transferencia de archivos es más rápido y agradable que enviar disquetes por correo, y a menudo resulta ser la única forma práctica de transferir datos entre equipos que tienen formatos de disco incompatibles.

Para asegurarse de la correcta transferencia de los datos, las computadoras utilizan protocolos de transferencia de archivos, que definen procedimientos para intercambiar datos, junto con instrucciones para coordinar el proceso. La mayoría de protocolos realizan corrección de errores. Es decir, detectan cuándo los datos son incorrectos o se han perdido debido al ruido presente en la conexión, y volverán a retransmitirlos automáticamente hasta que se reciban automáticamente.

Entre los protocolos de transferencia de archivos con corrección de errores introducidos desde la aparición de las microcomputadoras, sólo han destacado alrededor de una docena de ellos. Debido a que los principales programas de comunicaciones ofrecen una selección de estos protocolos, generalmente podremos elegir, no existe una opción definitiva. Algunos protocolos funcionan bien con algunos módems y mal con otros, otros son lentos y sólo deben usarse cuando la computadora remota no nos deja otra elección. Finalmente, algunos son extraordinariamente rápidos, pero sólo los incorporan algunos programas. Para saber qué protocolo nos dará los mejores resultados, necesitamos adquirir alguna base sobre el funcionamiento de los protocolos.

Durante la última década, los aficionados y profesionales han desarrollado una oleada de nuevos protocolos como FTP. Mientras muchos de ellos han quedado en la oscuridad, algunos se han impuesto por méritos propios.

1.5.1 Protocolo de Transferencia de Archivos (FTP)

El File Transfer Protocol (FTP) es una utilería para la administración de archivos a través de máquinas, sin tener que establecer una sesión remota mediante Telnet[7]. FTP lo habilita para transferir archivos de ida y vuelta, administrar directorios y tener acceso a correo electrónico. FTP no está diseñado para habilitar el acceso a otra máquina para ejecutar programas, pero es la mejor utilería para el manejo de archivos.

El protocolo de transferencia de archivos utiliza dos canales de TCP. El puerto 20 de TCP es el canal de datos, en tanto que el puerto 21 es el canal de comandos. FTP difiere de la mayor parte de otros programas de transferencia de archivos en que utiliza dos canales, permitiendo transferencias simultáneas de comandos FTP y de datos. También es distinto en otro aspecto importante: FTP conduce todas las transferencias de archivos en primer plano, y no en segundo plano. FTP no utiliza integradores o colas de espera. Al utilizar TCP, FTP elimina la necesidad de preocuparse sobre confiabilidad o la administración de la conexión, porque FTP se puede apoyar en TCP para realizar adecuadamente esas funciones.

En el lenguaje FTP, los dos canales que existen entre ambas máquinas se conocen como el intérprete de protocolo (PI) y el proceso de transferencia de datos (DTP). PI transfiere las instrucciones entre las dos implementaciones mediante el canal de datos 21 de TCP, y DTC transfiere archivos sobre el canal de datos 20 de TCP. Esto se muestra en la figura 1.3.

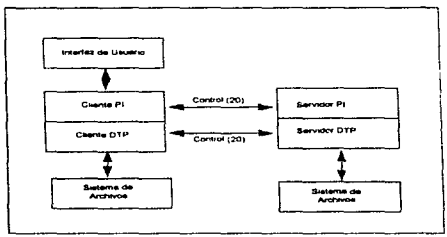


Fig. 1.3 Conexión FTP en dos canales.

FTP es similar a Telnet en que utiliza un programa de servidor que corre continuamente y un programa por separado que se ejecuta en el cliente. En sistema UNIX, estos programas se conocen como FTPD y FTP, respectivamente.

[7] TCP/IP. Timothy Parker. Cap.3.

1.5.2 Puertos

Todas las aplicaciones de capa superior que utilizan TCP tienen un número de puerto que las identifica. En teoría, los números de puerto que pueden asignar en máquinas individuales, o según desee el administrador, pero se han adoptado algunas reglas convencionales para permitir una mejor comunicación entre varias implementaciones de TCP. Esto permite que el número de puerto identifique el tipo de servicio que un sistema TCP está solicitado de otro. Los números de puerto se pueden cambiar, aunque esto puede causar dificultades. La mayor parte de los sistemas mantiene un archivo de los números de puerto y sus servicios correspondientes.

Típicamente, los números de puerto mayores a 255 se reservan para el uso privado de la máquina local, pero los números inferiores a 255 se utilizan para procesos de uso frecuente. Internet Assigned Numbers Authority publica una lista de los números de puertos de uso frecuente. En la tabla 1.6 se muestra algunos de los números de puertos que se utilizan comúnmente, los números 0 y 255 son reservados.

Puerto Número	Nombre del Proceso	Descripción
1	TCPMUX	Multiplexor de servicio de puerto TCP
5	RJE	Entrada de tarea remota
7	ECHO	Eco
9	DISCARD	Descartar
11	USERS	Usuarios activos
13	DAYTIME	Hora hábil
17	QUOTE	Cita día
19	CHARGEN	Generador de caracteres
20	FTP-DATA	Protocolo de transferencia de archivos- Datos
21	FTP	Protocolo de transferencia de archivos- Control
23	TELNET	Telnet

Tabla 1.6 Puerto TCP de uso frecuente.

Cada circuito de comunicaciones dentro y fuera de la capa TCP se identifica en forma única mediante la combinación de dos números, los cuales en conjunto se conocen como socket.

1.5.3 Conexiones FTP

Por lo general, FTP se inicia con el nombre o con la dirección de la máquina destino. Igual que para Telnet, deberá ser posible convertir el nombre en una dirección IP para que tenga éxito el comando. La máquina destino también se puede especificar desde la línea de comandos FTP.

Cuando FTP se conecta con la máquina destino, usted será capaz de registrarse como si fuera una conexión a través de Telnet. Algunos sistemas permiten un registro de entrada anónimo o huésped, para transferir archivos FTP (utilizando por lo general su nombre de registro de entrada como contraseña y registro de acceso), en tanto que la mayor parte le exige que tenga acceso normal a la máquina. En redes grandes, donde se utilizan sistemas como "Sección Amarilla", por lo general en la mayor parte de las máquinas se permiten los registros de entrada. Si no utiliza está, deberá estar en el archivo válido de usuario para obtener acceso FTP. Al igual que en Telnet, puede registrarse en la estación remota con una identificación de usuario distinta a la de su registro de máquina local. Para transferir archivos, debe tener los permisos adecuados en la máquina remota.

Después de registrarse mediante FTP, en realidad no está en la máquina remota. Lógicamente sigue en el cliente, por lo que todas las instrucciones referentes a transferencias de archivo y a movimientos de directorios deben relacionarse con su máquina local y no con la remota.

El proceso que sigue FTP cuando se establece una conexión es el siguiente:

1. Registro de entrada: verifica identificación y contraseña del usuario.
2. Define directorio: identifica el directorio de inicio.
3. Define modo de transferencia de archivos: define tipo de transferencia.
4. Inicia transferencia de datos: habilita los comandos de usuario.
5. Detiene la transferencia de los datos: cierra la conexión.

Estos pasos se ejecutan en orden para cada conexión. Para controlar FTP un usuario tiene cierto número de comandos a su disposición.

1.5.3 Transferencias FTP de Terceros

FTP permite que ocurra transferencias a través de una tercera máquina, colocada entre cliente y servidor. Este procedimiento se conoce como transferencia vía terceros, y a veces resulta necesario para obtener los permisos adecuados de acceso a máquina remota. En la figura 1.4 se muestra el diagrama de una transferencia vía terceros, con la conexión de control realizada a través a través de una tercera máquina.

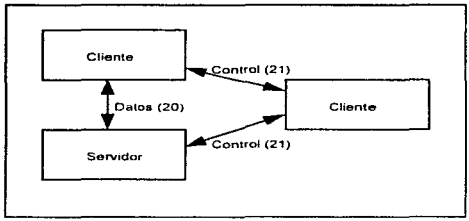


Fig. 1.4 Transferencia FTP por terceros.

Al establecer una conexión vía terceros, el cliente abre las conexiones de control entre la máquina remota y segundo cliente que maneja el canal de control. Solamente el canal de control pasa a través del segundo cliente, en tanto que el canal de datos pasa directamente entre dos extremos.

Cuando se emite una solicitud de transferencia, ésta se transmite a través del segundo cliente, mismo que verifica los permisos y a continuación pasa la solicitud al servidor. La transferencia de los datos puede tener lugar directamente, en vista de que los permisos se verificaron en el canal de control.

1.5.4 Acceso FTP Anónimo

Para habilitar capacidades de transferencia de archivos, FTP requiere una identificación y contraseña de usuario, pero existe un método más liberal para permitir acceso general a un archivo o directorio, conocido como FTP anónimo. Este elimina la necesidad de una cuenta de registro de entrada en la maquina remota, y permite por lo general un registro de entrada anónimo mediante una contraseña que puede ser huésped o el nombre de registro de entrada real de usuario.

Si el sistema remoto está definido para permitir registros de entrada anónimos, se le solicitará contraseña y a continuación se hará una advertencia sobre limitaciones al acceso. Si en el sistema remoto hay un archivo que usted necesita, el comando get lo transferirá. Los sitios FTP anónimos se están volviendo comunes, especialmente en vista del interés creciente por Internet.

Una vez descritos los antecedentes de ambiente sistemático en el cual el trabajo será desarrollado, podremos profundizar en el próximo capítulo, la problemática establecida, definiendo el esquema actual, los objetivos y la propuesta de soluciones factibles para la elaboración de este sistema.

Capítulo
DOS

DEFINICIÓN DEL PROBLEMA Y ESTUDIO DE FACTIBILIDAD

Este capítulo nos introduce a nuestro caso de estudio. Comenzamos con una breve descripción de la situación actual para poder identificar el problema a resolver.

Una vez identificado el problema se plantearán los alcances y objetivos deseados para poder proponer distintas soluciones .

2.1 Problemática Actual

La trascendencia de las funciones que realiza el Banco de México se deriva de la responsabilidad que nuestro país le ha conferido como Banco Central. Estas funciones están referidas al ámbito monetario y crediticio en que se desenvuelve la actividad económica del país.

El Banco de México tiene por finalidad proveer a la economía del país de moneda nacional. En la consecusión de esta finalidad tiene como objetivo prioritario, procurar la estabilidad del poder adquisitivo de dicha moneda, promover el sano desarrollo del sistema financiero y propiciar el buen funcionamiento de los sistemas de pagos.

El Banco de México debe realizar las funciones siguientes: Regular la emisión y circulación de la moneda, los cambios, la intermediación y los servicios financieros, así como los sistemas de pagos; operar con las instituciones de crédito como banco de reserva y acreditante de última instancia; prestar servicios de tesorería al Gobierno Federal y actuar como agente financiero del mismo; fungir como asesor del Gobierno Federal en materia económica y, particularmente, financiera; participar en el Fondo Monetario Internacional y en otros organismos de cooperación financiera internacional o que agrupen a bancos centrales, y operar con dichos organismos, con bancos centrales

y otras personas morales extranjeras que ejerzan funciones de autoridad en materia financiera.

Para realizar las funciones mencionadas se han desarrollado distintos sistemas de software que interactuan entre el Banco de México y las instituciones financieras. Entre estos se encuentran sistemas de vital importancia, tales como: SPEUA (Sistema de Pagos Electrónicos de Uso Ampliado), SIAC (Sistema de Información A Cuentahabientes), SAGA (Sistema de Administración de Garantías), etc. como se muestra en la figura 2.1.

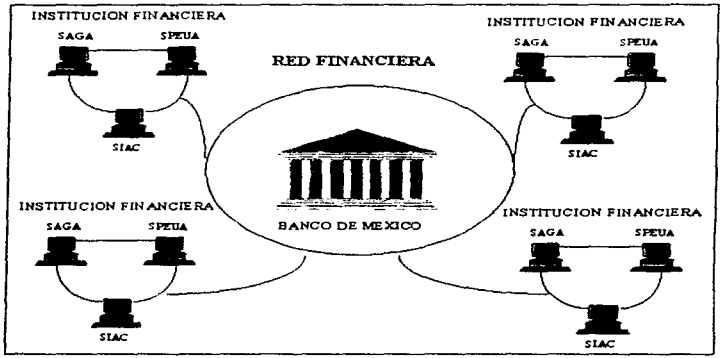


Fig. 2.1 Interacción de las instituciones financieras con el Banco de México.

Debido a que estos sistemas permiten la comunicación en línea y el registro de transacciones con el Banco de México, es muy importante mantenerlos actualizados de forma oportuna.

Estos sistemas están construídos bajo la arquitectura cliente/servidor. La parte del servidor se encuentra localizada en el centro de cómputo del Banco de México y la parte del cliente en las terminales de las instituciones financieras.

La parte del cliente se modifica constantemente por aspectos tales como seguridad, confiabilidad, mejor manejo de errores, etc., y esto origina distintas versiones de los sistemas utilizados.

El principal problema es proporcionar a las instituciones financieras las nuevas versiones de sistemas en forma oportuna.

Actualmente se necesita disponer de recursos humanos y materiales tanto por parte del Banco de México como de las instituciones afines para poder distribuir e instalar las aplicaciones. Esta distribución se hace por medio de discos flexibles, lo cual es muy poco eficiente porque retarda la actualización y provoca un alto costo de recursos, además de no permitir un control confiable de las versiones distribuidas.

Como se puede observar en la figura 2.2, el proceso de distribuición y actualización de versiones de software se realiza de la siguiente forma:

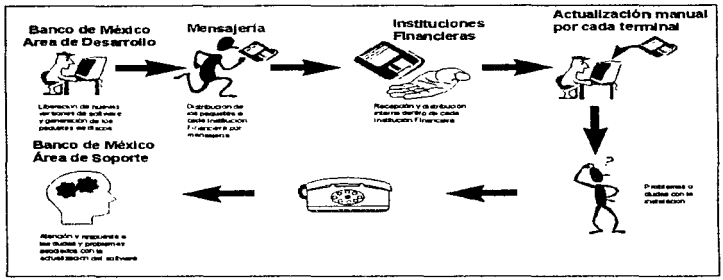


Fig 2.2 Esquema actual de distribución y actualización de software.

Al liberarse una nueva versión de alguno de los sistemas, el personal del Banco de México origina varios juegos de discos de la última versión del sistema, éstos son distribuídos a las instituciones financieras para que realicen manualmente la instalación del software.

Este proceso presenta algunos inconvenientes tales como la poca confiabilidad en la distribución de los discos, ya que la integridad de la información en el paquete no está garantizada porque puede sufrir pérdida o corrupción durante la transportación; además de la falta de control de licencias instaladas en cada institución y las implicaciones derivadas de una mala instalación.

2.2 Finalidad del Sistema

La finalidad del sistema por desarrollar, consiste en construir una herramienta tal que permita al Banco de México tener control sobre la distribución y actualización de los sistemas en forma más confiable. La cual debe cubrir los siguientes aspectos:

- La herramienta deberá estar controlada por el Banco de México.
- La distribución del software deberá ser lo más confiable y eficiente posible.
- Optimizar el tiempo entre la liberación y la actualización del software en las terminales.
- Minimizar la intervención humana en la instalación del software.
- Permitir un control de versiones distribuidas.
- Reducir los recursos y costos involucrados en el esquema actual.

2.3 Alcance y Objetivo del Sistema

El sistema propuesto deberá ser capaz de soportar el entorno actual, que cuenta con 277 terminales en 107 instutuciones financieras en toda la República Mexicana, con una tasa de crecimiento aproximado de 50 terminales al año. Además de que la herramienta contará con una vida útil mínima de 5 años.

Se sistematizará el proceso actual, desde la generación de los juegos de discos hasta la actualización en las terminales remotas con el propósito de proporcionar a las instituciones financieras un sistema de distribución y actualización que permita la transferencia e instalación de aplicaciones en sus terminales de forma práctica y oportuna minimizando la utilización de recursos humanos y materiales.

2.4 Soluciones Alternativas

Como consecuencia del objetivo antes planteado, se proponen las siguientes soluciones:

2.4.1 Propuesta 1

Desarrollo de un sistema implementado en la red interbancaria, la cual es una red de área extensa (como la Internet) que utiliza TCP/IP (Transmission Control Protocol / Internet Protocol), el cual es un estándar de comunicaciones que permite la interconexión entre diversas redes. Dicho sistema tendrá las siguientes características (Figura 2.3):

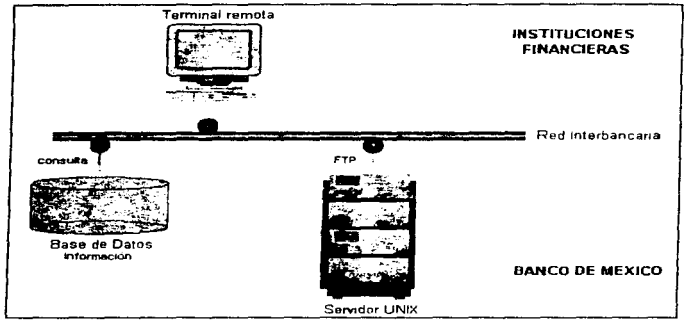


Fig. 2.3 Propuesta 1, con estructura de archivos en UNIX y Base de Datos.

- Se implementará bajo el modelo cliente/servidor, ya que es la fase actual del esquema de sistemas distribuidos y esto permite tener una gran cantidad de información dispersa alrededor del mundo.

- Se diseñará una estructura de archivos en un servidor UNIX donde se almacenarán las aplicaciones a transferir. Estarán compactadas e incluirán todos los archivos y librerías necesarias para su instalación y funcionamiento.

- Deberá contar con una base de datos que contenga la información de las versiones de los sistemas, las terminales, los permisos de acceso, las instituciones, y la ubicación de los archivos compactos de cada aplicación en el servidor UNIX.

- La parte del cliente deberá estar desarrollada en un lenguaje de cuarta generación que permita crear interfaces gráficas tipo windows amigables al usuario, establecer contectividad con la base de datos, y que sea flexible para utilizar programas o librerías externas.

- Debido a que el protocolo de red que se utilizará es TCP/IP, se implementarán algunas librerías que utilicen Windows Sockets para aspectos de comunicación.

- Deberá identificar las terminales que se conectan hacia un servidor de datos en el banco central del país. Esto servirá para manejar privilegios y tener control sobre las versiones de aplicaciones que puede transferir e instalar cada terminal, así como para llevar un histórico y generar estadísticas.

- Realizará la transferencia de aplicaciones desde el servidor UNIX hasta la terminal que solicitó el sistema. Esta transferencia se hará vía FTP (File Transfer Protocol) y para esto se tendrá que hacer uso de Winsock, que es el conjunto de especificaciones o estándares para crear aplicaciones TCP/IP para Windows. La programación de esta transferencia se desarrollará en un lenguaje que nos permita crear librerías para poder utilizar los Windows Sockets.

- Realizará la instalación de la aplicación en forma automática. Para esto necesitamos un programa externo que nos permita ejecutar algunos comandos de Windows para hacer copias de archivos, respaldos, configuración del sistema y preparar el ambiente para ejecutar las aplicaciones.

- Se podrá ejecutar tanto en Windows 3.x como Windows 95.

2.4.2 Propuesta 2

Desarrollo de un esquema de tres capas, cumpliendo los objetivos de la propuesta 1 pero basada en el diagrama de figura 2.4, con las siguientes diferencias:

- Se tendría un servidor entre el cliente y la base de datos, el cual debe controlar todos los procesos en ambiente UNIX, tales como caídas de procesos, selección de la solicitud adecuada y envío de la respuesta al cliente.

- Se eliminaría el uso de FTP (File Transfer Protocol) y se utilizaría un servicio propietario, para el envío de la versión.

- En la base de datos se tendrían todas las características necesarias para que el servidor pueda seleccionar la actualización adecuada.

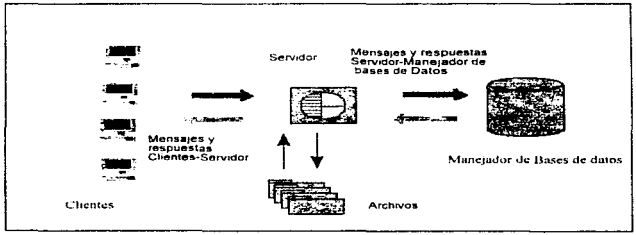


Fig. 2.4 Propuesta 2, con servidor y transferencia de archivos propietarios.

2.4.3 Propuesta 3

Desarrollo de un sistema en Intranet con las siguientes características:

- Implementación con un lenguaje y en una plataforma de red orientados a objetos.
- Contará con una estructura de archivos en UNIX donde se almacenarán los sistemas a transferir en forma compacta.
- Del mismo modo que en las propuestas anteriores, deberá tener una base de datos para la administración y el control de versiones.
- Generación de páginas dinámicas de acuerdo a privilegios y peticiones del usuario.
- Realizará la instalación con un programa externo para Windows.

2.5 Solución Recomendada

En está sección realizaremos el análisis de cada una de las propuestas antes descritas así como del procedimiento actual. A fin de poder seleccionar la solución más recomendable.

Procedimiento Actual

El procedimiento actual basa su costo operativo, principalmente en los siguientes puntos :

- Desplazamiento de personal
- Generación de papelería
- Soporte técnico del Banco de México
- Soporte técnico de la Institución

Estos puntos se aplican cada vez que, una de las instituciones financieras, que requierá la actualización de su software, en cada una de las nuevas versiones liberadas por el Banco de México.

Una de las situaciones a considerar en este método, es que cuando una versión de un sistema como SPEUA pudiese fallar, el tiempo de la instalación de una nueva copia de software implica una pérdida de tiempo importante, ya que cada institución sólo maneja una copia de software, y esto da lugar a un desplazamiento con el Banco de México, dando lugar a un costo por intereses y multas para la institución financiera, al no poder transferir fondos vía electrónica. Si sabemos de antemano que el flujo de cada operación de SPEUA es como mínimo de $80,000.00 pesos, el interés que provoca estas transacciones al no transferirlo oportunamente genera pérdidas significativas para la institución.

Como se anotó anteriormente, se encuentran costos altos en la preparación y distribución del software, pudiendo definir además los siguiente puntos:

Ventajas:

- El método cumple el objetivo de la distribución y actualización de software, aunque no con el control que se desea.

Desventajas:

- Costo alto, por la preparación de la versión para cada institución financiera que requiera una copia del software.
- Retardo en la instalación del software, provocado por el desplazamiento de personal y posibles fallas de la papelería distribuida.
- Poca confiabilidad en el control de versiones distribuidas.
- Posibilidad de copias no permitidas del software distribuido.
- Alta intervención humana, puede provocar errores en la instalación.

Para poder evaluar en forma equitativa los modelos de los proyectos estableceremos los mismos costos según la tabla 2.1, para un análisis de recuperación de la inversión a una tasa lo más viable posible.

CONCEPTO	COSTO PESOS
SALARIOS 1 LÍDER DE PROYECTO 2 ANALISTAS 3 PROGRAMADORES	160,000
LICENCIAS DE SOFTWARE BASE DE DATOS FRONT END INTERFACES DE COMUNICACIONES	30,000
HARDWARE NECESARIO MEMORIA ESPACIO DE DISCO	50,000
SERVIDOR DE SERVICIO O PARA WWW SEGÚN SEA EL CASO	100,000
TOTAL	404,000

Tabla 2.1 Costos estimados para el análisis del desarrollo del proyecto.

Estos costos son estimados para uso exclusivo del análisis de costos. Se estableció en un sueldo de 4,000 pesos para cada programador, un sueldo de 6,000 pesos por analista y un sueldo de 8,000 por líder de proyecto en tiempo de ocupación

aproximado. El proyecto está estimado en un tiempo de desarrollo de 7 meses, a un flujo de capital entrante mensual de 55,000 pesos, y a un interes anual del 19%, tomado esto como base para evaluar todas las propuestas. El análisis que se realizará, será de tiempo de recuperación de la inversión hecha, que es importante para determinar si la inversión es recuperable o no, también se determinará el valor presente neto (VPN), para determinar si es una inversión aprovechable, así como el cálculo de la tasa interna de rendimiento (TIR) que indica de forma más clara cuál es la maxima tasa que puede llegar a dar un proyecto de inversión.

Los cálculos y conclusiones están basados en los conceptos y fórmulas de valor presente neto, cálculo de la tasa interna de rendimiento y periodo de recuperación, explicados en el apéndice A de este trabajo.

Propuesta 1

En la propuesta 1 podemos definir que su metodología se implanta en la red financiera ya existente, por la cual los clientes ya tienen contacto con el Banco de México. El costo operativo de esta propuesta se encuentra principalmente en los siguientes puntos :

- Preparación de versión.
- Transmisión de información.
- Soporte operativo Banco de México
- Soporte operativo Institución Financiera.

Los últimos tres puntos se aplican para cada institución financiera, que requieran la actualización de su software, en cada una de las nuevas versiones liberadas por el Banco de México; con excepción del primer punto que sólo requiere una preparación general para todas las instituciones.

Los costos de inversión y recuperación en tiempo para esta propuesta son los siguientes, tomando en cuenta que el monto de inversión para esta propuesta no requiere otro servidor, sino solo memoria y espacio adicional en disco de los servidores ya existentes:

FLUJO MENSUAL ENTRANTE	55,000		
INVERSION TOTAL EJERCIDA	(240,000)	VALOR ACTUAL INVERSIÓN	AHORRO
TASA DE INTERÉS MENSUAL	1.58%		
VALOR PRESENTE NETO	**121,731**	(304,000)	361,731
TASA INTERNA DE RETORNO	**51%**		
PERÍODO DE RECUPERACIÓN DE LA INVERSIÓN	**4.36**	MESES	

Ventajas:

- Distribución de Software confiable y eficiente.
- Optimización del tiempo de liberación.
- Reducción de recursos humanos.
- Permite el control de versiones distribuidas.
- Reducción de recursos materiales.
- Existencia de red financiera.

Desventajas:

- Aumento del tráfico en la red financiera.
- Dependencia de la red financiera.

Propuesta 2

La propuesta 2 contempla la implantación de un sistema que sea el que atienda las solicitudes, implementando un nuevo servidor, basando su costo operativo, principalmente, en los siguientes puntos :

- Preparación de versión.
- Transmisión de información.
- Soporte operativo Banco de México.
- Soporte especializado de mantenimiento al nuevo servidor.
- Soporte operativo Institución Financiera.

Los últimos cuatro puntos aplican para cada institución financiera que requieran la actualización de su software, en cada una de las nuevas versiones liberadas por el Banco de México; con excepción del primer punto que sólo requiere una preparación general para todas las instituciones.

Los costos de inversión y recuperación en tiempo para esta propuesta son los siguientes, tomando en cuenta que se requiere para esta propuesta un servidor adicional, sin necesidad de crecer en memoria y disco otros servidores:

FLUJO MENSUAL ENTRANTE	55,000		
INVERSION TOTAL EJERCIDA	(290,000)	VALOR ACTUAL	
		INVERSIÓN	AHORRO
TASA DE INTERÉS MENSUAL	1.58%		
VALOR PRESENTE NETO	**71,731**	(354,000)	361,731
TASA INTERNA DE RETORNO	**25%**		
PERÍODO DE RECUPERACIÓN DE LA INVERSIÓN	**5.27**	MESES	

Ventajas:

- Distribución de software confiable y eficiente.
- Optimización del tiempo de liberación.
- Reducción de recursos humanos.
- Permite el control de versiones distribuidas.
- Reducción de recursos materiales.
- Existencia de red financiera.

Desventajas:

- Aumento del tráfico en la red financiera.
- Dependencia de la red financiera.
- Mantenimiento del nuevo servidor de UNIX.

Propuesta 3

La propuesta 3 está basada en la red de Intranet, que actualmente existente para las instituciones financieras, encontrándose el costo operativo, principalmente, en los siguientes puntos :

- Preparación de versión.
- Transmisión de información.
- Soporte operativo Banco de México.
- Soporte especializado de mantenimiento del servidor WEB.
- Soporte operativo Institución Financiera.

Los últimos cuatro puntos se aplican para cada institución financiera que requieran la actualización de su software, en cada una de las nuevas versiones liberadas por el Banco de México, con excepción del primer punto que sólo requiere una preparación general para todas las instituciones.

Los costos de inversión y recuperación en tiempo para esta propuesta son los siguientes, tomando en cuenta que se requiere para esta propuesta un servidor adicional, sin necesidad de crecer en memoria y disco otros servidores:

FLUJO MENSUAL ENTRANTE	55,000	VALOR ACTUAL	
INVERSION TOTAL EJERCIDA	(290,000)	INVERSION	AHORRO
TASA DE INTERÉS MENSUAL	1.58%		
VALOR PRESENTE NETO	**71,731**	(354,000)	361,731
TASA INTERNA DE RETORNO	**25%**		
PERÍODO DE RECUPERACIÓN DE LA INVERSIÓN	**5.27**	MESES	

Ventajas:

- Optimización del tiempo de liberación.
- Reducción de recursos humanos.
- Facilidad de acceso a la página de Internet
- Reducción de recursos materiales.
- Existencia de red financiera.

Desventajas:

- Aumento del tráfico en la red financiera.
- Dependencia de la red financiera.
- Mantenimiento constante del Servidor de WEB.
- Bajo control de transmisión de archivos.

Conforme a lo antes expuesto, se observa que todas las opciones cumplen el fin señalado, pero la solución más viable a su realización y desarrollo en estos momentos es la propuesta número 1, ya que cumple con la mayoría de los requisitos solicitados por el Banco de México, además de cumplir con el costo más bajo, y con los estudios de cálculo de recuperación de inversión mejores, en comparación a las otras dos propuestas.

2.6 Plan de Desarrollo

Expuesto lo anterior y una vez seleccionada la propuesta a realizar, estamos en la posibilidad de generar un plan de desarrollo del sistema, el cual deberá establecer las tareas, los responsables y las dependencias correspondientes para llevar a cabo la implementación y utilización del sistema.

El plan propuesto para el desarrollo se muestra en la tabla 2.2, en la cual se exponen los puntos principales para la implementación del sistema.

Tareas

Las tareas pueden considerarse como las actividades definidas de un plan de desarrollo, en donde cada actividad será enlistada en orden de ejecución, además deberá contemplar el tiempo estimado y los recursos necesarios para lograr el objetivo de la actividad. Cuando surja la necesidad, las tareas pueden dividirse en subtareas, con el fin de definir tareas específicas de desarrollo.

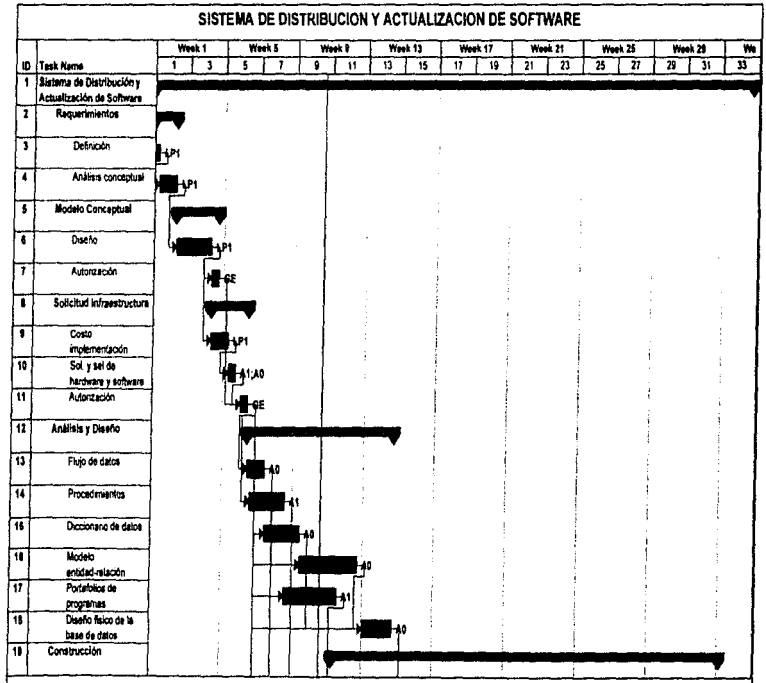


Tabla 2.2 Plan de desarrollo.

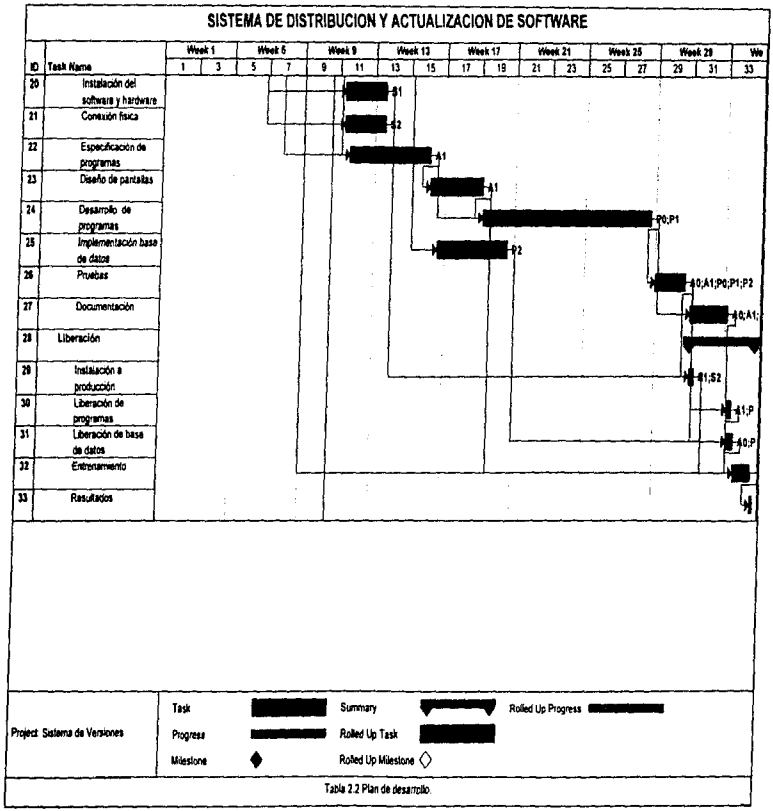


Tabla 2.2 Plan de desarrollo.

En este trabajo, hemos dividido el plan de actividades en 6 tareas principales, para el desarrollo del proyecto

- Requerimientos.
- Modelo conceptual
- Solicitud de infraestructura
- Análisis y diseño del sistema
- Construcción
- Liberación

Cada una de estas actividades define subtareas que hacen que el sistema quede totalmente definido.

Las subtareas están descritas en el plan de trabajo, y mencionan el tiempo de ejecución en días, así como la dependencia y los recursos necesarios para esta actividad.

Recursos

Los recursos, son el poder humano necesario para la implementación del sistema, éstos son descritos y usados en diferentes actividades, ya sea en forma simultáneas o paralelas que es un nivel de trabajo diario de porcentaje de tareas (50% actividad uno y 50% actividad dos, o bien a tareas dependientes, es decir, ejecutar y concluir una tarea y empezar la actividad dependiente de la que concluyó. Podemos mencionar que hay actividades aisladas, pero son igualmente importantes para la conclusión del sistema.

En el caso de nuestro plan se asignan los siguientes recursos:

1. Líder de proyecto "LP1", es el responsable del proyecto a realizar.
2. Analista 1 "A1", apoyará el análisis de los programas de usuario y comunicaciones correspondientes.
3. Analista 2 "A2", apoyará el análisis de la interacción con la base de datos, así como las utilerías para la explotación de éstos.
4. Programador "P1", se encargará del desarrollo de tareas específicas de la parte de usuario ("front-end").
5. Programador "P2", será el encargado del desarrollo de tareas específicas a la parte comunicaciones, transporte y validación de datos de la red.
6. Programador "P3", realizará las interfaces a la base de datos para el uso de éstos en los ambientes antes descritos a los otros programadores, éste es el especialista en los ambientes de bases de datos.
7. Soporte técnico 1 y 2 "S1 y S2", son los encargados de implementar la parte de hardware requerida para la ejecución del sistema .

Dependencia

La dependencia indica que una tarea antes de ser iniciada requiere que una actividad haya sido concluida. Es decir que las actividades tienen un previo y un después que debe de cumplirse para la conclusión del plan en forma satisfactoria.

La dependencia por lo general es representada por flechas de flujos de tareas, las cuales indican de donde surgen y a que tareas despiertan.

Descripción

A continuación describiremos cada una de las tareas en las que se fundamenta el plan de desarrollo para la elaboración completa del sistema:

Definición. En esta parte se define con claridad los objetivos y alcance del problema a resolver.

Análisis Conceptual. Aquí se realiza un análisis general de la solución en sus componentes principales.

Diseño. Realizamos el diseño de la idea general de la propuesta a resolver tomando en cuenta todas las entidades posibles.

Autorización diseño. Esta es una tarea en la que el líder de proyecto tiene que pedir la autorización justificando que se trata de un proyecto rentable y necesario.

Costo implementación. Aquí analizará el costo del presupuesto solicitado para los recursos humanos.

Selección y solicitud de hardware y software necesario. Aquí una vez que se selecciona el hardware y software a utilizar, se presupuesta el hardware adicional a comprar así como el software necesario de compra o solicitud de licencia.

Autorización de presupuesto. Se solicita que se autorice el presupuesto para poder comenzar el desarrollo formal del sistema.

Análisis flujo de datos. Se explican las entidades y el flujo de datos que aplicaría en el nuevo sistema.

Análisis procedimientos. Se establecen los procedimientos nuevos en los que el usuario se internará con el nuevo sistema.

Diccionario de datos. Esta es la base de datos de información necesaria para la definición en cualquier sistema.

Modelo entidad relación. Esta es la etapa de definición de cada entidad y la relación existente entre éstas (análisis de bases de datos).

Portafolios de programas. Se establecerán los módulos, entidades, utilerías que deberán generarse para el desarrollo del sistema.

Diseño físico de la base de datos. Aquí se definirá la estructura final y física de la base de datos que usará el sistema.

Instalación de hardware y software. El personal encargado deberá instalar el software solicitado y verificar que no se encuentren problemas en su uso.

Conexiones físicas. El personal encargado realizará las instalaciones de prueba y de producción necesarias para que el sistema funcione.

Especificación de programas. Define perfectamente cada uno de los programas del portafolios de programas.

Diseño de pantallas. Define las pantallas a detalle, las cuales serán necesarias para el desarrollo de los programas.

Desarrollo de programas. Aquí los programadores realizan todos los componentes del sistema.

Implementación de la base de datos. Se activa el uso y los accesos a la base de datos del sistema.

Pruebas. Una vez desarrollado el sistema, éste es probado en cada uno de sus componentes.

Documentación. Se genera la documentación del sistema como respaldo de la información referente.

Instalación. Se realizan los últimos ajustes en hardware y software de cada aplicación.

Liberación de programas. Se traslada a producción los programas de sistema desarrollado.

Liberación de la base de datos. Se modifica o traslada la base de datos de producción para el nuevo sistema.

Entrenamiento. Se capacita a cada centro operativo que requiera el uso de este nuevo sistema.

Resultados. Es importante establecer si se cumplió o no la meta establecida, así como pensar que mejoras podrían hacerse posteriormente al nuevo sistema implementado.

2.7 Presentación de Resultados

Como resultado de este estudio, podemos concluir que el método actual tiene desventajas, sobre todo por el ámbito de seguridad, por la parte de costos, es importante mencionar que el método actual funciona; sin embargo, es necesario se apliquen nuevos métodos basados en la tecnología existente para optimar el modelo actual, cuando se presente la distribución y actualización de nuevas versiones de los sistemas que interactúan con el Banco de México.

Las propuestas de solución están basadas sobre estructuras posibles ya existentes en la institución, todas cumplen el objetivo final, aunque cada una tiene un entorno totalmente diferente, gracias a esto la decisión por el estudio de recuperación

de costos y el análisis de ventajas y desventajas, son los que hacen que la selección de la propuesta sea mas cuidadosa y confiable.

Una vez seleccionada la propuesta a implementar, se procedio a realizar el estudio de un plan de actividades, el cual, con una duración aproximada de 7 meses desde su inicio, sugerimos llevar a cabo, para el desarrollo del proyecto.

Expuesto lo anterior, nos adentraremos al análisis y diseño de la propuesta con el fin de poder desarrollar los procesos involucrados en la distribución y actualización de software tomando en cuenta lo planteado en este capítulo.

Capítulo
TRES

ANÁLISIS Y DISEÑO

En este capítulo realizaremos el análisis del sistema basado en los resultados del estudio de factibilidad llevado a cabo en el capítulo anterior. Con base en el análisis de la información procederemos a realizar el diseño del sistema.

El objetivo de esta fase es determinar qué y cómo se debe hacer para resolver el problema.

Plantearemos para empezar un modelo lógico del sistema por medio de un diagrama de flujo y un diccionario de datos, para obtener así un diseño físico y generar el portafolio de programas.

3.1 Metodología de Procesos

Las metodologías de análisis combinan procedimientos sistemáticos con una notación única para analizar los dominios de información y funcionalidad de un problema de software; suministran un conjunto de reglas para subdividir el problema y definen una forma de representación para las visiones lógicas y físicas. En esencia, los métodos de análisis de software facilitan al ingeniero de sistemas aplicar principios de análisis fundamentales, dentro del contexto de un método bien definido.

La metodología utilizada para el análisis y diseño del sistema es la metodología de Chris Gane y Trish Sarson.

Esta metodología es parte de un método de desarrollo de software en el cual un problema es descompuesto en un flujo de datos.

El método es enfocado a algunos aspectos de software como el flujo de datos, control de flujos, relación de módulos, definición de objetos.

Describiendo la metodología, mostramos las herramientas de apoyo para el análisis y diseño que ésta utiliza.

Herramientas de análisis estructurado

Las herramientas de análisis consisten en una serie de métodos que ayudan a capturar el modelo del sistema que desea el usuario.

Los modelos, en su mayoría, son representaciones en papel del futuro sistema, abstracciones de lo que al final será una combinación de Hardware y Software de computadora. La razón de construir modelos es el enfatizar ciertas propiedades críticas del sistema. Esto permite comunicarse con el usuario de una manera enfocada, sin distraer con asuntos y características ajenas al sistema. En caso de no haber captado en forma correcta los requerimientos del usuario o que estos hayan cambiado en el tiempo, es posible realizar los cambios pertinentes o desechar el modelo y hacer uno nuevo. Esto es importante, pues es necesario eliminar a lo posible la creación de un producto final que no sea aceptable, pues se corre el riesgo de tener que realizar cambios en la etapa de construcción del sistema, la cual es por experiencia altamente costosa.

Los puntos de enfoque durante el análisis y utilización de herramientas con el mismo fin deben de ser:

- Concentrarse en las propiedades importantes del sistema y al mismo tiempo restar atención a otras menos importantes.
- Discutir cambios y correcciones de los requerimientos del usuario, a bajo costo y con el riesgo mínimo.
- Verificar que el análisis haya captado correctamente el ambiente del usuario y que lo haya respaldado con información documental para que durante el diseño del sistema y construcción del mismo sea factible.

Las herramientas de análisis son básicamente gráficas y textuales. Las gráficas proporcionan una manera fácil de leer para que en la presentación al usuario se muestren los componentes principales, al igual que las conexiones o relaciones entre los mismos. Las herramientas son.

- Diagrama de procesos (Flujo de datos).
- Diccionario de datos.
- Especificación de procesos.

Diagrama de procesos (Flujo de datos)

El diagrama de procesos es la herramienta que nos permite conocer el proceso de una determinada organización, así como los datos que viajan a través de ellos. El aspecto de proceso de un sistema ciertamente es algo importante de modelar y de verificar con el usuario. Los puntos a resolver son:

- ¿ Cuáles son las funciones que debe desempeñar el sistema ?
- ¿ Cuáles son las interacciones entre dichas funciones ?
- ¿ Qué transformaciones debe llevar a cabo el sistema ?
- ¿ Qué entradas se transforman en que salidas ?
- ¿ Qué tipo de labor debe de realizar el sistema ?
- ¿ En dónde obtiene la información para llevar a cabo dicha labor ?
- ¿ Dónde entrega los resultados de su labor ?

El diagrama de procesos debe cubrir cada uno, y en conjunto, los puntos mencionados, para garantizar la correcta comprensión del sistema. Para esto se apoya en diferentes elementos:

Procesos: Se representan por rectángulos con puntas redondeadas y una línea superior donde se escribe el número consecutivo correspondiente de proceso en el diagrama. Estos elementos representan las diversas funciones individuales que el sistema lleva a cabo. Los procesos transforman entradas en salidas.

Flujos: Se representan por líneas con dirección o doble dirección con flechas. Estos elementos muestran las conexiones entre los procesos (funciones del sistema) y representan la información que dichos procesos requieren como entrada o la información que generan como salida.

Almacenamiento de datos: Se representan como dos líneas horizontales paralelas, con una pequeña caja del lado izquierdo, donde se escribe el número consecutivo correspondiente de almacenamiento de datos en el diagrama. Muestran colecciones o conjunto de datos que el sistema debe recordar por un periodo de tiempo.

Entidades externas: Se representan mediante rectángulos con puntas en ángulo recto. Muestran los agentes externos al sistema de los cuales fluye información tanto hacia afuera como hacia adentro. Típicamente se trata de individuos, o grupos de individuos (clientes, empresas, departamentos) con los cuales se mantiene una relación la cual puede ser manual o asistida por computadora.

En la figura 3.1 se muestra la simbología utilizada para los elementos mencionados.

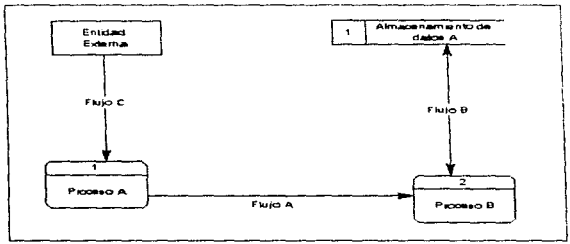


Fig. 3.1 Símbolos de la Metodología.

Descomposición de procesos: Junto a la definición de procesos, es posible realizar un análisis más a detalle de las actividades y flujos de información dentro de un mismo proceso, como se muestra en la figura 3.2, donde se presenta la descomposición del Proceso A. Esto significa tener niveles de procesos, los cuales deberán respetar los flujos de entradas y salidas de tal manera que sea consistente el modelo a través de los diferentes niveles de descomposición.

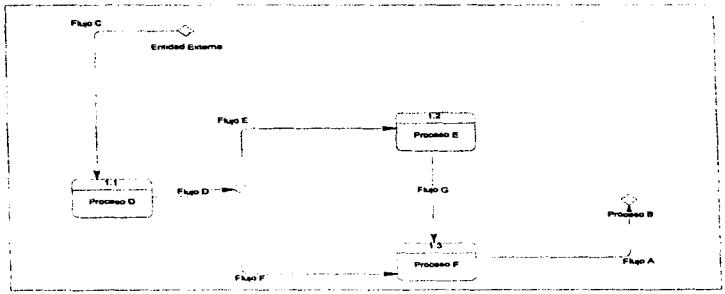


Fig. 3.2 Descomposición del proceso A.

Aun cuando el diagrama de procesos proporciona una visión bastante conveniente de los componentes funcionales del sistema, no da detalles de éstos. Para mostrar detalles acerca de qué información se transforma y de cómo se transforma, se ocupan dos herramientas textuales de modelado adicionales.

Dicccionario de datos: El diccionario de datos contiene las definiciones de todos los datos mencionados en el diagrama de procesos. Los datos compuestos se definen en términos de sus componentes; los elementos se definen en términos del significado de cada uno de los valores que puede asumir. Por lo tanto, el diccionario de datos está compuesto de definiciones de flujo de datos, archivos y datos usados en los procesos.

Especificación de procesos: La especificación de procesos contiene la descripción del proceso, métodos y procedimientos que permiten transformar los flujos de información de entrada en flujo de información de salida. Generalmente en tal especificación se toman en cuenta tanto los flujos de entrada como de salida, junto con los procesos internos y los flujos de información internos del proceso.

Las metodologías de análisis de requerimientos suministran un enfoque sistemático para el análisis de problemas. Aunque cada método tiene un conjunto único de procedimientos y simbología, todos dan los mecanismos para establecer y representar el dominio de la información, particionar el dominio funcional y modelar los procedimientos tanto en el mundo físico como en el lógico.

3.2 Diagrama de Flujo de Datos

Gráfica del proceso principal

Una vez definida la metodología procederemos a realizar el análisis del sistema.

El proceso principal se presenta en el diagrama de la figura 3.3. En ella se pueden identificar 4 procesos importantes, que son:

Lista de procesos

Nombre	Código
Tramitar licencias	TRAMITAR_LICENCIAS
Configurar equipo	CONFIGURAR_EQUIPO
Registrar licencias autorizadas	REGISTRAR_LICENCIAS_AUTORIZADAS
Instalar sistemas	INSTALAR_SISTEMAS

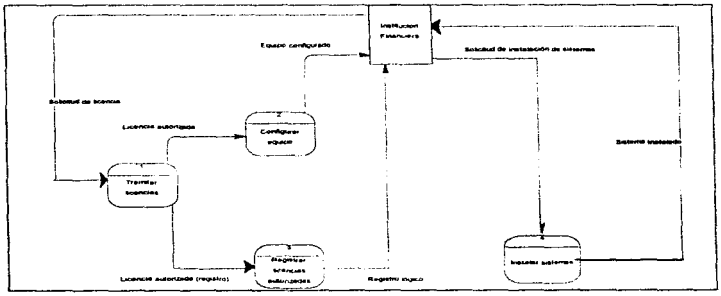


Fig. 3.3 Proceso principal

Gráficas de los procesos:

Proceso Registrar licencias autorizadas

El proceso de registro de licencias comprende la actualización de información en la base de datos y se descompone en varios procesos como se muestra en la figura 3.4.

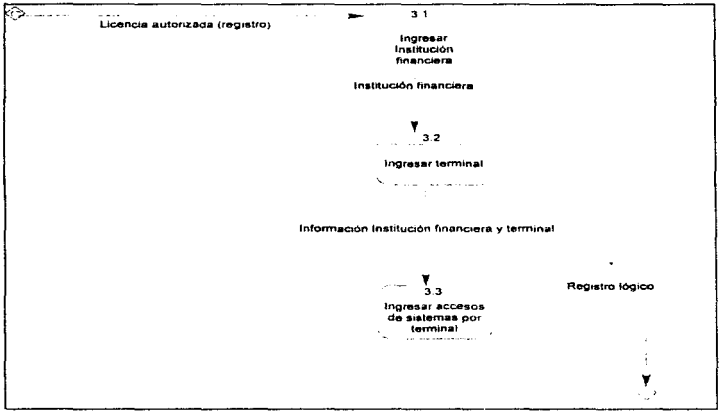


Fig. 3.4 Proceso registrar licencias autorizadas.

Proceso Instalar sistemas

En este proceso se lleva a cabo la petición, transferencia e instalación del sistema, validando una serie de parámetros de seguridad como se muestra en la figura 3.5.

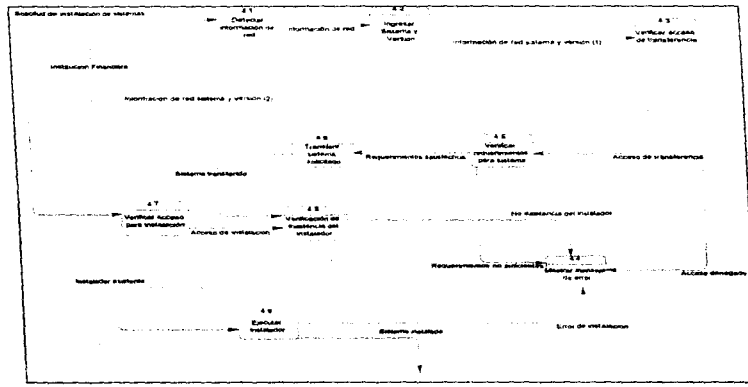


Fig. 3.5 Proceso instalar sistemas.

3.3 Procedimientos

A continuación se explicará cada uno de los procesos involucrados en el sistema.

Proceso Tramitar licencias

Nombre:	Tramitar licencias
Código:	TRAMITAR_LICENCIAS
Número:	1
Nivel inferior:	No

Descripción del proceso

La solicitud la recibe la Gerencia de Trámite.

Esta verifica datos y firmas, y la envía a la oficina de Registros.

Lista de referencias del proceso

Conexión desde	Conexión a	Fuente	Destino
Licencia autorizada	Configurar equipo (Proceso)	X	
Licencia autorizada (registro)	Registrar licencias autorizadas (Proceso)	X	
Solicitud de licencia	Institucion Financiera (Entidad externa)		X

Proceso Configurar equipo

Nombre:	Configurar equipo
Código:	CONFIGURAR_EQUIPO
Número:	2
Nivel inferior:	No

Descripción del proceso

Instalación del protocolo de comunicación TCP/IP y conexión a la red financiera.

Instalación del cliente del sistema de Versiones.

Preparación de ambiente para ejecutar el sistema.

Lista de referencias del proceso

Conexión desde	Conexión a	Fuente	Destino
Equipo configurado	Institución Financiera (Entidad externa)	X	
Licencia autorizada	Tramitar licencias (Proceso)		X

Proceso Registrar licencias autorizadas

Nombre:	Registrar licencias autorizadas
Código:	REGISTRAR_LICENCIAS_AUTORIZADAS
Número:	3
Nivel inferior:	No

Descripción del proceso

Ingresar a la base de datos la información necesaria para operar aplicaciones del Banco de México de acuerdo a la institución financiera y terminal asociada.

Para el acceso a los sistemas, es necesario que inicialmente se verifique la información referente a la institución, en caso de que sea de recién creación en la base de datos.

Con la información de la institución dada de alta, se procede a ingresar la información referente a la terminal.

Se especifican los derechos para actualización de sistemas a la terminal en la base de datos.

Lista de referencias del proceso

Conexión desde	Conexión a	Fuente	Destino
Licencia autorizada (registro)	Tramitar licencias (Proceso)		X
Registro lógico	Institución Financiera (Entidad externa)	X	

Lista de SubProcesos del proceso Registrar licencias autorizadas

Nombre	Código
Ingresar institución financiera	INGRESAR_INSTITUCION_FINANCIERA
Ingresar terminal	INGRESAR_TERMINAL
Ingresar accesos de sistemas por terminal	INGRESAR_ACCESOS_DE_SISTEMAS_POR_TERMINAL

SubProceso Ingresar Institución financiera

Nombre:	Ingresar Institución financiera
Código:	INGRESAR_INSTITUCION_FINANCIERA
Número:	3.1
Nivel inferior:	No

Descripción del SubProceso

Se da de alta en la base de datos toda la información de la institución financiera que solicitó la licencia de la terminal.

Lista de referencia del SubProceso

Conexión desde	Conexión a	Fuente	Destino
Institución financiera	Ingresar terminal (SubProceso)	X	
Licencia autorizada (registro)	Licencia autorizada (registro) (Flujo de datos)		X

SubProceso Ingresar terminal

Nombre:	Ingresar terminal
Código:	INGRESAR_TERMINAL
Número:	3.2
Nivel Inferior:	No

Descripción del SubProceso

Se da de alta en la base de datos toda la información de la terminal que se conectará a la red financiera para actualizar aplicaciones a través del sistema de actualizador de versiones.

Lista de referencia del SubProceso

Conexión desde	Conexión a	Fuente	Destino
Información Institución financiera y terminal	Ingresar accesos de sistemas por terminal (SubProceso)	X	
Institución financiera	Ingresar Institución financiera (SubProceso)		X

SubProceso Ingresar accesos de sistemas por terminal

Nombre:	Ingresar accesos de sistemas por terminal
Código:	INGRESAR_ACCESOS_DE_SISTEMAS_POR_TERMINAL
Número:	3.3
Nivel Inferior:	No

Descripción del SubProceso

Se especifica en la base de datos qué sistemas y de qué versión pueden ser actualizados por la terminal.

Esto sirve para llevar un control de seguridad al permitir a las instituciones actualizar o no un sistema.

También permite llevar el control de versiones instaladas para poder obtener reportes estadísticos.

Lista de referencia del SubProceso

Conexión desde	Conexión a	Fuente	Destino
Información Institución financiera y terminal	Ingresar terminal (SubProceso)		X
Registro lógico	Registro lógico (Flujo de datos)	X	

Proceso Instalar sistemas

Nombre:	Instalar sistemas
Código:	INSTALAR_SISTEMAS
Número:	4
Nivel inferior:	No

Descripción del proceso

Se lleva a cabo la transferencia y la instalación del sistema especificado. Para esto se deben verificar aspectos de seguridad y control tales como identificar la terminal conectada a la red financiera, sus derechos para transferir e instalar sistemas y su situación actual.

Lista de referencias del proceso

Conexión desde	Conexión a	Fuente	Destino
Sistema instalado	Institucion Financiera (Entidad externa)	X	
Solicitud de instalación de sistemas	Institucion Financiera (Entidad externa)		X

Lista de subprocesos del proceso Instalar sistemas

Nombre	Código
Detectar información de red	DETECTAR_INFORMACION_DE_RED
Ingresar Sistema y Versión	INGRESAR_SISTEMA_Y_VERSION
Verificar acceso de transferencia	VERIFICAR_ACCESO_DE_TRANSFERENCIA
Mostrar mensaje de error	MOSTRAR_MENSAJE_DE_ERROR
Verificar requerimientos para sistema	VERIFICAR_REQUERIMIENTOS_PARA_SISTEMA
Transferir sistema solicitado	TRANSFERIR_SISTEMA_SOLICITADO
Verificar acceso para instalación	VERIFICAR_ACCESO_PARA_INSTALACION
Verificación de existencia del sistema	VERIFICACION_DE_EXISTENCIA_DEL_SISTEMA
Ejecutar instalación	EJECUTAR_INSTALACION

Subproceso Detectar información de red

Nombre:	Detectar información de red
Código:	DETECTAR_INFORMACION_DE_RED
Número:	4.1
Nivel inferior:	No

Descripción del subproceso

Para el Banco de México es muy importante la seguridad de la información, es por eso que se debe tener un control de acceso al sistema.

Una forma única de identificar cualquier terminal dentro de la red financiera es por medio de la dirección IP.

En esta parte del proceso se detecta la direccón IP de la terminal remota.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Información de red Solicitud de instalación de sistemas	Ingresar Sistema y Versión (Proceso) Solicitud de instalación de sistemas (Flujo de datos)	X	X

Subproceso Ingresar Sistema y Versión

Nombre:	Ingresar Sistema y Versión
Código:	INGRESAR_SISTEMA_Y_VERSION
Número:	4.2
Nivel Inferior:	No

Descripción del subproceso

El usuario hace la solicitud del sistema y versión a actualizar.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Información de red	Detectar información de red (Proceso)		X
Información de red sistema y versión (1)	Verificar acceso de transferencia (Proceso)	X	
Información de red sistema y versión (2)	Verificar acceso para instalación (Proceso)	X	

Subproceso Verificar acceso de transferencia

Nombre:	Verificar acceso de transferencia
Código:	VERIFICAR_ACCESO_DE_TRANSFERENCIA
Número:	4.3
Nivel Inferior:	No

Descripción del subproceso

Con la Dirección IP se determina el número de terminal.

Con el número de terminal se verifica si el Sistema y versión especificados son permitidos para la terminal y si se puede realizar su transferencia.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Acceso denegado	Mostrar mensaje de error (Proceso)	X	
Acceso de transferencia	Verificar requerimientos para sistema (Proceso)	X	
Información de red sistema y versión (1)	Ingresar Sistema y Versión (Proceso)		X

Subproceso Mostrar mensaje de error

Nombre:	Mostrar mensaje de error
Código:	MOSTRAR_MENSAJE_DE_ERROR
Número:	4 4
Nivel inferior:	No

Descripción del subproceso

Despliega mensaje de error.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Acceso denegado	Verificar acceso de transferencia (Proceso)		X
Error de instalación	Ejecutar instalación (Proceso)		X
No existencia del instalador	Verificación de existencia del instalador (Proceso)		X
Requerimientos no suficientes	Verificar requerimientos para sistema (Proceso)		X

Subproceso Verificar requerimientos para sistema

Nombre:	Verificar requerimientos para sistema
Código:	VERIFICAR_REQUERIMIENTOS_PARA_SISTEMA
Número:	4 5
Nivel inferior:	No

Descripción del subproceso

Verifica que la terminal cumpla con los requerimientos necesarios para la transferencia e instalación del sistema de acuerdo a la información obtenida en el proceso anterior.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Acceso de transferencia	Verificar acceso de transferencia (Proceso)		X
Requerimientos no suficientes	Mostrar mensaje de error (Proceso)	X	
Requerimientos satisfechos	Transferir sistema solicitado (Proceso)	X	

Subproceso Transferir sistema solicitado

Nombre:	Transferir sistema solicitado
Código:	TRANSFERIR_SISTEMA_SOLICITADO
Número:	4.6
Nivel inferior:	No

Descripción del subproceso

Se ejecuta la transferencia del sistema solicitado desde un servidor hasta la terminal especificada.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Requerimientos satisfechos	Verificar requerimientos para sistema (Proceso)		X
Sistema transferido	Verificación de existencia del instalador (Proceso)	X	

Subproceso Verificar acceso para instalación

Nombre:	Verificar acceso para instalación
Código:	VERIFICAR_ACCESO_PARA_INSTALACION
Número:	4.7
Nivel inferior:	No

Descripción del subproceso

La terminal pudo haber obtenido el sistema y la versión por otro medio.

Es por eso que se necesita verificar si dicha terminal tiene permisos para la instalación del sistema y versión solicitados.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Información de red sistema y versión (2)	Ingresar Sistema y Versión (Proceso)		X
Acceso de instalación	Verificación de existencia del sistema (Proceso)	X	

Subproceso Verificación de existencia del instalador

Nombre:	Verificación de existencia del sistema
Código:	VERIFICACION_DE_EXISTENCIA_DEL_SISTEMA
Número:	4.8
Nivel inferior:	No

Descripción del subproceso

Verifica que se haya transferido bien y completo el sistema solicitado, en la terminal.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Acceso de instalación	Verificar acceso para instalación (Proceso)		X
No existencia del instalador	Mostrar mensaje de error (Proceso)	X	
Instalador existente	Ejecutar instalación (Proceso)	X	
Sistema transferido	Transferir sistema solicitado (Proceso)		X

Subproceso Ejecutar instalación

Nombre:	Ejecutar instalación
Código:	EJECUTAR_INSTALACION
Número:	4.9
Nivel inferior:	No

Descripción del subproceso

Descompacta el sistema transferido a la terminal y realiza la instalación de la aplicación.

Lista de referencia del subproceso

Conexión desde	Conexión a	Fuente	Destino
Error de instalación	Mostrar mensaje de error (Proceso)	X	
Instalador existente	Verificación de existencia del sistema (Proceso)		X
Sistema instalado	Sistema instalado (Flujo de datos)	X	

3.4 Diccionario de Datos

Elementos de información

Lista de elementos de información

Nombre	Código	Tipos	
actualizar_beta	ACTUALIZAR_BETA	A1	char (1)
actualizar_version	ACTUALIZAR_VERSION	A1	char (1)
colonia_fiscal	COLONIA_FISCAL	VA40	varchar (40)
cp_fiscal	CP_FISCAL	A5	char (5)
descripcion	DESCRIPCION	A70	char (70)
dir_host	DIR_HOST	VA50	varchar (50)
dir_local	DIR_LOCAL	VA50	varchar (50)
direccion_fiscal	DIRECCION_FISCAL	VA80	varchar (80)
direccion_ip	DIRECCION_IP	A18	char (18)
direccion_red	DIRECCION_RED	A18	char (18)
ent_federativa_fiscal	ENT_FEDERATIVA_FISCAL	VA25	varchar (25)
error	ERROR	I	integer
espacio_disco	ESPACIO_DISCO	I	integer
fecha_actual	FECHA_ACTUAL	DT	datetime
fecha_beta	FECHA_BETA	DT	datetime
fecha_suscripcion	FECHA_SUSCRIPCION	DT	datetime
fecha_version	FECHA_VERSION	DT	datetime
host_unix	HOST_UNIX	VA20	varchar (20)
institucion	INSTITUCION	I	integer
localidad_fiscal	LOCALIDAD_FISCAL	VA15	varchar (15)
login_unix	LOGIN_UNIX	VA20	varchar (20)
mem_principal	MEM_PRINCIPAL	I	integer
mem_secundaria	MEM_SECUNDARIA	I	integer
nombre_corto	NOMBRE_CORTO	A15	char (15)
Nombre_Institucion	NOMBRE_INSTITUCION	A70	char (70)
nombre_sistema	NOMBRE_SISTEMA	A70	char (70)
nombre_usuario	NOMBRE_USUARIO	VA50	varchar (50)
obtenible	OBTENIBLE	A1	char (1)
password_unix	PASSWORD_UNIX	VA30	varchar (30)
permiso_beta	PERMISO_BETA	A1	char (1)
rfc	RFC	VA13	varchar (13)
sistema	SISTEMA	A10	char (10)
sistema_operativo	SISTEMA_OPERATIVO	SF	short float
tam_host	TAM_HOST	I	integer
terminal	TERMINAL	SI	short integer
tipo_version	TIPO_VERSION	A1	char (1)
Ubicacion_Terminal	UBICACION_TERMINAL	A70	char (70)
valida	VALIDA	A1	char (1)
version	VERSION	VA15	varchar (15)
version_actual	VERSION_ACTUAL	VA15	varchar (15)
version_permitida	VERSION_PERMITIDA	A1	char (1)

Elemento de información actualizar_beta

Nombre:	actualizar_beta
Código:	ACTUALIZAR_BETA
Etiqueta:	S/N Determina el rango de versiones beta que puede tomar
Dominio:	S, N
Tipo:	A
Longitud:	1

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información actualizar_version

Nombre:	actualizar_version
Código:	ACTUALIZAR_VERSION
Etiqueta:	S/N Determina el rango de versiones obtenibles para la terminal
Dominio:	S, N
Tipo:	A
Longitud:	1

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información colonia_fiscal

Nombre:	colonia_fiscal
Código:	COLONIA_FISCAL
Etiqueta:	Colonia fiscal de la institución
Tipo:	VA
Longitud:	40

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información cp_fiscal

Nombre:	cp_fiscal
Código:	CP_FISCAL
Etiqueta:	Código postal fiscal de la institución
Tipo:	A
Longitud:	5

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información descripcion

Nombre:	descripcion
Código:	DESCRIPCION
Etiqueta:	Descripción del error
Dominio:	descripcion
Tipo:	A
Longitud:	70

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Información de red	INFORMACION_DE_RED	Flujo de datos
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Acceso denegado	ACCESO_DENEGADO	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos
No existencia del sistema	NO_EXISTENCIA_DEL_SISTEMA	Flujo de datos
Error de instalación	ERROR_DE_INSTALACION	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos
Instalación no válida	INSTALACION_NO_VALIDA	Flujo de datos

Elemento de información dir_host

Nombre:	dir_host
Código:	DIR_HOST
Etiqueta:	Ruta completa donde se localiza el archivo compacto.exe en el servidor UNIX
Tipo:	VA
Longitud:	50

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos

Elemento de información dir_local

Nombre:	dir_local
Código:	DIR_LOCAL
Etiqueta:	Ruta completa donde se almacenará la versión en el cliente
Tipo:	VA
Longitud:	50

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos

Elemento de información direccion_fiscal

Nombre:	direccion_fiscal
Código:	DIRECCION_FISCAL
Etiqueta:	Dirección fiscal de la institución
Tipo:	VA
Longitud:	80

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información direccion_ip

Nombre:	direccion_ip
Código:	DIRECCION_IP
Etiqueta:	Dirección IP que identifica a la terminal dentro de la red financiera
Tipo:	A
Longitud:	18

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Información de red	INFORMACION_DE_RED	Flujo de datos
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION __1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION __2_	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información direccion_red

Nombre:	direccion_red
Código:	DIRECCION_RED
Etiqueta:	Dirección de la tarjeta de red de la terminal
Tipo:	A
Longitud:	18

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO _	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información ent_federativa_fiscal

Nombre:	ent_federativa_fiscal
Código:	ENT_FEDERATIVA_FISCAL
Etiqueta:	Entidad federativa fiscal de la institucion
Tipo:	VA
Longitud:	25

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información error

Nombre: error
Código: ERROR
Etiqueta: Número de error
Tipo: I
Longitud: 0

Lista de referencias

Nombre	Código	Referencia
Acceso denegado	ACCESO_DENEGADO	Flujo de datos
No existencia del sistema	NO_EXISTENCIA_DEL_SISTEMA	Flujo de datos
Error de instalacion	ERROR_DE_INSTALACION	Flujo de datos
Instalación no valida	INSTALACION_NO_VALIDA	Flujo de datos

Elemento de información espacio_disco

Nombre: espacio_disco
Código: ESPACIO_DISCO
Etiqueta: Mínimo espacio libre requerido para poder ejecutar la aplicación (en KB)
Tipo: I
Longitud: 0

Descripción

Incluye el espacio utilizado para descompactar el archivo compacto.exe

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos

Elemento de información fecha_actual

Nombre: fecha_actual
Código: FECHA_ACTUAL
Etiqueta: Fecha en la cual se solicito cambio de versión
Tipo: DT
Longitud: 0 0

Descripción

Fecha en la cual se solicito cambio de versión.
Su valor inicial debe ser nulo

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información fecha_beta

Nombre:	fecha_beta
Código:	FECHA_BETA
Etiqueta:	Fecha en la cual se solicitó cambio de versión beta
Tipo:	DT
Longitud:	0

Descripción

Fecha en la cual se solicitó cambio de versión beta.

Su valor inicial debe ser nulo

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información fecha_suscripcion

Nombre:	fecha_suscripcion
Código:	FECHA_SUSCRIPCION
Etiqueta:	Fecha a partir de la cual se registro la terminal como perteneciente al sistema
Tipo:	DT
Longitud:	0

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información fecha_version

Nombre:	fecha_version
Código:	FECHA_VERSION
Etiqueta:	Fecha de referencia de la versión
Tipo:	DT
Longitud:	0

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Información de red	INFORMACION_DE_RED	Flujo de datos
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Acceso denegado	ACCESO_DENEGADO	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos
No existencia del sistema	NO_EXISTENCIA_DEL_SISTEMA	Flujo de datos
Error de instalación	ERROR_DE_INSTALACION	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos
Instalación no válida	INSTALACION_NO_VALIDA	Flujo de datos

Elemento de información host_unix

Nombre:	host_unix
Código:	HOST_UNIX
Etiqueta:	Dirección IP del host de UNIX donde se encuentra la aplicación a transferir
Tipo:	VA
Longitud:	20

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos

Elemento de información institución

Nombre: institución
Código: INSTITUCION
Etiqueta: Clave de la institución
Tipo: I
Longitud: 0

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información localidad_fiscal

Nombre: localidad_fiscal
Código: LOCALIDAD_FISCAL
Etiqueta: Localidad fiscal de la institución
Tipo: VA
Longitud: 15

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información login_unix

Nombre: login_unix
Código: LOGIN_UNIX
Etiqueta: Login de acceso al servidor UNIX
Tipo: VA
Longitud: 20

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos

Elemento de información mem_principal

Nombre:	mem_principal
Código:	MEM_PRINCIPAL
Etiqueta:	Tamaño mínimo necesario de memoria principal para el funcionamiento del sistema
Tipo:	I
Longitud:	0

Descripción

Tamaño mínimo necesario de memoria principal para el buen funcionamiento de la aplicación (en KB)

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos

Elemento de información mem_secundaria

Nombre:	mem_secundaria
Código:	MEM_SECUNDARIA
Etiqueta:	Tamaño mínimo necesario de memoria secundaria para el funcionamiento del sistema
Tipo:	I
Longitud:	0

Descripción

Tamaño mínimo necesario de memoria secundaria para el buen funcionamiento de la aplicación (en KB)

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos

Elemento de información nombre_corto

Nombre:	nombre_corto
Código:	NOMBRE_CORTO
Etiqueta:	Nombre corto de la institución
Tipo:	A
Longitud:	15

Lista de referencias

Nombre	Codigo	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Información de red	INFORMACION_DE_RED	Flujo de datos
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Acceso denegado	ACCESO_DENEGADO	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos
No existencia del sistema	NO_EXISTENCIA_DEL_SISTEMA	Flujo de datos
Error de instalación	ERROR_DE_INSTALACION	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos
Instalación no válida	INSTALACION_NO_VALIDA	Flujo de datos

Elemento de Información Nombre_Institucion

Nombre:	Nombre_Institucion
Código:	NOMBRE_INSTITUCION
Etiqueta:	Nombre completo de la institución
Tipo:	A
Longitud:	70

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información nombre_sistema

Nombre:	nombre_sistema
Código:	NOMBRE_SISTEMA
Etiqueta:	Nombre completo del sistema registrado
Dominio:	descripcion
Tipo:	A
Longitud:	70

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información nombre_usuario

Nombre:	nombre_usuario
Código:	NOMBRE_USUARIO
Etiqueta:	Nombre del usuario
Tipo:	VA
Longitud:	50 0

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información obtenible

Nombre:	obtenible
Código:	OBTENIBLE
Etiqueta:	S/N indica si la versión registrada puede ser distribuida
Tipo:	A
Longitud:	1

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_ TERMINAL	Flujo de datos

Elemento de información password_unix

Nombre:	password_unix
Código:	PASSWORD_UNIX
Etiqueta:	Password de acceso al servidor UNIX
Tipo:	VA
Longitud:	30

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos

Elemento de información permiso_beta

Nombre:	permiso_beta
Código:	PERMISO_BETA
Etiqueta:	S/N Indica si puede obtener la versión beta de la aplicación
Tipo:	A
Longitud:	1

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información *rfc*

Nombre:	rfc
Código:	RFC
Etiqueta:	Registro Federal de Contribuyentes de la institución
Tipo:	VA
Longitud:	13

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información sistema

Nombre:	sistema
Código:	SISTEMA
Etiqueta:	Sistema a distribuir
Tipo:	A
Longitud:	10

Lista de referencias

Nombre	Código	Referencia
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos

Elemento de información sistema_operativo

Nombre: sistema_operativo
Código: SISTEMA_OPERATIVO
Etiqueta: Versión mínima de sistema operativo
Tipo: SF
Longitud: 0

Descripción

Versión mínima de sistema operativo en el cliente para que la aplicación funcione correctamente

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos

Elemento de información tam_host

Nombre: tam_host
Código: TAM_HOST
Etiqueta: Tamaño del archivo compacto en bytes
Tipo: I
Longitud: 0

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos

Elemento de información terminal

Nombre: terminal
Código: TERMINAL
Etiqueta: Número de Terminal
Tipo: SI
Longitud: 0

Descripción

Número único que identifica cada terminal que utiliza el sistema

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información tipo_version

Nombre: tipo_version
Código: TIPO_VERSION
Etiqueta: Distingue una versión Beta (B) de una de Producción (P)
Tipo: A
Longitud: 1

Lista de referencias

Nombre	Código	Referencia
Acceso válido	ACCESO_VALIDO	Flujo de datos

Elemento de información Ubicacion_Terminal

Nombre: Ubicacion_Terminal
Código: UBICACION_TERMINAL
Etiqueta: ubicación de la terminal
Tipo: A
Longitud: 70

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos

Elemento de información valida

Nombre:	valida
Código:	VALIDA
Etiqueta:	S/N indica si la versión actual es válida. Si no es válida, no es obtenible
Tipo:	A
Longitud:	1

Lista de referencias

Nombre	Código	Referencia
Licencia autorizada	LICENCIA_AUTORIZADA	Flujo de datos
Licencia autorizada (registro)	LICENCIA_AUTORIZADA__REGISTRO_	Flujo de datos
Información de red	INFORMACION_DE_RED	Flujo de datos
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Acceso denegado	ACCESO_DENEGADO	Flujo de datos
Acceso válido	ACCESO_VALIDO	Flujo de datos
Requerimientos satisfechos	REQUERIMIENTOS_SATISFECHOS	Flujo de datos
Instalación válida	INSTALACION_VALIDA	Flujo de datos
No existencia del sistema	NO_EXISTENCIA_DEL_SISTEMA	Flujo de datos
Error de instalacion	ERROR_DE_INSTALACION	Flujo de datos
Institución financiera	INSTITUCION_FINANCIERA	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos
Solicitud de licencia	SOLICITUD_DE_LICENCIA	Flujo de datos
Instalación no válida	INSTALACION_NO_VALIDA	Flujo de datos

Elemento de información version

Nombre:	version
Código:	VERSION
Etiqueta:	Nombre o nivel de la versión
Tipo:	VA
Longitud:	15

Lista de referencias

Nombre	Código	Referencia
Información de red sistema y versión (1)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__1_	Flujo de datos
Información de red sistema y versión (2)	INFORMACION_DE_RED_SISTEMA_Y_VERSION__2_	Flujo de datos
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos
Solicitud de instalación de sistemas	SOLICITUD_DE_INSTALACION_DE_SISTEMAS	Flujo de datos

Elemento de información version_actual

Nombre:	version_actual
Código:	VERSION_ACTUAL
Etiqueta:	Nombre de la versión que obtuvo en la última ejecución.
Tipo:	VA
Longitud:	15

Descripción

Nombre de la versión que obtuvo en la última ejecución.

Su valor inicial debe ser nulo (NULL)

Lista de referencias

Nombre	Código	Referencia
Información Institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

Elemento de información version_permitida

Nombre:	version_permitida
Código:	VERSION_PERMITIDA
Etiqueta:	U/. (espacio en blanco). /A Determina que versión puede solicitar la terminal
Tipo:	A
Longitud:	1

Descripción

Si en este campo tiene el valor U, la terminal solamente podrá obtener la última versión según le corresponda, en base al campo actualizar_versión; si tiene valor A, la terminal sólo podrá obtener la versión anterior; y si tiene valor " " (blanco) podrá obtener Ultima o Anterior

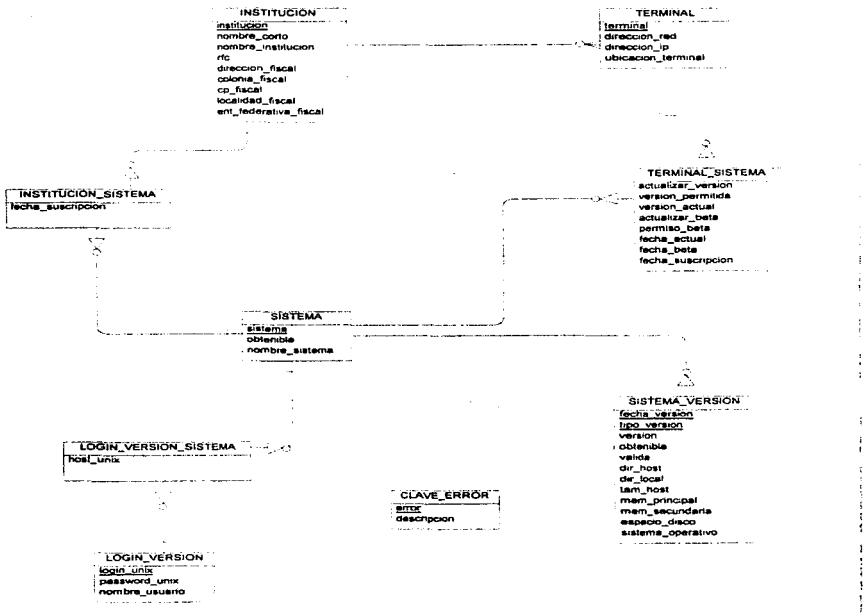
Lista de referencias

Nombre	Código	Referencia
Información institución financiera y terminal	INFORMACION_INSTITUCION_FINANCIERA_Y_TERMINAL	Flujo de datos

3.5 Modelo Entidad-Relación

Gráficas del modelo Entidad - relación

A continuación se muestra el diagrama Entidad - relación de la base de datos del sistema.



Modelo de información

En seguida se especificarán las entidades, atributos y relaciones de la base de datos.

Información

Nombre del proyecto:	Sistema que actualiza y distribuye versiones
Nombre:	Versiones
Código:	Versiones

Dominios:

Lista de dominios

Nombre	Código	Tipo
descripcion	descripcion	A70

Descripción del dominio

Nombre:	descripcion		
Código:	descripcion		
Tipo:	A		
Longitud:	70	**Precisión:**	0

Lista de referencias

Nombre	Atributo	Referencia
CLAVE_ERROR	descripcion	Entidad
INSTITUCION	nombre_institucion	Entidad
SISTEMA	nombre_sistema	Entidad
TERMINAL	ubicacion_terminal	Entidad

Atributos:

Lista de atributos

Nombre	Código	Tipo
actualizar_beta	actualizar_beta	A1
actualizar_version	actualizar_version	A1
colonia_fiscal	colonia_fiscal	VA40
cp_fiscal	cp_fiscal	A5
descripcion	descripcion	A70
dir_host	dir_host	VA50
dir_local	dir_local	VA50
direccion_fiscal	direccion_fiscal	VA80
direccion_ip	direccion_ip	A18
direccion_red	direccion_red	A18
ent_federativa_fiscal	ent_federativa_fiscal	VA25
error	error	I
espacio_disco	espacio_disco	I
fecha_actual	fecha_actual	DT
fecha_beta	fecha_beta	DT
fecha_suscripcion	fecha_suscripcion	DT

Nombre	Código	Tipo
fecha_version	fecha_version	DT
host_unix	host_unix	VA20
institucion	institucion	I
localidad_fiscal	localidad_fiscal	VA15
login_unix	login_unix	VA20
mem_principal	mem_principal	I
mem_secundaria	mem_secundaria	I
nombre_corto	nombre_corto	A15
nombre_institucion	nombre_institucion	A70
nombre_sistema	nombre_sistema	A70
nombre_usuario	nombre_usuario	VA50
obtenible	obtenible	A1
password_unix	password_unix	VA30
permiso_beta	permiso_beta	A1
rfc	rfc	VA13
sistema	sistema	A10
sistema_operativo	sistema_operativo	SF
tam_host	tam_host	I
terminal	terminal	SI
tipo_version	tipo_version	A1
ubicacion_terminal	ubicacion_terminal	A70
valida	valida	A1
version	version	VA15
version_actual	version_actual	VA15
version_permitida	version_permitida	A1

Atributos actualizar_beta

Nombre:	actualizar_beta		
Código:	actualizar_beta		
Etiqueta:	S/N Determina el rango de versiones beta que puede tomar		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos actualizar_version

Nombre:	actualizar_version		
Código:	actualizar_version		
Etiqueta:	S/N Determina el rango de versiones obtenibles para la terminal		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos colonia_fiscal

Nombre:	colonia_fiscal		
Código:	colonia_fiscal		
Etiqueta:	Colonia fiscal de la institución		
Tipo:	VA		
Longitud:	40	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos cp_fiscal

Nombre:	cp_fiscal		
Código:	cp_fiscal		
Etiqueta:	Código postal fiscal de la institución		
Tipo:	A		
Longitud:	5	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos descripcion

Nombre:	descripcion		
Código:	descripcion		
Etiqueta:	Descripción del error		
Domain:	descripcion		
Tipo:	A		
Longitud:	70	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	CLAVE_ERROR	CLAVE_ERROR

Atributos dir_host

Nombre:	dir_host		
Código:	dir_host		
Etiqueta:	Ruta completa donde se localiza el archivo compacto.exe en el servidor UNIX		
Tipo:	VA		
Longitud:	50	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos dir_local

Nombre:	dir_local		
Código:	dir_local		
Etiqueta:	Ruta completa donde se almacenará la versión en el cliente		
Tipo:	VA		
Longitud:	50	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos direccion_fiscal

Nombre:	direccion_fiscal		
Código:	direccion_fiscal		
Etiqueta:	Dirección fiscal de la institución		
Tipo:	VA		
Longitud:	80	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos direccion_ip

Nombre:	direccion_ip		
Código:	direccion_ip		
Etiqueta:	Dirección IP que identifica a la terminal dentro de la red financiera		
Tipo:	A		
Longitud:	18	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL	TERMINAL

Atributos direccion_red

Nombre:	direccion_red		
Código:	direccion_red		
Etiqueta:	Dirección de la tarjeta de red de la terminal		
Tipo:	A		
Longitud:	18	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL	TERMINAL

Atributos ent_federativa_fiscal

Nombre:	ent_federativa_fiscal		
Código:	ent_federativa_fiscal		
Etiqueta:	Entidad federativa fiscal de la institución		
Tipo:	VA		
Longitud:	25	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos error

Nombre:	error		
Código:	error		
Etiqueta:	Número de error		
Tipo:	I		
Longitud:	0	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	CLAVE_ERROR	CLAVE_ERROR

Atributos espacio_disco

Nombre:	espacio_disco
Código:	espacio_disco
Etiqueta:	Mínimo espacio libre requerido para poder ejecutar la aplicacion (en KB)
Tipo:	I
Longitud:	0 **Precisión:** 0

Descripción

Incluye el espacio utilizado para descompactar el archivo compacto.exe

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos fecha_actual

Nombre:	fecha_actual
Código:	fecha_actual
Etiqueta:	Fecha en la cual se solicito cambio de versión
Tipo:	DT
Longitud:	0 **Precisión:** 0

Descripción

Fecha en la cual se solicito cambio de versión.

Su valor inicial debe ser nulo.

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos fecha_beta

Nombre:	fecha_beta
Código:	fecha_beta
Etiqueta:	Fecha en la cual se solicitó cambio de versión beta
Tipo:	DT
Longitud:	0 **Precisión:** 0

Descripción

Fecha en la cual se solicitó cambio de versión beta.

Su valor inicial debe ser nulo.

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos fecha_suscripcion

Nombre:	fecha_suscripcion		
Código:	fecha_suscripcion		
Etiqueta:	Fecha a partir de la cual se registro la terminal como perteneciente al sistema		
Tipo:	DT		
Longitud:	0	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION_SISTEMA	INSTITUCION_SISTEMA
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos fecha_version

Nombre:	fecha_version		
Código:	fecha_version		
Etiqueta:	Fecha de referencia de la versión		
Tipo:	DT		
Longitud:	0	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos host_unix

Nombre:	host_unix		
Código:	host_unix		
Etiqueta:	Dirección IP del host de UNIX donde se encuentra la aplicación a transferir		
Tipo:	VA		
Longitud:	20	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	LOGIN_VERSION_SISTEMA	LOGIN_VERSION_SISTEMA

Atributos institucion

Nombre: institucion
Código: institucion
Etiqueta: Clave de la institución
Tipo: I
Longitud: 0 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos localidad_fiscal

Nombre: localidad_fiscal
Código: localidad_fiscal
Etiqueta: Localidad fiscal de la institución
Tipo: VA
Longitud: 15 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos login_unix

Nombre: login_unix
Código: login_unix
Etiqueta: Login de acceso al servidor UNIX
Domain:
Tipo: VA
Longitud: 20 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	LOGIN_VERSION	LOGIN_VERSION

Atributos mem_principal

Nombre: mem_principal
Código: mem_principal
Etiqueta: Tamaño mínimo necesario de memoria principal para el funcionamiento del sistema
Tipo: I
Longitud: 0 **Precisión:** 0

Descripción

Tamaño mínimo necesario de memoria principal para el buen funcionamiento de la aplicación (en KB)

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos mem_secundaria

Nombre: mem_secundaria
Código: mem_secundaria
Etiqueta: Tamaño mínimo necesario de memoria secundaria para el funcionamiento del sistema
Tipo: I
Longitud: 0 **Precisión:** 0

Descripción

Tamaño mínimo necesario de memoria secundaria para el buen funcionamiento de la aplicación (en KB)

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos nombre_corto

Nombre: nombre_corto
Código: nombre_corto
Etiqueta: Nombre corto de la institución
Tipo: A
Longitud: 15 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos nombre_institucion

Nombre: nombre_institucion
Código: nombre_institucion
Etiqueta: Nombre completo de la institución
Domain: descripcion
Tipo: A
Longitud: 70 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos nombre_sistema

Nombre:	nombre_sistema		
Código:	nombre_sistema		
Etiqueta:	Nombre completo del sistema registrado		
Domain:	descripcion		
Tipo:	A		
Longitud:	70	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA	SISTEMA

Atributos nombre_usuario

Nombre:	nombre_usuario		
Código:	nombre_usuario		
Etiqueta:	Nombre del usuario		
Tipo:	VA		
Longitud:	50	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	LOGIN_VERSION	LOGIN_VERSION

Atributos obtenible

Nombre:	obtenible		
Código:	obtenible		
Etiqueta:	S/N indica si la versión registrada puede ser distribuida		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA	SISTEMA
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos password_unix

Nombre:	password_unix		
Código:	password_unix		
Etiqueta:	Password de acceso al servidor UNIX		
Tipo:	VA		
Longitud:	30	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	LOGIN_VERSION	LOGIN_VERSION

Atributos permiso_beta

Nombre:	permiso_beta		
Código:	permiso_beta		
Etiqueta:	S/N Indica si puede obtener la versión beta de la aplicación		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos rfc

Nombre:	rfc		
Código:	rfc		
Etiqueta:	Registro Federal de Contribuyentes de la institución		
Domain:			
Tipo:	VA		
Longitud:	13	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	INSTITUCION	INSTITUCION

Atributos sistema

Nombre:	sistema		
Código:	sistema		
Etiqueta:	Sistema a distribuir		
Domain:			
Tipo:	A		
Longitud:	10	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA	SISTEMA

Atributos sistema_operativo

Nombre:	sistema_operativo		
Código:	sistema_operativo		
Etiqueta:	Versión mínima de sistema operativo		
Domain:			
Tipo:	SF		
Longitud:	0	**Precisión:**	0

Descripción

Versión mínima de sistema operativo en el cliente para que la aplicación funcione correctamente

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos tam_host

Nombre:	tam_host		
Código:	tam_host		
Etiqueta:	Tamaño del archivo compacto en bytes		
Tipo:	I		
Longitud:	0	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos terminal

Nombre:	terminal		
Código:	terminal		
Etiqueta:	Número de Terminal		
Tipo:	SI		
Longitud:	0	**Precisión:**	0

Descripción

Número único que identifica cada terminal que utiliza el sistema

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL	TERMINAL

Atributos tipo_version

Nombre:	tipo_version		
Código:	tipo_version		
Etiqueta:	Distingue una versión Beta (B) de una de Producción (P)		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos ubicacion_terminal

Nombre:	ubicacion_terminal		
Código:	ubicacion_terminal		
Etiqueta:	Ubicación de la terminal		
Domain:	descripcion		
Tipo:	A		
Longitud:	70	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL	TERMINAL

Atributos valida

Nombre:	valida		
Código:	valida		
Etiqueta:	S/N indica si la versión actual es válida. Si no es válida, no es obtenible.		
Tipo:	A		
Longitud:	1	**Precisión:**	0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos version

Nombre:	version
Código:	version
Etiqueta:	Nombre o nivel de la versión
Tipo:	VA
Longitud:	15 **Precisión:** 0

Lista de referencias

Referencia	Nombre	Código
Entidad	SISTEMA_VERSION	SISTEMA_VERSION

Atributos version_actual

Nombre:	version_actual
Código:	version_actual
Etiqueta:	Nombre de la versión que obtuvo en la última ejecución.
Tipo:	VA
Longitud:	15 **Precisión:** 0

Descripción

Nombre de la versión que obtuvo en la última ejecución.

Su valor inicial debe ser nulo (NULL)

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Atributos version_permitida

Nombre:	version_permitida
Código:	version_permitida
Etiqueta:	U/, (espacio en blanco), /A Determina que versión puede solicitar la terminal
Tipo:	A
Longitud:	1 **Precisión:** 0

Descripción

Si en este campo tiene el valor U, la terminal solamente podrá obtener la última versión según le corresponda en base al campo actualizar_versión; si tiene valor A, la terminal sólo podrá obtener la versión anterior; y si tiene valor " " (blanco) podrá obtener Ultima o Anterior

Lista de referencias

Referencia	Nombre	Código
Entidad	TERMINAL_SISTEMA	TERMINAL_SISTEMA

Entidades:

Lista de Entidades

Nombre	Código
CLAVE_ERROR	CLAVE_ERROR
INSTITUCION	INSTITUCION
INSTITUCION_SISTEMA	INSTITUCION_SISTEMA
LOGIN_VERSION	LOGIN_VERSION
LOGIN_VERSION_SISTEMA	LOGIN_VERSION_SISTEMA
SISTEMA	SISTEMA
SISTEMA_VERSION	SISTEMA_VERSION
TERMINAL	TERMINAL
TERMINAL_SISTEMA	TERMINAL_SISTEMA

Entidad CLAVE_ERROR

Nombre:	CLAVE_ERROR
Código:	CLAVE_ERROR
Etiqueta:	Tabla que almacena las claves de los errores que se pueden presentar
Número:	**Genera tabla:** Si

Lista de atributos

Nombre	Código	Tipo	I	M
error	error	I	Si	Si
descripcion	descripcion	A70	No	Si

Entidad INSTITUCION

Nombre:	INSTITUCION
Código:	INSTITUCION
Etiqueta:	Tabla que contiene la información de las instituciones que utilizan Versiones
Número:	**Genera tabla:** Si

Lista de atributos

Nombre	Código	Tipo	I	M
institucion	institucion	I	Si	Si
nombre_corto	nombre_corto	A15	No	Si
nombre_institucion	nombre_institucion	A70	No	Si
rfc	rfc	VA13	No	Si
direccion_fiscal	direccion_fiscal	VA80	No	Si
colonia_fiscal	colonia_fiscal	VA40	No	Si
cp_fiscal	cp_fiscal	A5	No	Si
localidad_fiscal	localidad_fiscal	VA15	No	No
ent_federativa_fiscal	ent_federativa_fiscal	VA25	No	No

Lista de referencias

Relación	Entidad	Tarjeta	R
InstitucionSistema(institucionSistema)	INSTITUCION_SISTEMA(INSTITUCION_SISTEMA)	0,n	Si
InstitucionTerminal(institucionTerminal)	TERMINAL(TERMINAL)	0,n	No

Entidad INSTITUCION_SISTEMA

Nombre:	INSTITUCION_SISTEMA
Código:	INSTITUCION_SISTEMA
Etiqueta:	Tabla que indica los sistemas que puede actualizar cada institución
Número:	**Genera tabla:** Si

Descripción

En esta tabla se insertan las instituciones que se desea que obtengan la versión de cierto sistema. No se podrán insertar instituciones que no estén en la tabla INSTITUCION.

Lista de atributos

Nombre	Código	Tipo	I	M
fecha_suscripcion	fecha_suscripcion	DT	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
InstitucionSistema(InstitucionSistema)	INSTITUCION(INSTITUCION)	1,1	Si
SistemaInstitucion(SistemaInstitucion)	SISTEMA(SISTEMA)	1,1	Si

Entidad LOGIN_VERSION

Nombre:	LOGIN_VERSION
Código:	LOGIN_VERSION
Etiqueta:	Tabla que contiene las claves de acceso al servidor UNIX
Número:	**Genera tabla:** Si

Lista de atributos

Nombre	Código	Tipo	I	M
login_unix	login_unix	VA20	Si	Si
password_unix	password_unix	VA30	No	Si
nombre_usuario	nombre_usuario	VA50	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
LoginVersionSistema(LOGINVERSIONSISTEMA)	LOGIN_VERSION_SISTEMA(LOGIN_VERSION_SISTEMA)	0,n	Si

Entidad LOGIN_VERSION_SISTEMA

Nombre:	LOGIN_VERSION_SISTEMA
Código:	LOGIN_VERSION_SISTEMA
Etiqueta:	Tabla que especifica los sistemas habilitados para cada login
Número:	**Genera tabla:** Si

Descripción

Tabla que especifica los sistemas habilitados para cada login. Además indica el host de UNIX donde se encuentra la aplicación a transferir.

Lista de atributos

Nombre	Código	Tipo	I	M
host_unix	host_unix	VA20	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
LoginVersionSistema(LOGINVERSIONSISTEMA)	LOGIN_VERSION(LOGIN_VERSION)	1,1	Si
SistemaLogin(SISTEMALOGIN)	SISTEMA(SISTEMA)	1,1	Si

Entidad SISTEMA

Nombre:	SISTEMA
Código:	SISTEMA
Etiqueta:	Tabla que almacena la información de los sistemas actualizados por Versiones
Número:	**Genera tabla:** Si

Lista de atributos

Nombre	Código	Tipo	I	M
sistema	sistema	A10	Si	Si
obtenible	obtenible	A1	No	Si
nombre_sistema	nombre_sistema	A70	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
SistemaLogin(SISTEMALOGIN)	LOGIN_VERSION_SISTEMA(LOGIN_VERSION_SISTEMA)	0,n	Si
SistemaInstitucion(SistemaInstitucion)	INSTITUCION_SISTEMA(INSTITUCION_SISTEMA)	0,n	Si
SistemaTerminal(SistemaTerminal)	TERMINAL_SISTEMA(TERMINAL_SISTEMA)	0,n	Si
SistemaVersion(SistemaVersion)	SISTEMA_VERSION(SISTEMA_VERSION)	0,n	Si

Entidad SISTEMA_VERSION

Nombre:	SISTEMA_VERSION
Código:	SISTEMA_VERSION
Etiqueta:	Tabla que contiene los datos de las versiones a distribuir de cada sistema
Número:	**Genera tabla:** Si

Lista de atributos

Nombre	Código	Tipo	I	M
fecha_version	fecha_version	DT	Si	Si
tipo_version	tipo_version	A1	Si	Si
version	version	VA15	No	Si
obtenible	obtenible	A1	No	Si
valida	valida	A1	No	Si
dir_host	dir_host	VA50	No	Si
dir_local	dir_local	VA50	No	Si
tam_host	tam_host	I	No	Si
mem_principal	mem_principal	I	No	Si
mem_secundaria	mem_secundaria	I	No	Si
espacio_disco	espacio_disco	I	No	Si
sistema_operativo	sistema_operativo	SF	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
SistemaVersion(SistemaVersion)	SISTEMA(SISTEMA)	1,1	Si

Entidad TERMINAL

Nombre:	TERMINAL		
Código:	TERMINAL		
Etiqueta:	Esta tabla almacena las características de cada terminal que accesa Versiones		
Número:		**Genera tabla:**	Si

Descripción

Esta tabla almacena las características de cada terminal que accesa al sistema de Versiones.

Lista de atributos

Nombre	Código	Tipo	I	M
terminal	terminal	Si	Si	Si
direccion_red	direccion_red	A18	No	Si
direccion_ip	direccion_ip	A18	No	Si
ubicacion_terminal	ubicacion_terminal	A70	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
InstitucionTerminal(InstitucionTerminal)	INSTITUCION(INSTITUCION)	1,1	No
TerminalSistema(TerminalSistema)	TERMINAL_SISTEMA(TERMINAL_SISTEMA)	0,n	Si

Entidad TERMINAL_SISTEMA

Nombre:	TERMINAL_SISTEMA		
Código:	TERMINAL_SISTEMA		
Etiqueta:	Tabla que indica que sistemas puede actualizar una terminal		
Número:		**Genera tabla:**	Si

Descripción

En esta tabla se lleva el control de la versión que tiene cada terminal. Obviamente no se permite insertar terminales que no existan en la tabla TERMINAL.

Lista de atributos

Nombre	Código	Tipo	I	M
actualizar_version	actualizar_version	A1	No	Si
version_permitida	version_permitida	A1	No	Si
version_actual	version_actual	VA15	No	Si
actualizar_beta	actualizar_beta	A1	No	Si
permiso_beta	permiso_beta	A1	No	Si
fecha_actual	fecha_actual	DT	No	No
fecha_beta	fecha_beta	DT	No	No
fecha_suscripcion	fecha_suscripcion	DT	No	Si

Lista de referencias

Relación	Entidad	Tarjeta	R
SistemaTerminal(SistemaTerminal)	SISTEMA(SISTEMA)	1,1	Si
TerminalSistema(TerminalSistema)	TERMINAL(TERMINAL)	1,1	Si

Relaciones:

Lista de relaciones

Nombre	Código
InstitucionSistema	InstitucionSistema
InstitucionTerminal	InstitucionTerminal
LoginVersionSistema	LOGINVERSIONSISTEMA
SistemaLogin	SISTEMALOGIN
SistemaInstitucion	SistemaInstitucion
SistemaTerminal	SistemaTerminal
SistemaVersion	SistemaVersion
TerminalSistema	TerminalSistema

Relación InstitucionSistema

Nombre: InstitucionSistema
Código: InstitucionSistema
Entidad 1: INSTITUCION
Entidad 2: INSTITUCION_SISTEMA
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 —> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 —> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre Institución y Sistema.

Relación InstitucionTerminal

Nombre: InstitucionTerminal
Código: InstitucionTerminal
Entidad 1: INSTITUCION
Entidad 2: TERMINAL
Entidad 2 dependiente de Entidad 1: No
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre Institución y terminales.

Relación LoginVersionSistema

Nombre: LoginVersionSistema
Código: LOGINVERSIONSISTEMA
Entidad 1: LOGIN_VERSION
Entidad 2: LOGIN_VERSION_SISTEMA
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre LOGIN_VERSION y LOGIN_VERSION_SISTEMA.

Relación SistemaLogin

Nombre: SistemaLogin
Código: SISTEMALOGIN
Entidad 1: SISTEMA
Entidad 2: LOGIN_VERSION_SISTEMA
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre LOGIN_VERSION_SISTEMA y SISTEMA.

Relación SistemaInstitucion

Nombre: SistemaInstitucion
Código: SistemaInstitucion
Entidad 1: SISTEMA
Entidad 2: INSTITUCION_SISTEMA
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre Sistema e Institución.

Relación SistemaTerminal

Nombre: SistemaTerminal
Código: SistemaTerminal
Entidad 1: SISTEMA
Entidad 2: TERMINAL_SISTEMA
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre Sistema y Terminal.

Relación SistemaVersion

Nombre: SistemaVersion
Código: SistemaVersion
Entidad 1: SISTEMA
Entidad 2: SISTEMA_VERSION
Entidad 2 dependiente de Entidad 1: Si
Entidad 1 --> Entidad 2:
Etiqueta:
Ocurrencia: Muchos - Opcional
Entidad 2 --> Entidad 1:
Etiqueta:
Ocurrencia: Uno - Obligatorio

Descripción

Relación entre Sistema y Versión.

Relación TerminalSistema

Nombre:	TerminalSistema
Código:	TerminalSistema
Entidad 1:	TERMINAL
Entidad 2:	TERMINAL_SISTEMA
Entidad 2 dependiente de Entidad 1:	Si
Entidad 1 --> Entidad 2:	
Etiqueta:	
Ocurrencia:	Muchos - Opcional
Entidad 2 --> Entidad 1:	
Etiqueta:	
Ocurrencia:	Uno - Obligatorio

Descripción

Relación entre Terminal y Sistema.

3.6 Portafolio de Programas

El sistema está compuesto por un conjunto de programas que permiten realizar los procesos antes mencionados.

A continuación se muestra la lista de programas que intervienen en la aplicación para posteriormente detallar cada uno de ellos.

Nombre de programa	Objetivo
Programa de identificación de terminal	Identificar la terminal remota por medio de su dirección IP
Programa de verificación de acceso de transferencia	Verificar permiso de transferencia
Programa de verificación de requerimientos del sistema	Verificar requerimientos del sistema
Programa de transferencia	Transferencia del sistema
Programa de verificación de existencia del compacto	Verificación del compacto
Programa de verificación de permisos para instalación	Verificación de permisos
Programa de Instalación	Instalación del sistema

Nombre	Programa de identificación de terminal
Objetivo	Identificar la terminal remota por medio de su dirección IP
Datos de entrada	
Datos de salida	Dirección IP
Descripción	Este es un programa que detecta la dirección IP de la terminal remota por medio de sockets

Nombre	Programa de verificación de acceso de transferencia
Objetivo	Verificar permiso de transferencia
Datos de entrada	Sistema, Versión, Dirección IP
Datos de salida	Requisitos del sistema
Descripción	Este programa accesa a la base de datos y verifica si la terminal tiene permiso para transferir el sistema solicitado y en la versión especificada. Si el acceso de transferencia es válido, el programa regresa información a la terminal externa que servirá para verificar los requerimientos del sistema o aplicación solicitada, y también información para la conexión al servidor de aplicaciones. En caso de que el acceso no sea válido retorna error.

Nombre	Programa de verificación de requerimientos del sistema
Objetivo	Verificar requerimientos del sistema
Datos de entrada	Espacio en disco, mem. Principal, mem. Secundaria, Sistema Operativo, Tamaño del archivo, Directorio local
Datos de salida	Mensaje
Descripción	Este programa verifica que la terminal cumpla con los requisitos para la transferencia e instalación del sistema, tales como, espacio en disco, versión de sistema operativo, memoria principal y secundaria, etc.

Nombre	Programa de transferencia
Objetivo	Transferencia del sistema
Datos de entrada	Dir. Local, Dir. Host, Dirección IP, Sistema, Versión, Login, Password, Tamaño del Archivo
Datos de salida	Mensaje
Descripción	Este programa ejecuta la transferencia de la aplicación desde el servidor UNIX hasta la terminal remota a través de FTP

Nombre	Programa de verificación de existencia del compacto
Objetivo	Verificación del compacto
Datos de entrada	Tamaño del Archivo, Dir. Local
Datos de salida	Mensaje
Descripción	Este programa verifica que el sistema solicitado se haya transferido exitosamente y se encuentre en el subdirectorio de trabajo

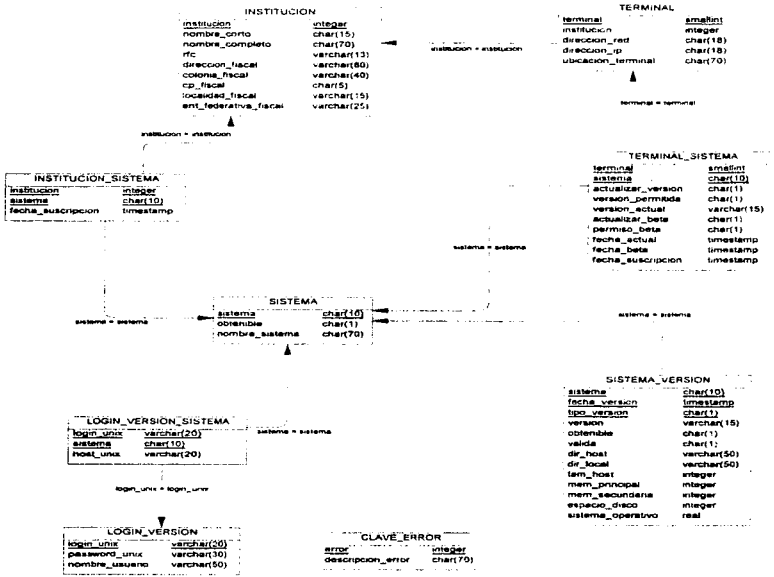
Nombre	Programa de verificación de permisos para instalación
Objetivo	Verificación de permisos
Datos de entrada	Sistema, Versión, Dirección IP
Datos de salida	Mensaje
Descripción	Este programa consulta en la base de datos si el sistema especificado puede ser instalado por la terminal. Si la terminal no tiene permisos de instalación retorna error

Nombre	Programa de Instalación
Objetivo	Instalación del sistema
Datos de entrada	Sistema, Versión
Datos de salida	Mensaje
Descripción	Este programa descompacta la aplicación transferida en el subdirectorio de trabajo para posteriormente realizar su instalación y configurar el entorno en que se ejecutará

3.7 Diseño físico de la base de datos

Del modelo lógico de la base de datos obtuvimos el modelo físico siguiente, el cuál se implementará posteriormente.

Gráficas del modelo físico



Modelo de Información

Nombre del proyecto:	Sistema que actualiza y distribuye versiones
Nombre:	Versiones
Código:	Versiones

Dominios:

Lista de dominios

Nombre	Código	Tipo
descripción	descripción	char(70)

Descripción de dominios

Nombre:	descripción		
Código:	descripción		
Obligatorio:	No		
Tipo:	char(70)		
Longitud:	70	**Precisión:**	0

Lista de Referencia

Tabla Código	Column Nombre	Código de la columna	Etiqueta
CLAVE_ERROR	descripcion_error	descripcion_error	Descripción del error
INSTITUCION	nombre_completo	nombre_completo	Nombre completo de la institución
SISTEMA	nombre_sistema	nombre_sistema	Nombre completo del sistema registrado
TERMINAL	ubicacion_terminal	ubicacion_terminal	Ubicación de la terminal

Tablas:

Lista de Tablas

Nombre	Código
CLAVE_ERROR	CLAVE_ERROR
INSTITUCION	INSTITUCION
INSTITUCION_SISTEMA	INSTITUCION_SISTEMA
LOGIN_VERSION	LOGIN_VERSION
LOGIN_VERSION_SISTEMA	LOGIN_VERSION_SISTEMA
SISTEMA	SISTEMA
SISTEMA_VERSION	SISTEMA_VERSION
TERMINAL	TERMINAL
TERMINAL_SISTEMA	TERMINAL_SISTEMA

Tabla CLAVE_ERROR

Nombre:	CLAVE_ERROR
Código:	CLAVE_ERROR
Etiqueta:	Tabla que almacena las claves de los errores que se pueden presentar
Fuente:	Entidad CLAVE_ERROR

Lista de columnas

Nombre	Código	Tipo	P	M
Descripcion_error	descripcion_error	char(70)	No	Si
Error	error	integer	Si	Si

Lista de Indices

Código del índice	P	F	U	C	Código de la columna	Orden
CLAVE_ERROR_PK	Si	No	Si	No	error	ASC

Tabla INSTITUCION

Nombre:	INSTITUCION
Código:	INSTITUCION
Etiqueta:	Tabla que contiene la información de las instituciones que utilizan Versiones
Fuente:	Entidad INSTITUCION

Lista de columnas

Nombre	Código	Tipo	P	M
colonia_fiscal	colonia_fiscal	varchar(40)	No	Si
cp_fiscal	cp_fiscal	char(5)	No	Si
direccion_fiscal	direccion_fiscal	varchar(80)	No	Si
ent_federativa_fiscal	ent_federativa_fiscal	varchar(25)	No	No
Institucion	institucion	integer	Si	Si
localidad_fiscal	localidad_fiscal	varchar(15)	No	No
nombre_completo	nombre_completo	char(70)	No	Si
nombre_corto	nombre_corto	char(15)	No	Si
Rfc	rfc	varchar(13)	No	Si

Lista de Indices

Código del índice	P	F	U	C	Código de la columna	Orden
INSTITUCION_PK	Si	No	Si	No	institucion	ASC

Lista de Referencia desde

Referenciado por	Llave primaria	Llave foránea
INSTITUCION_SISTEMA	institucion	institucion
TERMINAL	institucion	institucion

Tabla INSTITUCION_SISTEMA

Nombre:	INSTITUCION_SISTEMA
Código:	INSTITUCION_SISTEMA
Etiqueta:	Tabla que indica los sistemas que puede actualizar cada institución
Número:	
PK constraint:	
Fuente:	Entidad INSTITUCION_SISTEMA

Descripción

En esta tabla se insertan las instituciones que se desea que obtengan la versión de cierto sistema. No se podrán insertar instituciones que no estén en la tabla INSTITUCION.

Lista de columnas

Nombre	Código	Tipo	P	M
fecha_suscripcion	fecha_suscripcion	timestamp	No	Si
institucion	institucion	integer	Si	Si
sistema	sistema	char(10)	Si	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
INSTITUCION_SISTEMA_PK	Si	Si	Si	No	institucion	ASC
					sistema	ASC

Lista de referencia hacia

Referencia hacia	Llave primaria	Llave foránea
INSTITUCION	institucion	institucion
SISTEMA	sistema	sistema

Tabla LOGIN_VERSION

Nombre:	LOGIN_VERSION
Código:	LOGIN_VERSION
Etiqueta:	Tabla que contiene las claves de acceso al servidor UNIX
Fuente:	Entidad LOGIN_VERSION

Lista de columnas

Nombre	Código	Tipo	P	M
login_unix	login_unix	varchar(20)	Si	Si
nombre_usuario	nombre_usuario	varchar(50)	No	Si
password_unix	password_unix	varchar(30)	No	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
LOGIN_VERSION_PK	Si	No	Si	No	login_unix	ASC

Lista de Referencia desde

Referenciado por	Llave primaria	Llave foránea
LOGIN_VERSION_SISTEMA	login_unix	login_unix

Tabla LOGIN_VERSION_SISTEMA

Nombre:	LOGIN_VERSION_SISTEMA
Código:	LOGIN_VERSION_SISTEMA
Etiqueta:	Tabla que especifica los sistemas habilitados para cada login
Fuente:	Entidad LOGIN_VERSION_SISTEMA

Opciones

Descripción

Tabla que especifica los sistemas habilitados para cada login.

Además indica el host de UNIX donde se encuentra la aplicación a transferir.

Lista de columnas

Nombre	Código	Tipo	P	M
host_unix	host_unix	varchar(20)	No	Si
login_unix	login_unix	varchar(20)	Si	Si
sistema	sistema	char(10)	Si	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
LOGIN_VERSION_SISTEMA_PK	Si	Si	Si	No	login_unix	ASC
					sistema	ASC

Lista de referencia hacia

Referencia hacia	Llave primaria	Llave foránea
LOGIN_VERSION	login_unix	login_unix
SISTEMA	sistema	sistema

Tabla SISTEMA

Nombre:	SISTEMA
Código:	SISTEMA
Etiqueta:	Tabla que almacena la información de los sistemas actualizados por Versiones
Fuente:	Entidad SISTEMA

Lista de columnas

Nombre	Código	Tipo	P	M
nombre_sistema	nombre_sistema	char(70)	No	Si
obtenible	obtenible	char(1)	No	Si
sistema	sistema	char(10)	Si	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
SISTEMA_PK	Si	No	Si	No	sistema	ASC

Lista de Referencia desde

Referenciado por	Llave primaria	Llave foránea
INSTITUCION_SISTEMA	sistema	sistema
LOGIN_VERSION_SISTEMA	sistema	sistema
TERMINAL_SISTEMA	sistema	sistema
SISTEMA_VERSION	sistema	sistema

Tabla SISTEMA_VERSION

Nombre:	SISTEMA_VERSION
Código:	SISTEMA_VERSION
Etiqueta:	Tabla que contiene los datos de las versiones a distribuir de cada sistema
Fuente:	Entidad SISTEMA_VERSION

Lista de columnas

Nombre	Código	Tipo	P	M
dir_host	dir_host	varchar(50)	No	Si
dir_local	dir_local	varchar(50)	No	Si
espacio_disco	espacio_disco	integer	No	Si
fecha_version	fecha_version	timestamp	Si	Si
mem_principal	mem_principal	integer	No	Si
mem_secundaria	mem_secundaria	integer	No	Si
obtenible	obtenible	char(1)	No	Si
sistema	sistema	char(10)	Si	Si
sistema_operativo	sistema_operativo	real	No	Si
tam_host	tam_host	integer	No	Si
tipo_version	tipo_version	char(1)	Si	Si
valida	valida	char(1)	No	Si
version	version	varchar(15)	No	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
SISTEMA_VERSION_PK	Si	No	Si	No	sistema	ASC
					fecha_version	ASC
					tipo_version	ASC

Lista de referencia hacia

Referencia hacia	Llave primaria	Llave foránea
SISTEMA	sistema	sistema

Tabla TERMINAL

Nombre:	TERMINAL
Código:	TERMINAL
Etiqueta:	Esta tabla almacena las características de cada terminal que accesa
Versiones	
Fuente:	Entidad TERMINAL

Lista de columnas

Nombre	Código	Tipo	P	M
direccion_ip	direccion_ip	char(18)	No	Si
direccion_red	direccion_red	char(18)	No	Si
institucion	institucion	integer	No	Si
terminal	terminal	smallint	Si	Si
ubicacion_terminal	ubicacion_terminal	char(70)	No	Si

Lista de Indices

Código del índice	P	F	U	C	Código de la columna	Orden
InstitucionTerminal_FK	No	Si	No	No	institucion	ASC
TERMINAL_PK	Si	No	Si	No	terminal	ASC

Lista de referencia hacia

Referencia hacia	Llave primaria	Llave foránea
INSTITUCION	institucion	institucion

Lista de Referencia desde

Referenciado por	Llave primaria	Llave foránea
TERMINAL_SISTEMA	terminal	terminal

Tabla TERMINAL_SISTEMA

Nombre:	TERMINAL_SISTEMA
Código:	TERMINAL_SISTEMA
Etiqueta:	Tabla que indica que sistemas puede actualizar una terminal
Fuente:	Entidad TERMINAL_SISTEMA

Descripción

En esta tabla se lleva el control de la versión que tiene cada terminal. Obviamente no se permite insertar terminales que no existan en la tabla TERMINAL.

Lista de columnas

Nombre	Código	Tipo	P	M
actualizar_beta	actualizar_beta	char(1)	No	Si
actualizar_version	actualizar_version	char(1)	No	Si
fecha_actual	fecha_actual	timestamp	No	No
fecha_beta	fecha_beta	timestamp	No	No
fecha_suscripcion	fecha_suscripcion	timestamp	No	Si
permiso_beta	permiso_beta	char(1)	No	Si
sistema	sistema	char(10)	Si	Si
terminal	terminal	smallint	Si	Si
version_actual	version_actual	varchar(15)	No	Si
version_permitida	version_permitida	char(1)	No	Si

Lista de índices

Código del índice	P	F	U	C	Código de la columna	Orden
TERMINAL_SISTEMA_PK	Si	Si	Si	No	terminal	ASC
					sistema	ASC

Lista de referencia hacia

Referencia hacia	Llave primaria	Llave foránea
SISTEMA	sistema	Sistema
TERMINAL	terminal	Terminal

Una vez realizado el análisis y diseño del sistema procederemos a revisar las herramientas de hardware y software que nos permitirán desarrollar el sistema.

Capítulo
CUATRO

DESARROLLO E IMPLEMENTACIÓN

En este capítulo comenzamos por mostrar de forma específica como será resuelto el problema y se definirán cuales serán los recursos empleados en la solución; lo anterior se hará mediante una evaluación del software y hardware, y se mostrará el costo aproximado de estos. Además se mostrarán algunos de los aspectos de la implementación del sistema, tales como, la especificación de los programas y distribución de los archivos en el servidor. Además se diseñarán las pantallas de la parte del cliente de la aplicación. Finalmente se concluirá con un calendario de implementación del sistema.

4.1 Análisis y selección del software

En este apartado se analizarán algunas de las alternativas de software que se encuentran disponibles en el mercado, específicamente se abordarán los siguientes casos: Manejadores de Bases de Datos Relacionales, Herramientas de Desarrollo de Aplicaciones y Herramientas de procesamiento de scripts. No se han incluido los sistemas operativos en las evaluaciones porque estos pueden verse determinados por las plataformas en las que se ejecuten tanto el Manejador de Bases de Datos como la Herramienta de Desarrollo de Aplicaciones.

4.1.1 Manejadores de Bases de Datos Relacionales

En este apartado realizaremos una evaluación y comparación de las características de los principales Manejadores de Bases de Datos Relacionales (RDBMS) que se encuentran disponibles en el mercado: ORACLE7 versión 7.3, SYBASE SQL SERVER 11, INFORMIX-ONLINE 7.2, MICROSOFT SQL SERVER 6.5, IBM DB2 2.11, CA-OPENINGRES 1.2. Todos ellos se caracterizan por contar con optimizadores de consultas, manejo de *triggers*, vistas, integridad declarativa y soporte para bases de datos distribuidas. También pueden mencionarse aspectos que cubren las nuevas tendencias

en informática como el soporte para Internet, llamadas a procedimientos remotos (*rpc's*) y completo aprovechamiento de las plataformas con múltiples procesadores.

Si sólo se observan algunas características superficiales de los RDBMS, parece que puede tomarse cualquiera y ajustarse de tal forma que se adapte al problema que se quiere resolver, sin embargo, debemos efectuar una evaluación más cuidadosa para verificar la extensión y la calidad de la implementación de cada producto.

Modelo de Datos Relacional

Aunque todos los Manejadores de Bases de Datos Relacionales analizados se denominan relacionales, su soporte del modelo de datos relacional debe ser revisado cuidadosamente. Obviamente todos ellos soportan los conceptos básicos relacionales, tales como datos almacenados en tablas y accesos a estos datos mediante operaciones de conjunto de alto nivel (Algebra Relacional), la mayoría de ellos empleando SQL[1]. Sin embargo ninguno de ellos soporta dominios, en un futuro será crucial que estos productos atiendan este concepto fundamental del modelo de datos relacional. Deben permitir de esta forma definir los dominios y entonces especificar las columnas de las tablas, y preferiblemente también los parámetros y variables de los procedimientos almacenados en términos de dominios. Este proceso es necesario para asegurar estricta verificación de tipos, tal como puede hacerse en algunos lenguajes de programación.

Aunque todos ellos aseguran soportar *constraints* para integridad declarativa, sólo Informix y Oracle soportan el borrado en cascada como una opción cuando se viola un *constraint* de integridad referencial, y sólo DB2 cumple completamente con el conjunto vacío, borrado en cascada, y ninguna acción tal como lo prescribe el estándar ANSI SQL-92. Deben verificarse también como son implementados los *constraints*, la mayoría de los productos emplean un mecanismo bastante rudimentario. Por ejemplo, casi todos crean índices ocultos para implementar los *constraints* de llave primaria. Además DB2, Informix y Oracle restringen la creación de un índice único adicional sobre una columna que haya sido previamente indexado de forma oculta al haberse definido previamente un *constraint único*[2].

Objetos de Bases de Datos

Todos los productos revisados soportan BLOB's (*binary large objects*), los cuales pueden ser empleados para almacenar imágenes texto, documentos, voz, grabaciones de audio y cualquier otro dato sin estructura. Sin embargo, debe considerarse cuidadosamente como los BLOB's son procesados por las herramientas y lenguajes de desarrollo de aplicaciones (*front-end*). No todas las herramientas pueden manejar fácilmente este tipo de datos. Con algunos lenguajes, tales como C y C++, usando un preprocesador SQL, puede procesarse un texto BLOB como una serie de segmentos de texto. Algunos ambientes de desarrollo de cuarta generación pueden no ser capaces de procesar los BLOB's proporcionados por el servidor.

[1] Lenguaje de Consultas Estructurado (Structured Query Language).
[2] Se llama *constraint único* aquel que determina que las columnas sobre las cuales ha sido definido el constraint, deben formar una combinación única dentro de toda la tabla.

Deben observarse como son implementados los *triggers*, lo cual es un factor importante. Algunos productos emplean sus mecanismos de *triggers* de forma oculta para implantar sus *constraints* de integridad declarativa. Esto es aún más importante si se desean implementar las reglas del negocio usando *triggers*. Por ejemplo, CA-OpenIngres sólo tiene *triggers* basados en registro (llamados reglas) que 'disparan' todas las operaciones del *trigger*, y pueden tenerse cualquier número de *triggers* por tabla, cada uno con su propio nombre, y de esta forma pueden implantarse las reglas del negocio de forma modular. Por otro lado, Informix, DB2 y Oracle tienen *triggers* basados en registro y en conjuntos de registros, los cuales pueden ejecutarse antes y después de la operación que 'disparó' el *trigger*, sin embargo, sólo puede tenerse un trigger por cada condición de 'disparo', lo cual significa que algunas veces deben integrarse reglas del negocio no relacionadas en un mismo *trigger*. Sin embargo, puede mejorarse esta situación llamando *stored procedures* diferentes para cada función de procesamiento de las reglas del negocio.

Consultas

El nivel de bloqueo proporcionado por los diversos DBMS ha sido un tema bastante explotado en el material de mercadotecnia de los productos para opacar a sus oponentes. No hay una guía clara para poder decidir cuando un bloqueo a nivel de registro es mejor que un bloqueo a nivel de página; el tipo de bloqueo más adecuado dependerá siempre de la aplicación, y de la carga de trabajo y aislamiento que se requiera. Es, sin embargo, extremadamente importante determinar los requerimientos de control de la concurrencia de las aplicaciones y verificar cuando el producto puede satisfacer las necesidades.

Todos los RDBMS's evaluados en este apartado, aseguran soportar el estándar ANSI SQL-92 en el nivel básico '*Entry*'. Es importante recordar que este nivel realmente cubre el nivel más bajo de las operaciones de definición de las bases de datos y de manipulación de éstas. Los aspectos más importantes son cubiertos en los niveles superiores '*Intermediate*' y '*Full*' de este estándar. Este estándar fue publicado en 1992, y bastante de su contenido era conocido desde antes. Es hora de que los consumidores hagan presión sobre los proveedores de los RDBMS's para que implementen los niveles superiores del estándar.

Uno de los aspectos que también deben revisarse es el caso de la operación *outer join*. Esta operación es usada más de lo que generalmente se piensa, por tanto es extremadamente importante verificar que el RDBMS soporte los tipos necesarios de *outer joins* en los cuales pudiera estarse interesado, tales como el *outer join* completo, por la izquierda y por la derecha. Debe revisarse la sintaxis usada, ya que aunque todos los productos manejan alguna variación de la operación *outer join*, sólo CA-OpenIngres y Microsoft SQL Server soportan la sintaxis especificada por el estándar ANSI SQL-92, la cual indica que el *outer join* debe ser especificado explícitamente en la cláusula FROM de la instrucción SELECT. Los demás productos emplean alguna variación de la sintaxis de Oracle, en la cual el *outer join* es especificado usando algunos símbolos adicionales tales como un asterisco o un signo '+' en las condiciones del *join* de la cláusula WHERE.

Conectividad y distribución

El soporte para bases de datos distribuidas ofrecidos por los proveedores de RDBMS's varía considerablemente. Algunos productos tales como CA-OpenIngres, DB2 y Sybase ofrecen un *shell* de múltiples bases de datos que permiten accesar tablas remotas. Cuando se accesan datos almacenados en múltiples bases de datos en la misma transacción, estos tres productos pueden aplicar el método de *two-phase commit* (2PC). Otros productos dejan accesar remotamente datos en una transacción y dan las facilidades para implementar el método 2PC de forma programada. Sin embargo, debe programarse el método dentro de la aplicación.

Algunos productos implementan también el uso de llamadas a procedimientos remotos (RPC's), los cuales permiten accesar datos en bases de datos remotas como si estos datos estuvieran almacenados localmente. Por ejemplo, un stored procedure llamado por una aplicación puede realizar verificación de la integridad en una base de datos remota, sin que la aplicación deba preocuparse de realizarlo. Sybase en particular, tiene una poderosa implementación de RPC's.

Replicación

La replicación es un tema tan amplio que exige en sí misma, una investigación propia. Hay muchos aspectos que considerar cuando evalúan productos de replicación, tales como la topología soportada, la arquitectura de estos productos, funcionalidad, administración y herramientas de monitoreo, y aún más importante, es la carga de trabajo soportada. Si el replicador no puede proporcionar la capacidad de manejo de transacciones para una topología determinada, simplemente no trabajará de forma adecuada. Es importante verificar si puede replicar transacciones de forma bidireccional en una configuración punto a punto. De los seis manejadores analizados, sólo DB2, CA-OpenIngres, Oracle y Sybase pueden ser configurados para replicar transacciones en una configuración punto a punto. Otro aspecto consiste en si los RDBMS's pueden replicar hacia otros RDBMS's en una configuración heterogénea. La mayoría de los productos pueden replicar hacia diferentes bases de datos a través de sus productos *gateways*; pero sólo unos cuantos, como el Replication Server de Sybase, pueden replicar hacia y desde diversas bases de datos.

Internet

Aunque los seis productos revisados soportan de una u otra forma el *Internet* y el *World Wide Web*, ellos pueden ser particionados en dos tipos diferentes: los basados en *scripts* y los basados en servidor. Productos tales como Informix, CA-OpenIngres y Sybase caen en la clase de los basados en scripts, en los cuales se pueden incluir operaciones SQL o scripts de Perl en las páginas HTML o utilizar scripts de CGI para poder accesar una base de datos especificada cuando la página Web es activada. Estos scripts deben asegurarse que los datos recuperados están en un formato legible para HTML. Productos tales como DB2, Oracle y Microsoft SQL Server caen dentro del tipo de los basados en servidor, en los cuales un proceso servidor dedicado actúa como un *gateway* hacia la base de datos designada para recuperar los datos y enviarlos hacia la página Web en una forma HTML. Además, estos productos incluyen herramientas para

ayudar a desarrollar aplicaciones Web, tales como el SQL Server Web Assistant que viene incluido con Microsoft SQL Server.

Evaluación

Cada uno de los productos revisado en esta sección tienen excelentes características en algunas áreas mientras su calidad es dudable en otras, es por esto que no hay un claro ganador o perdedor. Lo importante de evaluar un RDBMS consiste en determinar que conjunto de características es importante para las aplicaciones que se van a desarrollar, ponderar estas características de acuerdo a su importancia y finalmente comparar los productos de acuerdo a los requerimientos. Deben leerse reportes de los analistas y suplementos de comparación que aparecen en revistas especializadas y así obtener buenos puntos de referencia acerca de las características y debilidades de cada uno de los productos de interés. Sin embargo, sólo un análisis exhaustivo de las aplicaciones podrá establecer los requerimientos que debe satisfacer el RDBMS.

Deben agregarse muchos factores adicionales en la evaluación, tales como costo, manejo de las licencias, costos de soporte, disponibilidad del soporte, herramientas de terceros, soluciones y grupos de usuarios con los cuales se puedan intercambiar ideas. Es también buena idea discutir los productos con algunos de los usuarios y desarrolladores.

A continuación se encuentra la tabla 4.1 que resume las principales características de los productos revisados.

	Oracle 7 Ver. 7.3	Sybase SQL Server 11	Informix Online 7.2	Microsoft SQL Server 6.5	IBM DB2 2.1.1	CA-Openingres 1.2
Modelo de datos Relacional						
Dominios	No	No	No	No	No	No
Opciones de violación de integridad referencial	Restrictivas, excepto el borrado en cascada	Sólo restrictivas	Restrictivas, excepto el borrado en cascada	Sólo restrictivas	Restrictivas y en cascada	Sólo restrictivas
Personalizar mensajes referenciales	No	No	No	No	No	No
Cláusulas WHERE referenciales	No	No	No	No	No	No
Vistas actualizables (con opción de verificación)	Si	Si	Si	Si	Si, incluyendo vistas con unión	Si

Tabla 4.1 Cuadro comparativo de los principales Manejadores de Bases de Datos Relacionales (continúa).

	Oracle 7 Ver. 7.3	Sybase SQL Server 11	Informix Online 7.2	Microsoft SQL Server 6.5	IBM DB2 2.1.1	CA-Openingres 1.2
Objetos de Bases de Datos						
Tipos de datos definidos por el usuario	Si	Si	No	Si	Si	Si
BLOB's	Si	Si	Si	Si	Si	Si
Tipos de datos soportados	*image, video, text, messaging, spatial data types*	*binary, image, money, bit, text, varbinary*	*byte, text hasta 2Gb*		*Large object*	*byte, byte varying, long byte, long varchar, money, spatial data types*
Estructuras de tablas	*Heap* y *clustered*	*Heap* y *clustered*	Sin elección	Sin elección	Sin elección	*B-tree, hash, heap e ISAM*
Estructuras de Indices	*B-tree, bitmap* y *hash*	*B-tree*	*B+ tree* y *clustered*	Clustered	Clustered	*B-tree, hash e ISAM*
Facilidades de optimización 'Tunning'	Distribución de Indices y tablas	*Index pre-fetch, I/O buffer cache, block size, table partitioning*	*Extents*, fragmentación de tablas o *round robin*	*Fill factors*	Distribución de indices y tablas, *cluster ratio* y *cluster factor*	*Fill factors*, distribución de indices y tablas
Nivel de *triggers*	Basado en registro y basado en conjuntos de registros	Basado en conjuntos de registros	Basado en registro y basado en conjuntos de registros	Basado en conjuntos de registros	Basado en registro y basado en conjuntos de registros	Basado en registro
Ejecución del *trigger*	Antes y después de la operación que 'disparó' el trigger	Después de la operación que 'disparó' el trigger	Antes, para cada y después de la operación que 'disparó' el trigger	Después de la operación que 'disparó' el trigger	Antes y después de la operación que 'disparó' el tngger	Después de la operación que 'disparó' el tngger
Anidamiento de *triggers*	Si	Si	Si	Si	Si	Si
Lenguaje empleado en *stored procedures*	PL/SQL	Transact-SQL	SPL	Transact-SQL	SQL & 3GL	SQL
Anidamiento de *stored procedures*	Si	Si	Si	Si	Si	Si
Cursores en *stored procedures*	Si	Si	Si	Si	Si	No
Llamadas externas de *stored procedures*	RPC	RPC	Llamadas al sistema	Llamadas al sistema	Si	No
Eventos en *stored procedures*	Si	Controlados por tiempo	No	No	Mediante funciones definidas por el usuario	Alertas de eventos en la Base de datos

Tabla 4.1 Cuadro comparativo de los principales Manejadores de Bases de Datos Relacionales (continúa).

	Oracle 7 Ver. 7.3	**Sybase SQL Server 11**	**Informix Online 7.2**	**Microsoft SQL Server 6.5**	**IBM DB2 2.1.1**	**CA-Openingres 1.2**
Consultas						
Nivel de bloqueo	Tabla, registro	Tabla, página	Base de datos, tabla, página, registro	Base de datos, tabla, página, registro	Base de datos, tabla, página, registro	Base de datos, tabla, página, registro
Cumplimiento del estándar ANSI SQL	*Entry Level* de SQL-92	*Entry Level* de SQL-92	*Entry Level* de SQL-92	*Entry Level* de SQL-92	*Entry Level* de SQL-92	*Entry Level* de SQL-92
Cursores	Hacia adelante	Hacia adelante	Hacia adelante y hacia atras	Hacia adelante, hacia atras relativo y absoluto	Hacia adelante	Hacia adelante
Outer joins	Si	Si	Si	Si	Si	Si
Sintaxis ANSI de *outer joins*	No	No	No	Si	No	Si
API's	ODBC	DBLIB, CTLIB, ODBC	ESQL, TP/XA, CLI, ODBC	ESQL, DBLIB, Distributed Management Objects, ODBC	ESQL, ODBC	ESQL, TP/XA, ODBC
Administración de Bases de datos						
Herramientas	Oracle Enterprise Manager, Performance Pack	Sybase SQL Manager, SQL Monitor	SMI, DB/Cockpit, OnPerf	Enterprise Manager, Performance Monitor	Database Director, Visual Explain, Performance Monitor	IPM, VisualDBA, IMA
Soporte a SNMP	Si	Si	No	Si	Si	Si
Seguridad	C2	C2	C2, B1	Heredada de NT	Tres niveles	C2
Respaldos parciales y recuperación	Configurable	Configurable	No	Por tabla	Si	Por tabla
Internet						
Soporte a Internet	Oracle WebServer	web sql	ESQL ó 4GL CGI Interface Kit	Internet Information Server en Windows NT	DB2 WWW Connection	CA-Openingres /ICE
Conectividad, Distribución y *Middleware*						
Gateways a otros RDBMS's	Cualquier fuente de datos MVVS a través de EDA/SQL (Adabas, IDMS, IMS, SQL/DS, VSAM), cualquier fuente de datos APPC, AS/400, DRDA, DB2, Turbolmage, Sybase, Rdb, RMS, Informix, CA-Ingres, SQL Server, Teradata	Adabas, AS/400, DB2, IDMS, IMS, Informix, Ingres, ISAM, Microsoft SQL Server, Oracle, Rdb, RMS, archivos secuenciales, SQL/DS, Teradata, VSAM	Oracle, Sybase, IMS, DB2	No	Oracle, Sybase, Informix, Microsoft SQL Server	DB2, Datacom, IMS, IDMS, VSAM, Oracle, Rdb, Allbase, Informix, Oracle, Sybase

Tabla 4.1 Cuadro comparativo de los principales Manejadores de Bases de Datos Relacionales (continúa).

	Oracle 7 Ver. 7.3	Sybase SQL Server 11	Informix Online 7.2	Microsoft SQL Server 6.5	IBM DB2 2.1.1	CA-Openingres 1.2
Conectividad, Distribución y *Middleware*						
Bases de datos distribuidas	Incluido como parte del producto	OmniConnect	Online Server	No	DataJoiner	CA-Openingres /Star
Protocolo 2PC	Si	Si	Si	N/A	Si	Si, automático
Esquema heterogéneo de Bases de datos	A través de gateways	A través de DirectConnect	No	No	A través de DataJoiner	A través de gateways
Optimización	Si	Si	Si	No	Si	Si
RPC	Si	Si	No	Si	No	No
Replicación						
Registro	Replicación de *log* y de *triggers*	Empleando el *log*	Empleando el *log*	Empleando el *log*	Empleando el *log*	Replicación de reglas y *triggers*
Hot standby	Si	Si	Si	Si	Si	Si
Punto a punto	Si	Si	No	No	Si	Si
Replicación hacia otros RDBMS's	A través de gateways	A través de DirectConnect	No	A través de ODBC	A través de DataJoiner	A través de gateways
Replicación en cascada	Si	Si	No	No	No	Si
Otras características adicionales						
Longitud de nombres de objetos	30	30	18	30	18	32
Número máximo de columnas	254	250	2767	250	255	300
Tamaño de las columnas	2 Gb	1962	32,767	255	4005	2008
Numero de tablas	N/A	2 billones	477 millones	2 billones	Depende del espacio en el disco	N/A
Tamaño de las tablas	N/A	Depende del espacio en el disco	64 terabytes	2 terabytes	64 Gb	N/A
Plataformas (sistemas operativos)	Mayoría de los sistemas UNIX, Windows NT, VAX VMS, Windows 95, OS/2, Macintosh	Mayoría de los sistemas UNIX, Windows NT, VAX VMS, Windows 95, OS/2, Macintosh	Mayoría de los sistemas UNIX, Windows NT, Windows 95	Windows NT	Mayoría de los sistemas UNIX, Windows NT, VAX VMS, Windows 95, OS/2, Macintosh	Mayoría de los sistemas UNIX, Windows NT, VAX VMS, Windows 95

Tabla 4.1 Cuadro comparativo de los principales Manejadores de Bases de Datos Relacionales.

Puede observarse de la tabla 4.1 que los RDBMS's que cumplen con la mayoría de las características deseables son Oracle y Sybase, ambos cuentan con un manejo bastante robusto del modelo de datos relacionales y un amplio soporte de múltiples objetos dentro de la base de datos. Los bloqueos han sido optimizados de tal forma que el acceso concurrente sobre un registro o conjunto de registros provoque la mínima contención posible. Estos dos RDBMS's proporcionan además un control de seguridad de nivel C2. Desde el punto de vista administrativo, ambos poseen herramientas que

facilitan el manejo del entorno, y permiten que se realicen los respaldos ya sean totales o incrementales. Se tiene además un amplio soporte para conectividad hacia otras bases de datos y el empleo de la replicación en cascada, aspecto extremadamente útil cuando se trabaja con aplicaciones de misión crítica. En el caso de Banco de México se emplea Sybase, dado que, además de las características mencionadas, se trata de uno de los productos que se han vuelto un estándar en el campo de las aplicaciones financieras. Por esta razón se empleará Sybase como RDBMS para la aplicación que se desarrollará en este trabajo.

4.1.2 Herramientas de desarrollo de aplicaciones

La elección de la herramienta de desarrollo es un aspecto que debe revisarse con cuidado ya que muchas veces puede ser el aspecto que lleve a la realización exitosa de un proyecto o a su fracaso. Por ejemplo, las herramientas que no pueden crecer, no serían una buena elección para un proyecto de desarrollo cliente/servidor que requiere el soporte para 1000 usuarios o más. De igual forma, las herramientas que no pueden reunir las expectativas de desempeño, llevarán al fracaso a la aplicación después de que ésta sea entregada a los usuarios. Finalmente, las herramientas que no son soportadas por otras herramientas, ponen la responsabilidad sobre el desarrollador de construir vínculos externos hacia las demás herramientas y repositorios de objetos, lo cual involucra una gran cantidad de tiempo.

Este apartado cubre las características de cuatro herramientas de desarrollo líderes en el mercado entre las que se encuentran: Borland Delphi, Microsoft Visual Basic y Powersoft PowerBuilder. Antes de observar algunas características de estos productos, es necesario revisar como se clasifican estas herramientas.

Una herramienta de desarrollo es un compilador, generador de reportes o programa que permite a los desarrolladores construir y 'liberar' aplicaciones de base de datos cliente/servidor. Aunque la mayoría de las herramientas pueden crear aplicaciones para el lado del cliente, sólo unas cuantas son capaces de distribuir la carga del proceso entre una o más computadoras y generar objetos que corran en el servidor de base de datos. Pudiera pensarse que todas las herramientas son totalmente diferentes; no obstante, pueden encontrarse algunas características en común. Por ejemplo, la mayoría incluye un ambiente integrado de desarrollo (IDE). Este IDE es el lugar donde los desarrolladores pueden construir la interfase y definir el comportamiento de ésta por medio de instrucciones programadas, el IDE típicamente proporciona una paleta de objetos, depuradores integrados, editores de código y muchas utilerías extra. También pueden ligar varias bases de datos usando capas de *middleware* incluidas dentro de la herramienta, tales como ODBC y JDBC.

Todas las herramientas proporcionan algún tipo de lenguaje de programación, permitiendo a los desarrolladores personalizar el comportamiento de la aplicación. Además, la mayoría soporta el enfoque de desarrollo orientado a objetos. Hoy en día, con el desarrollo cliente/servidor multicapa, las herramientas son capaces de comunicarse con capas tales como TP Monitors y Object Request Brokers (ORB's) para objetos distribuidos. Pueden también crear objetos propietarios permitiendo al desarrollador

particionar los objetos y distribuirlos entre diversos servidores, a estas herramientas se les conoce como herramientas de aplicación particionada.

Una característica de interés es la capacidad de manejar las actividades de bajo nivel, a diferencia de los lenguajes de programación de propósito general tales como C o C++, en la que se requiere interactuar directamente con los API's del sistema operativo, las herramientas avanzadas, esconden la mayoría de estas actividades de bajo nivel al desarrollador, esto significa que los desarrolladores pueden construir aplicaciones rápidamente.

Tipos de herramientas

Cada herramienta de desarrollo cae en una o dos de las siguientes categorías: 3GL, especializada, multiplataforma, base de datos orientada a archivo, generación de reportes, generadores de código, CASE, aplicación particionada y desarrollo de Web.

Puede verse dentro de los 3GL a los lenguajes tradicionales de programación de propósito general tales como C++, Pascal, Cobol y Fortran. Los mejores ejemplos en el mercado cliente/servidor son: Microsoft Visual C++, Borland C++, Symantec C++ y Borland Object Pascal. Estas herramientas de propósito general no están hechas específicamente para aplicaciones de base de datos, sin embargo, usualmente incluyen bibliotecas de funciones para manejarlas. Este tipo de herramientas ha avanzado considerablemente en los últimos años; sin embargo, los 3GL siguen siendo aún muy primitivos respecto a las herramientas especializadas. Los desarrolladores deben convivir con modelos de memoria, funciones de sistema operativo, y aún con el I/O directamente desde el lenguaje de programación. Aunque la tendencia ha sido mover al desarrollador lejos de una interfase primitiva, empleando GUI's, mecanismos de base de datos y bibliotecas de objetos avanzados. Una característica que tienen a su favor estos entornos de desarrollo es que al tratarse de verdaderos compiladores, producen código más eficiente en los ejecutables, lo cual les da una ventaja en desempeño.

Las herramientas son 'especializadas' porque están hechas con el propósito específico de crear aplicaciones de base de datos cliente/servidor. Dentro de este grupo puede encontrarse a PowerBuilder, Delphi y Visual Basic. Este tipo de herramientas se caracteriza por dar al desarrollador toda la funcionalidad para diseñar, construir y entregar una aplicación, también proporcionan lenguajes de programación de alto nivel (la mayoría un 4GL), un IDE, un depurador y una biblioteca de objetos que pueden usarse dentro de la aplicación. Por sus características, estas herramientas son preferidas para construir aplicaciones cliente/servidor, sin embargo, dado que usualmente utilizan intérpretes y no compiladores, puede perderse algo del desempeño de la aplicación.

Las herramientas multiplataforma son muy similares a las herramientas especializadas pero son capaces de liberar una misma aplicación en cualquier número de plataformas. Ejemplos de herramientas multiplataforma son JAM7 de JYACC Corp., Unify de Unify Corp. y Uniface de Compuware Corp. Los desarrolladores que emplean herramientas multiplataforma deben enfrentar aspectos de programación tales como las diferencias en los GUI's (presencia o ausencia de ciertos controles en pantalla), variables propias del sistema operativo e interfaces con éste, y la disponibilidad de soporte de capas como OLE, ODBC.

Las herramientas de base de datos orientadas a archivo incluyen productos tales como Microsoft Visual FoxPro y Access, Lotus Aproach y Borland Visual dBase. Estas herramientas están diseñadas para emplear bases de datos que residen en archivos. Debido a que los archivos no son verdaderos servidores de bases de datos, el procesamiento de la base de datos reside en las aplicaciones desarrolladas.

Aunque las herramientas de generación de reportes no están diseñadas para generar verdaderas aplicaciones de propósito general, habilitan a los desarrolladores a crear aplicaciones especializadas con el propósito de recuperar y procesar los datos. Para hacer esto, se permite a los desarrolladores diseñar visualmente los reportes y generar la secuencia de instrucciones SQL requeridas para recuperar información de la base de datos. Ejemplos de este tipo de herramientas son Report Smith de Borland y Cristal Report de Seagate Software.

Los generadores de código trabajan como las herramientas especializadas, pero estos generan código 3GL para los compiladores tradicionales en lugar de proporcionar sus propios mecanismos de 'liberación' de la aplicación. Un ejemplo de generador de código es ProtoGen de Protosoft.

Las herramientas CASE habilitan a los desarrolladores a diseñar y 'liberar' aplicaciones y bases de datos. Emplean sofisticados subsistemas de diagramación, permitiendo a los desarrolladores entender el sistema a nivel lógico, antes de generar los objetos de aplicación y el esquema de la base de datos. Las barreras entre el modelado y la elaboración de la aplicación están siendo superadas. Dos ejemplos de herramientas de modelado que generan aplicaciones para Visual Basic, PowerBuilder y otras herramientas de desarrollo son Rational Software de Rational Rose y Logic Works Erwin.

Las herramientas de aplicación particionada tales como IBI's Cactus, Fortés Forté y Dynasty dejan a los desarrolladores crear una versión lógica de una aplicación y entonces separan y distribuyen los objetos en los servidores que se tengan disponibles. A diferencia de las herramientas de desarrollo especializadas las cuales siempre usan el modelo cliente servidor de dos capas, las herramientas de aplicación particionada usan el modelo de *n* capas, permitiendo a los desarrolladores ejecutar objetos de aplicación en cualquier número de servidores o clientes. Emplean los mecanismos ORB y de mensajería para permitir a los objetos compartir información.

Finalmente, las herramientas de desarrollo de Web son la nueva categoría de herramientas, y realmente son nuevas versiones de las herramientas antes mencionadas y que se les ha añadido la funcionalidad de generar aplicaciones para uso en el Internet o en las Intranets. Para realizarlo, las herramientas Web habilitan tecnologías tales como HTML, CGI, NSAPI, ISAPI, Java y ActiveX. Ejemplos de estas herramientas pueden encontrarse en Microsoft Visual J++ y Symantec Café Pro. Ejemplos de herramientas que permiten construir aplicaciones cliente/servidor y de Web son Unify, Uniface y PowerBuilder. El punto clave de estas herramientas es que las aplicaciones se ejecutan en los *browsers* del Web, mientras que las aplicaciones tradicionales cliente/servidor tienen que construir sus propias interfaces y deben ser 'liberadas' mediante ejecutables con algunos archivos adicionales de soporte.

Capas Dentro de las Herramientas de Desarrollo de Aplicaciones

Como puede observarse, existen cientos de herramientas de desarrollo, cada una con su propio enfoque hacia el desarrollo de las aplicaciones. Antes de seleccionar una herramienta, deben analizarse las características comunes encontradas en la mayoría de las herramientas y entonces comparar de herramienta en herramienta. Aun cuando las herramientas pueden parecer muy diferentes, hay cinco capas básicas encontradas en la mayoría de ellas: la capa de acceso a la base de datos, la capa de repositorio, la capa de diseño de la interfase, la capa de programación y la capa de liberación.

La capa de acceso a la base de datos es un intermediario entre la base de datos y la herramienta, maneja todas las llamadas de bajo nivel a la base de datos. Esta capa debe ser independiente a la base de datos y capaz de comunicarse con cualquier tipo de base de datos disponible en el mercado como Oracle, Sybase e Informix. Típicamente esto significa que la herramienta debe conocer como cargar y descargar los *drivers* (manejadores) requeridos para accesar las bases de datos.

La capa de acceso a la base de datos deja que la herramienta obtenga los datos de varias maneras. Primero, pueden accesarse los datos como objetos nativos de la aplicación, sólo realizan manipulación de las propiedades de los objetos para trabajar con los datos. Segundo, pueden accesarse los datos con el tradicional esquema relacional. Lo cual permite que se trabaje con tablas, columnas, renglones y relaciones y no objetos. La mayoría de los desarrolladores encuentran esta forma más natural que el enfoque con objetos. Finalmente pueden accesarse los datos usando el *driver* nativo de interfase a través de la capa de acceso o simplemente saltándola y trabajando directamente sobre el API de la base de datos. Lo importante es que de esta última forma pueden invocarse servicios que de otra forma sería imposible llamar, tales como los stored procedures.

La capa de repositorio es sólo otro nivel de abstracción por debajo de la capa de base de datos. Aunque algunas herramientas tales como PowerBuilder, Uniface y Oracle Developer/2000 usan sofisticados repositorios, unas cuantas herramientas tales como Delphi y Visual Basic no los emplean (Visual Basic 5.0 incluye un repositorio). El alcance y capacidad de los repositorios varía grandemente de herramienta a herramienta.

La capa de repositorio permite el almacenamiento de los datos en la base de datos tales como reglas del negocio que pueden aplicarse a un atributo en particular de la base de datos, o un color o *font* para un atributo de la base de datos que pueda ser usado en la aplicación. Un repositorio puede definir también el comportamiento, tal como un evento que ocurre cuando un renglón es agregado a una tabla. Esto significa que trabajan de manera muy similar a los *stored procedures* y *triggers*. Algunas herramientas, tales como JAM 7, usan un repositorio para almacenar objetos de interfaces; ciertos repositorios pueden almacenar y procesar objetos de aplicación, además de definir parámetros de seguridad para una aplicación. Los repositorios pueden residir en el cliente, o pueden existir en la base de datos, permitiendo que el repositorio sea compartido por otros desarrolladores.

En muchos casos, los repositorios proporcionan algunas características de orientación de objetos a la herramienta. Debido a que los atributos de los elementos de datos son compartidos a través de la aplicación, simplemente cambiando el atributo en el repositorio, automáticamente se realizarán los cambios en la aplicación completa.

Como se mencionó anteriormente, la capa de diseño de la interfases es un subsistema de la herramienta que crea las pantallas, ventanas y menús con los cuales el usuario trabajará. Aunque difieren grandemente de herramienta a herramienta, todas las capas de diseño de la interfase proporcionan una paleta desde la cual pueden ser obtenidos los controles que servirán para diseñar los objetos. Se pueden modificar las propiedades de estos controles a través de las ventanas de propiedades y agregar código para definir su comportamiento dentro de la aplicación. Por ejemplo, las propiedades pueden controlar que sucede en la aplicación cuando se presiona un botón de *OK*, o cuando un usuario selecciona un registro dentro de una *data window*.

La capa de programación es el subsistema de la herramienta que permite al desarrollador personalizar la aplicación a través de un lenguaje de programación. La mayoría de las herramientas de hoy en día son manejadas por eventos, lo cual significa que el comportamiento de la aplicación depende de que el usuario 'dispare' algún evento mediante la interacción con la interfase, por ejemplo, al buscar algún dato en una base de datos o imprimiendo algún reporte. Típicamente los desarrolladores ponen código detrás de los controles usando editores de código que son accesibles desde la capa de interfase. El tipo de código varía de herramienta en herramienta. Por ejemplo, PowerBuilder utiliza PowerScript, se trata de un 4GL parecido a Basic; y Visual Basic emplea VBA (Visual Basic for Applications), el cual es una versión mejorada de Basic. Algunas herramientas emplean tradicionales 3GL's, tales como Optima++, la cual usa C++; y Delphi, la cual utiliza Object Pascal. Más allá del editor de código, esta capa también proporciona capacidades de depuración de código.

La capa de liberación permite trasladar la aplicación en algo que el usuario pueda ejecutar. Típicamente esto significa que la herramienta pueda crear código P y proporcionar un intérprete de este código, los cuales deben existir en el lado del cliente para poder habilitar la aplicación. Hoy en día, sin embargo, la tendencia es emplear verdaderos compiladores que habiliten a la aplicación para que pueda ejecutarse como un programa nativo ejecutable. Las ventajas de emplear verdaderos compiladores residen en el desempeño y en el uso más eficiente de los recursos. El uso de un verdadero compilador fue una de las características que hacen que herramientas como Delphi sean productos sobresalientes en el mercado. Muchas herramientas especializadas, incluyendo PowerBuilder y VisualBasic, se están moviendo hacia el uso de verdaderos compiladores.

En la tabla 4.2 se encuentra un cuadro comparativo que resume las principales características de los productos más comerciales en México. En esta tabla puede observarse que todos los productos cuentan con características que son deseables para la mayoría de las aplicaciones cliente; sin embargo, para nuestro proyecto utilizaremos Powersoft PowerBuilder 5.0. Esta es una herramienta en la que se han optimizado el acceso a bases de datos Sybase y la manipulación de los objetos, debido a que Powersoft es una división de software de Sybase. De esta forma se logrará que el soporte para el software del cliente y del servidor sea proporcionado por el mismo fabricante, minimizando de esta forma la posibilidad de errores o bugs por incompatibilidades entre el RDBMS y la herramienta de desarrollo.

	Borland Delphi Client/Server Suite 2.0	Borland Visual dBase 5.5	Informix NewEra 2	Microsoft Access '95	Microsoft Visual Basic 4	Powersoft PowerBuilder 5.0
PLATAFORMA						
Desarrollo Cliente	Windows Win95 Windows NT	Windows	Motif Windows	Win95 Windows NT	Windows win95 Windows NT	Mac Windows Win95 windows NT UNIX
Desarrollo Servidor	Windows Win95 Windows NT	Windows	Motif Windows	Win95 Windows NT	Win95 Windows NT	Mac Windows Win95 Windows NT UNIX
Despliegue Cliente	Windows Win95 Windows NT	Windows	Motif Windows	Win95 Windows NT	Windows win95 Windows NT	Mac Windows Win95 Windows NT UNIX
Despliegue Servidor	Windows Win95 Windows NT	Windows	UNIX	Win95 Windows NT	Win95 Windows NT	Mac Windows Win95 Windows NT UNIX
Bases de Datos Soportadas	Borland InterBase OBJ Informix MS SQL Server Oracle Sybase	Informix MS SQL Server Oracle Paradox Sybase Xbase	Propietaria	Excel FoxPro Lotus 123 Paradox Xbase	Access Foxpro Oracle Paradox Sybase	OBJ Informix MS SQL Server Sybase
Interfaz con la Base de Datos	ODBC	ODBC	DRDA ESQL/C/ESQL	ODBC	ODBC	ODBC
DESARROLLO DE APLICACIONES						
Generación	Propietario SQL	Propietario SQL	C propietario	SQL VBA	Basic	HTML propietario SQL
Ejecución de Compilación	Compilación a Código Máquina	Compilación a Código intermedio	Compilación a Código intermedio y Máquina	No	Compilación a P Código	Compilación intermedio a Código máquina
Ejecución de interprete	No	Si	Si	Si	No	Si
Soporte para multihilos	Si	No	No	Si	No	Si
Nivel de herencia	Bastante	Bastante	Bastante	Ninguno	Ninguno	Bastante
Soporte de MDI	Si	Si	No	Si	Si	Si
Objetos de terceros	Activex DLL VBX Librerías de Objetos Visuales	DLL VBX	C++ NewEra	Activex	DLL OLE Control OLE server	Activex C++ VBX
OLE	Control Activex drag-and-drop fuente y destino control UCX OLE Server.	Drag and drop fuente y destino	Ninguno	Drag-and-Drop fuente y destino	Drag and Drop fuente y destino	Drag and drop fuente y destino
ESCALABILIDAD						
Aplicaciones particionadas	Si	No	Si	No	Si	Si
Conexión de llamadas a Portadimientos almacenados	Informix InterBase ODBC Oracle Sybase SQL Server	Borland InterBase Oracle Sybase	Propietaria	Ninguna	Ninguna	OBJ Informix ODBC Oracle Sybase SQL Server
Generación de Procedimientos embebidos	OBJ Informix InterBase SQL Server Oracle	Borland InterBase Oracle Sybase	Ninguna	Solo para reglas de validación de Access	No	Ninguna
Soporte de TP monitor	CICS Encina Top End	Ninguno	Ninguno	Ninguno	Encina Tuxedo	CICS Encina Tuxedo

Tabla 4.2 Características principales de los productos más comerciales en México.

En el Banco de México se emplea PowerBuilder como plataforma principal de desarrollo de aplicaciones cliente, actualmente se encuentra en proceso de migración de sus aplicaciones basadas en *maniéramos* a aplicaciones cliente/servidor empleando Sybase System 11 como RDBMS y PowerBuilder versiones 4.0 y 5.0 en la parte del cliente.

4.1.3 Herramientas de Procesamiento de Scripts

Podemos observar que existen algunas tareas que deben realizarse como parte de una rutina todos los días. Para este tipo de tareas sería deseable alguna forma de poderlas automatizar, es en estos casos en los que las herramientas de procesamiento de scripts pueden ayudar. En el caso del sistema de distribución y actualizaciónt de software es necesario la utilización de una herramienta que realice la instalación, para hacerlo se necesita que la herramienta prepare el entorno para el correcto funcionamiento, lo cual incluye la revisión de los archivos de configuración de Windows 3.11 ó alguna posible modificación del registro cuando se trate de Windows 95 y genere el grupo de programas (en el caso de Windows 3.11) ó los accesos directos (en el caso de Windows 95).

En esta sección analizaremos siete productos que están disponibles en el mercado. Cuatro de ellos - Cenvi 2.11 de Nombas, Personal REXX 3.5 de Quercus Systems, PowerScript 1.0 de Desiderata Software y Winbatch 96B de Wilson WindowWare - son lenguajes de procesamiento por lotes (batch) que produces scripts en ASCII. Estos scripts son usualmente ejecutados dándole al script una extensión específica de un lenguaje y lanzándolo mediante doble-click sobre un acceso directo o un programa. Dos de los productos - SmartPad 3.5 de Softblox y Virtual Desk 2.0 de Desk - emplean un esquema orientado al procesamiento de macros. A algunos usuarios les gusta explotar la capacidad de estos productos al enviar *keystrokes*[3] o comandos de menú a las aplicaciones sin la necesidad de escribir alguna línea de código. El último producto, Prospero de Oberon, es una herramienta visual en la cual el flujo del programa es definido arrastrando líneas entre objetos en la pantalla, en vez de escribir líneas de código.

Aunque la automatización debiera ser proporcionada por el sistema operativo (DOS incluye lenguaje de procesamiento por lotes desde la versión 2.0), Windows nunca ha incluido un lenguaje de este tipo interconstruido dentro de su sistema operativo. Windows 3.x tenía un grabador de macros de teclado, pero fue olvidado en la transición hacia Windows 95 (una excepción a esta tendencia puede encontrarse en la nueva funcionalidad de Windows 95 al procesar el lenguaje batch de DOS, que incluso puede iniciar aplicaciones de Windows).

En la evaluación de las herramientas, se encontró que había una gran cantidad de características que por su utilidad podían ser consideradas críticas. Inicialmente se observó la flexibilidad proporcionada por lenguaje en términos de variables y estructuras de control. Puede verse una gran variedad, desde elegantes lenguajes parecidos a C, hasta sintaxis de código rápido y sucio. Como característica general puede encontrarse que el soporte para verdaderos arreglos es casi nulo.

También se buscó que cada producto dejara el control a las aplicaciones de Windows, integrándose a ellas y pasando los datos entre ellas. Enviar *keystrokes* y acciones del *mouse* puede ser útil, pero las llamadas a los menús directamente puede resultar en un código más robusto. La automatización de OLE (Object Linking and Embedding), soportada por cuatro de los siete productos, hace posible intercambiar objetos especiales (llamados *rich data objects*) entre aplicaciones con relativa facilidad.

Todas las herramientas proporcionan interfaces para escritura de código, pero sólo dos generan código basado en acciones grabadas. Sorprendentemente, muchos de los productos han dejado a un lado al editor de cajas de diálogo, haciendo más difícil la construcción de interfaces para capturar datos introducidos por el usuario. Algunos de estos lenguajes proporcionan encapsulamiento de las cajas de diálogo comunes de Windows. También se encontró que las facilidades de depuración varían desde no existente hasta módulos adecuados para proporcionar *breakpoints*, *trace* y ventanas de observación de variables. A continuación se encuentra una descripción de cada una de las herramientas revisadas.

[3] Se le llama *keystroke* a la presión de una tecla.

CEnvi

CEnvi 2.11 es un intérprete para el lenguaje propietario de la compañía Nombas, el Cmm (C minus minus), es un dialecto simplificado de C. En Cmm, las variables no están fuertemente tipificadas o declaradas, además de no contar con apuntadores. Tampoco existen estructuras definidas explícitamente, aunque pueden utilizarse tantas estructuras como se necesiten.

El resultado es un lenguaje interpretado con mucha de la potencia de C permitiendo las malas costumbres de programación del antiguo BASIC. Cenvi soporta la mayoría de las funciones de la biblioteca estándar de C y agrega otras funciones útiles interconstruidas. También proporciona bibliotecas de funciones adicionales que pueden ser incluidas dentro de los programas mediante la directiva `#include`. Estas bibliotecas encapsulan un gran número de llamadas a API's de Windows, dando al programador de Cmm un rápido acceso a servicios tales como invocación de un menú en cualquier aplicación que se esté ejecutando. Las bibliotecas son útiles, pero no ofrecen soporte en la verificación de errores y la documentación está limitada a simples comentarios en los archivos .LIB.

CEnvi omite algunas características importantes, incluyendo la grabación de eventos y el editor de cajas de diálogo, tampoco incluye un depurador aunque se anuncia que será incorporado en la siguiente versión. Las únicas formas de ejecutar los scripts son mediante doble click del ratón o empleando el comando Run, también puede activarse a través de accesos directos. Están disponibles versiones de CEnvi para DOS, OS/2, Windows 3.1 y una gran variedad de Unix. Además está en desarrollo una versión de Cmm para el Web. Los scripts de CEnvi pueden ser compilados para obtener archivos .EXE; aunque se requiere una licencia para distribuirlos.

Personal REXX

Personal REXX está disponible para Windows NT y Windows 3.1. Quercus recomienda ejecutar la versión de 16 bits para Windows 95 debido a algunos problemas detectados con la versión de 32 bits. Este lenguaje puede resultar atractivo para aquellos que buscan una herramienta de manipulación de patrones y de texto, áreas que son la especialidad de REXX. El lenguaje cuenta con variables sin tipo y variables apuntadas, las cuales son un excelente reemplazo a los arreglos y registros usados por otros lenguajes. Una variable apuntada tal como `foo.1.2`, por ejemplo, puede reemplazar un arreglo del tipo `foo[1,2]`. Personal REXX realiza substitución de variables en cualquier parte después de un punto, por lo que `foo.i.j` puede ser interpretado como `foo.1.2` si `i=1` y `j=2`.

La depuración en Personal REXX soporta el mecanismo de *trace*, no hay una ventana de observación de variables, pero puede examinarse el valor de cualquier variable en el *prompt* desplegado entre los pasos del *trace*.

Personal REXX da acceso a un número de funciones específicas de Windows que permiten enviar *keystrokes* o mensajes de Windows a cualquier programa. Pero ofrece menos control sobre Windows y las aplicaciones de Windows que ningún otro producto revisado en esta sección. Pueden escribirse cajas de diálogo sencillas que regresen un

simple SI/NO o mostrar una caja de entrada de texto, pero no existen mecanismos para construir complejas interfaces con el usuario.

PowerScript

PowerScript 1.0 es una herramienta de procesamiento de scripts para Windows 95 y Windows 3.1 que emplea un lenguaje propietario con sintaxis parecida a C. El lenguaje emplea variables con tipo que deben ser predefinidas y ofrece los tipos básicos de datos estructuras de control. No implementa los arreglos ni la estructura de control case, pero la manipulación de cadenas y patrones es bastante robusta.

Una característica interesante del lenguaje es lo que PowerScript llama *iterators*. Estas son funciones, las cuales, usadas en un ciclo For, permiten que se ejecuten a través de todas las instancias que el *iterator* encuentra en el ciclo. Por ejemplo, puede utilizarse el *iterator Files* en el inicio de un ciclo `For t in FILES (*.*,dll)` donde `t` es una variable de tipo cadena y que es sucesivamente definida para cada DLL en el disco. Otros *iterators* son proporcionados para buscar líneas en un archivo texto y para líneas en un texto del Portapapeles.

PowerScript no deja crear o mostrar elaboradas cajas de diálogo pero proporciona las funciones para obtener entradas de tipo cadena o entradas de una caja de diálogo conteniendo un sólo grupo de controles *radio buttons*. No hay un depurador explícito para realizar el *trace*, pero el paquete incluye un editor que ilumina la línea de un script que se está ejecutando.

Uno de los aspectos importantes de este producto es la capacidad de comunicarse directamente con otros programas. Pueden enviarse *keystrokes*, selecciones de menú y aún mensajes de Windows a ventanas y cajas de diálogo de las aplicaciones, sin embargo la automatización de OLE y DDE no está soportada. Otra característica importante es la capacidad de *iconizar* un script sobre el escritorio, lo cual permite que arrastrando y soltando objetos sobre el icono, por ejemplo archivos, se inicie la ejecución del script.

Prospero

Prospero versión 1.0 toma un enfoque visual para la integración y automatización de aplicaciones. Trabajando en un ambiente de desarrollo orientado a proyectos, se emplea el entorno visual para construir scripts a partir de objetos de aplicación. Para lograrlo se arrastran líneas entre los objetos que definan la lógica de programación. Se inicia arrastrando un bloque de construcción de un programa específico de una paleta que está disponible en el escritorio.

Prospero extiende las capacidades visuales con un rico conjunto de bloques que incluyen a las instrucciones de control *If-Then-Else* y a las iteraciones. También incluye un bloque de cajas de diálogo que incluye un editor de este tipo de objetos. El bloque *Function* toma entradas y ejecuta algún número de funciones. El bloque *Windows Common Dialog* permite implementar una ventana completamente compatible con el estándar de Windows. También se incluye un modo de depuración.

Ya que el propósito de Prospero es la automatización de otros programas, los bloques que encapsulan otros archivos y programas son cruciales. La versión 1.0 incluye bloques para Excel, Lotus Notes, Word y un modelo general de OLE, también se incluye una gran variedad de fuentes de datos incluyendo Access, dBase y bases de datos SQL. La versión 1.1 incluirá bloques para páginas Web, Microsoft Project y un soporte mejorado de OLE.

El enfoque de Prospero frecuentemente simplifica tareas complejas, como por ejemplo la transferencia de datos de Access a Word, mediante las conexiones entre los bloques de Prospero pueden detectarse errores fácilmente y encapsularse la mayoría de las dificultades. Por otra parte, Prospero puede hacer que lo sencillo sea difícil. Si se requiere repetir una tarea cuatro veces, se requiere colocar y enlazar tres iconos: uno para definir e incrementar la variable, uno para probar la condición y uno para ejecutar la iteración.

SmartPad

SmartPad versión 3.5, viene en dos variaciones: SmartPad Pro, la cual es una poderosa herramienta para crear aplicaciones y SmartPad Desktop, un *kit* que permite a los usuarios finales ejecutar aplicaciones o automatizar algunas tareas simples. SmartPad Pro incluye un completo entorno de desarrollo (IDE), colocando a esta herramienta como la mejor de las evaluadas en esta sección.

SmartPad Pro permite crear scripts y asignarlos a botones en las versiones Pro y Desktop. El lenguaje es una variante de Basic, con variables y estructuras de control estándar. Las variables tienen tipos, pero como en el tradicional BASIC, éstas no necesitan ser predeclaradas.

Una amplia variedad de funciones permiten controlar aplicaciones de Windows y las comunicaciones entre ellas. Los scripts de SmartPad pueden manejar OLE y comandos DDE, y SmartPad puede por sí mismo ser controlado por otra aplicación o por sus propios scripts por medio de DDE. Puede accesarse el Portapapeles, los comandos del menú de cualquier aplicación y enviar *keystrokes* o acciones del mouse.

El IDE de la versión Pro proporciona un completo depurador, incluyendo una ventana de visualización de variables y soporte para *breakpoints*. Pueden grabarse macros con acciones del *mouse* y teclado y traducirlas a acciones de alto nivel como selecciones del menú. También incluye un editor de cajas de diálogo que permite la edición visual y mediante código de sofisticadas interfaces con el usuario.

El usuario final arranca acciones de SmartPad con botones sobre una paleta de botones de SmartPad, las cuales son específicas a la aplicación y pueden flotar o colocarse sobre cualquiera de los cuatro lados de una ventana de aplicación. Un botón puede tener asignado un *hotkey*, y puede ser configurado como de inicio automático o de terminación automática, también pueden dispararse scripts por la detección de ciertas condiciones en el entorno de las aplicaciones. Sin embargo, no hay una forma simple de ejecutar un script por medio de un acceso directo o del comando Run.

Los usuarios sin acceso a la versión Pro pueden asignar cinco tipos de acciones a un botón: correr una aplicación, enviar *keystrokes* o comandos del menú a una aplicación,

ejecutar una macro de teclado o del *mouse*, ejecutar un comando DDE e intercambiar la paleta de botones.

Virtual Desk

En el ambiente de Virtual Desk versión 2.01, el escritorio se convierte en un ambiente vivo en el cual pueden colocarse objetos que reaccionan a un rico lenguaje de scripts. Sin embargo, para un ambiente de aplicaciones convencional, se ha encontrado un lento performance y algunos detalles que lo hacen poco funcional.

De la hoja de propiedades de cada objeto, pueden asignarse scripts a una gran variedad de acciones incluyendo el click y doble-click, arrastrar y soltar objetos. Estos scripts pueden ser accesados a través de botones en las hojas de propiedades que abren el script en el IDE.

Virtual Desk incluye un depurador y una ventana de observación de variables, sin embargo no es tan poderoso como el implementado en SmartPad. No incluye un editor de cajas de diálogo. La sintaxis del lenguaje es un punto intermedio entre C y BASIC. Las funciones para manipulación de archivos están presentes, pero la automatización OLE, DDE y el control de menús en otros programas no está soportado.

Puede observarse que es un software con un pobre performance, además de que la documentación y la ayuda en línea contienen algunas irregularidades.

WinBatch

WinBatch es uno de los precursores en el ámbito de los lenguajes de procesamiento de scripts, su lenguaje *Windows Interface Language* (WIL) fue introducido al mismo tiempo que Windows 3.0. Hoy en día WinBatch es una poderosa herramienta que se encuentra disponible para un gran número de plataformas incluyendo Windows 3.1, Windows 95, Windows NT sobre x86, Alpha, Mips y PowerPC. La herramienta WinBatch+Compiler permite compilar los scripts WIL en ejecutables que pueden ser distribuidos libremente.

El lenguaje WIL tiene considerable poder y flexibilidad, sin embargo carece de soporte para arreglos, manejando cadenas delimitadas por tabuladores. Las funciones *ItemCount* e *ItemExtract* son útiles para la manipulación de tales cadenas, aunque estas funciones no pueden emplearse cuando se trata de arreglos multidimensionales.

WIL tiene la riqueza de funciones interconstruidas para interactuar con aplicaciones Windows, incluyendo la capacidad de enviar *keystrokes* y selecciones del menú. La única capacidad de grabación es para *keystrokes*. WinBatch ofrece un modo de depuración que permite ir paso a paso sobre el código obteniendo el valor de cualquier variable. Sin embargo, no proporciona una ventana formal de observación de variables, ni saltos sobre subrutinas ni *breakpoints* condicionales. Incluye un editor de cajas de diálogo.

Los scripts de WinBatch son archivos de texto con la extensión .WBT; pueden ejecutarse mediante doble-click sobre el icono del archivo. Pueden agregarse scripts a

los menús contextuales de los archivos vistos en el Explorador o localizados en el escritorio, o puede definirse un menú de archivos *batch* para ejecutarse desde un ícono en el área de notificación de la barra de tareas de Windows 95.

El compilador WinBatch+Compiler permite generar archivos .EXE standalone y distribuirlos aún a aquellos equipos que no tienen WinBatch, haciendo a WinBatch+Compiler una opción atractiva y económica.

A continuación se muestra la tabla 4.3 que resume las características encontradas en cada uno de los productos.

	CEnvi 2.11	Personal REXX 3.5	Power Script 1.0	Prospero 1.0	SmartPad 3.5	Virtual Desk 2.0	WinBatch 96B
Lenguaje							
Arreglos				✓	✓	✓	
Variables apuntadas o listas de variables	✓	✓	✓				✓
DDE	✓			✓	✓		✓
Automatización OLE	✓			✓	✓		✓
Herramientas de desarrollo							
Editor de scripts o IDE		✓	✓	✓	✓	✓	
Grabación de *keystrokes* o clicks de mouse					✓		✓
Grabación de selecciones del menú					✓		
Editor de cajas de diálogo				✓	✓		✓
Depuración							
Trace		✓		✓	✓	✓	✓
Breakpoints				✓	✓	✓	✓
Ventana de visualización de variables					✓	✓	
Métodos de arranque de scripts							
Por medio de extensión	✓	✓	✓	✓			✓
Botón u objeto seleccionable con el mouse					✓	✓	
Menu pop-up					✓	✓	✓
Arrastrar y soltar			✓			✓	

Tabla 4.3 Características de las herramientas de procesamiento de scripts.

De las siete herramientas de procesamiento revisadas, sobresalen dos: SmartPad de SoftBlox y WinBatch de Wilson WindowWare, ambos proporcionan a los usuarios novatos y expertos la capacidad de utilizar botones para iniciar scripts. Los expertos pueden verse atraídos por SmartPad Pro para poder emplear su poderoso lenguaje de scripts, además de las facilidades incluidas como el IDE y el depurador.

WinBatch tampoco se queda atrás, su sintaxis está perfectamente adaptada para proporcionar una completa funcionalidad para las aplicaciones de Windows. También proporciona un editor de cajas de diálogo, un grabador de macros y muchos mecanismos para arrancar un script. El WinBatch+Compiler permite compilar y distribuir las aplicaciones libremente, es decir, sin tener que pagar un cargo por cada versión compilada.

De estos dos últimos productos mencionados se ha seleccionado finalmente WinBatch por ser el que actualmente se emplea en la automatización de tareas para aplicaciones Windows dentro del Banco de México.

4.2 Análisis y Selección del Hardware

4.2.1 Elección de la Plataforma para el RDBMS

En muchos aspectos la plataforma en la que se ejecuta el RDBMS determina el desempeño completo del servidor de base de datos, el RDBMS también depende de las capacidades proporcionadas por el sistema operativo para accesar eficientemente al disco, al caché y a los servicios de red. Por lo tanto, seleccionar la plataforma para el RDBMS es casi tan importante como seleccionar el RDBMS.

Aunque existen algunas excepciones, los servidores basados en SPARC, x86 y RISC, dominan el mercado, incluyendo todas las variedades de Unix y a Windows NT como proveedor de servicios de sistema operativo en versiones de uno y varios procesadores. Debido a la arquitectura interna de Unix, este sistema operativo es capaz de proporcionar verdaderas capacidades de multitarea y multihilo a sus aplicaciones nativas (incluyendo a los RDBMS's). Unix es vendido bajo muchos nombres para diversos procesadores incluyendo SCO Unix, UnixWare, Solaris para SPARC y x86, HP-UX y AIX para procesadores RISC.

Para explotar las mejores capacidades de Unix, los proveedores de RDBMS's como Oracle, Informix y Sybase han empleado las características de Unix de soportar numerosas conexiones de clientes como procesos *standalone* o hilos de un proceso. También han empleado el excelente desempeño de I/O, administración de memoria y las capacidades de administración de tareas.

El desempeño es todo en el mercado de los RDBMS's, para mantener y mejorar el performance, los proveedores de RDBMS's han implementado acceso a los recursos directamente, saltando al sistema operativo, un ejemplo de esto es la forma en la que algunos RDBMS's escriben al disco físico en las llamadas particiones crudas, en vez de hacer uso del sistema de archivos nativo de Unix y de esta forma se elimina la carga producida por las llamadas del sistema operativo empleadas al accesar el disco de la forma tradicional. La única desventaja de emplear particiones crudas reside en el hecho de que se tiene que emplear la utilería proporcionada por el RDBMS para realizar los respaldos. Otro truco empleado para mejorar el desempeño reside en correr el RDBMS como un proceso al nivel del *kernel*, algunas veces conocido como Ring 0. Algunos servidores de base de datos basados en Unix pueden hacer esto, aunque el mejor ejemplo se puede encontrar en Sybase ó Oracle ejecutándose en servidores NetWare como NLM's (*NetWare Loadable Modules*).

Como se menciona en el apartado 4.1, se ha seleccionado Sybase como manejador de base de datos para apegarse al estándar impuesto por Banco de México. En esta institución se cuenta con un servidor Sun Ultra Enterprise 4000 de la compañía Sun Microsystems para soportar el servidor de base de datos Sybase. Sun Microsystems es una empresa que ha ido tomando fuerza en los últimos años, el 60% de sus ventas se enfoca al mercado de servidores de misión crítica, y de esta cifra, el 25% se vendió al mercado gubernamental mexicano. La mayor base instalada de Sun en el sector público se centra en el INEGI, aunque otros de sus importantes clientes son la CFE, el ISSTE, la SCT e instituciones financieras entre las que se encuentra Banco de México. Cabe mencionar que la elección de Sun como proveedor de la plataforma para el RDBMS, se debe al rendimiento en el sistema de I/O logrado con la nueva arquitectura de la familia Ultra Enterprise, el cual es diez veces superior que el que puede obtenerse en el sistema

HP equivalente, se trata del HP9000 T520 y más de 16 veces que el obtenido por el mejor servidor de la línea SPM de IBM, el RS6000 R30; incluso rivaliza con las mejores computadoras de la línea CMOS de IBM.

Las principales ventajas de los servidores Sun se concretan en cuatro puntos: confiabilidad, disponibilidad, velocidad y crecimiento. Más allá de los adelantos relacionados con su tecnología, la empresa ha implementado desde hace dos años un cambio en sus estrategias de manera que ahora son ellos mismos los encargados del servicio y respaldo, a pesar de realizar todas sus ventas con asociados especialistas en el mercado, capaces de poder montar una solución de principio a fin. A continuación se muestra una descripción de los servidores enterprise 4000.

El equipo Sun Ultra Enterprise 4000 es el punto medio del último conjunto de servidores de 64 bits de Sun Microsystems con capacidad de hasta 30 nuevos procesadores Ultra-SPARC y con un innovador sistema de I/O ultra rápido llamado *Gigaplane central I/O bus*. La nueva familia de procesadores consta de cuatro miembros, se trata de los servidores Ultra Enterprise 3000, 4000, 5000 y 6000, todos ellos pueden ser utilizados como servidores Web tanto para Internet como para Intranet. Estos servidores están enfocados para cumplir con las expectativas futuras del hardware de UNIX basado en RISC, se trata en esencia de servidores con procesadores más rápidos, velocidad de I/O y ancho de banda del *bus* mejorados. Aunque Sun ha tardado en construir la versión de 64 bits de Solaris[4] que pueda aprovechar totalmente las capacidades del hardware mejorado, estos servidores muestran mejoras notables en el desempeño con las versiones beta de Solaris 2.5.1[5]. Esta versión administra mejor el multiprocesamiento y aprovecha las ventajas del *bus* de I/O. Se dice que esta familia de servidores está lista para recibir a la siguiente generación de sistemas operativos de 64 bits al poder manejar instrucciones y datos de 64 bits en el *bus*.

En las pruebas de desempeño se he mostrado como el *bus Gigaplane* es capaz de transferir 4 instrucciones de 64 bits (256 bits) en cada ciclo de reloj. El *bus Ultra Port Architecture* (UPA) conecta al *Gigaplane* con los periféricos consiste de un *bus* con capacidad de transferencia de 128 bits. Este *bus* está compuesto de tres conexiones, las dos primeras se conectan a tarjetas periféricas de 128 bits y la tercera que la conecta a la memoria principal. Estas conexiones pueden ser configuradas por software para trabajar con dispositivos periféricos de 64 bits tales como las tarjetas de I/O. Este *bus* corre a 83.3 MHz, la misma velocidad del *Gigaplane*.

Las tarjetas de I/O usan la misma arquitectura de *bus* y los mismos circuitos que el UPA, excepto que estos circuitos están diseñados para trabajar sólo con los 64 bits menos significativos, lo cual permite que los datos e instrucciones de 64 bits puedan ser utilizados a través de la arquitectura existente del sistema.

La arquitectura del sistema Sun Ultra Enterprise 4000

El sistema Sun Ultra Enterprise 4000 puede emplear entre uno y 14 procesadores de 167 MHz[6]. La arquitectura es extremadamente modular, se trata de un *backplane* que

[4] Sistema operativo de Sun.
[5] La versión 2.5.1 de Solaris es aún de 32 bits.
[6] Sun libero en el mes de mayo de 1996 procesadores a 300 MHz que pueden emplearse en la arquitectura de la familia Ultra Enterprise.

puede contener hasta ocho tarjetas controladoras de CPU's y de I/O, cada tarjeta puede contener hasta 2 CPU's , o dos controladoras de I/O; esto permite que pueda tenerse una gran variedad de combinaciones que debe contener al menos una tarjeta controladora de CPU's y una tarjeta controladora de I/O

Cada tarjeta controladora de CPU's puede albergar hasta 2 CPU's, cada CPU viene acompañado de un disipador de calor, 8 KB de SRAM para el *instruction cache*, 8 KB de SRAM para el *data cache*, y suficientes *slots* de memoria para soportar hasta 2 GB de memoria DRAM.

Las tarjetas controladoras de I/O, se componen de un *bus* central con un ancho de banda que puede mantener una velocidad de transferencia de 2 5 GB por segundo. Se trata de un *bus* de 256 bits con un diseño en el que se encuentran separados líneas de datos y direcciones, los cuales permiten flujos de datos a grandes velocidades, ya que se pueden obtener datos y direcciones de forma simultánea al no compartir las mismas líneas dentro del *bus*.

El *Gigaplane* interactúa con los dispositivos a través de las conexiones del UPA, este *bus* está compuesto de un canal de 128 bits con controladores separados de datos y direcciones que proporciona acceso al *Gigaplane* desde el CPU, la memoria y las tarjetas de I/O.

Cada tarjeta de I/O incorpora 4 puertos de UPA, dos de los cuales están conectados con el *Gigaplane* y los otros dos conectan al *Sbus*. Con esto se logra que cualquier dispositivo *Sbus* tenga acceso al *Gigaplane*.

El sistema incluye una tarjeta de gráficos que también sirve como interfases para dispositivos fast SCSI y Ethernet 10/100 Mbps, además de incluir dos interfases de fibra óptica.

En la Enterprise 4000, el almacenamiento masivo es externo a la unidad y se encuentra en un gabinete separado. Puede trabajar con los esquemas de discos RAID para tolerancia a fallas y redundancia de datos, tiene además una capacidad de 60 arreglos de discos logrando una capacidad aproximada de 19.4 TB.

Toda la serie emplea el sistema operativo Solaris 2.5.1, el cual está adaptado para dar soporte al multiprocesamiento. De interés a los usuarios y administradores del sistema es que esta versión de Solaris incorpora el software X/Open's Common Desktop Environment 1.0, que está destinada a ser un estándar en el campo de los entornos gráficos de usuario.

Además viene incluido un conjunto de software de administración de sistemas que está formado por: SyMon System Health Monitor 1.0, Solstice Disksuite 4.1 y Solstice Adminsuite 2.2. En el caso de Symon, se trata de agentes que recolectan información del servidor y el despliegue de estos datos puede realizarse en cualquier equipo Sun conectado en la red, como una estación de trabajo SPARC 20. Disksuite proporciona el soporte de administración de discos para emplear esquemas RAID de tolerancia a fallas como son: *disk mirroring* y RAID 5.

El equipo Ultra Enterprise 4000 contempla un gran número diagnósticos del hardware. Esto puede observarse durante el arranque de la máquina, además pueden introducirse comandos en el PROM de la máquina para realizar otro tipo de pruebas para diagnóstico de problemas. Existe un alto grado de tolerancia a fallas incluido en el sistema, muchos de los componentes pueden ser reemplazados sin necesidad de apagar el sistema, este es el caso de las tarjetas I/O.

Los precios de un equipo Ultra Enterprise 4000 varían dependiendo de la configuración. La configuración mínima cuesta aproximadamente $77,000 USD, y una configuración incluyendo 4 CPU's, 512 MB de memoria, 2 tarjetas de I/O y un monitor de 21 pulgadas puede costar alrededor de $150,000 USD.

A continuación se muestra la tabla 4.4 que muestra las características principales del servidor Ultra Enterprise 4000.

Procesador	
Número de procesadores	1 a 14
Arquitectura	Superescalar SPARC Version 9, UltraSPARC
Caché por procesador	Primario: 16 KB por instrucción, y 16 KB de datos interno Secundario: 1 Mb ó más en el caché externo
Interfase del CPU	De 1 a 14 slots UPA de 128 bits
Número de tarjetas	Máximo 8 tarjetas por sistema, la configuración mínima requiere una tarjeta de CPU/Memoria y una tarjeta de I/O
Tarjeta de CPU/Memoria	Soporta hasta dos procesadores y 16 slots para SIMM's de memoria
Tarjeta de I/O de Sbus	Ofrece dos canales Sbus, tres slots Sbus, SunFastEthernet, fast/wide SCSI, dos interfases de fibra óptica
Tarjeta de I/O grafica Sbus	Ofrece un canal Sbus, dos slots Sbus, un slot UPA para gráficos Creator y Creator3D
Tarjeta de I/O PCI	Ofrece cuatro canales PCI, dos slots PCI, SunFastEthernet, fast/wide SCSI
Memoria Principal	
256 MB a 14 GB de capacidad de memoria por sistema Opciones de expansión de memoria de 256 MB y 1GB (por cada grupo de 8 SIMM's) Hasta dos opciones de expansión de memoria por tarjeta	
Interfases Estándar	
Serial	Dos puertos RS-232/423
Sbus	Bus de datos de 64 bits, 25 MHz
PCI	Bus de datos de 64 bits, 66 MHz
Puertos de teclado y mouse	Uno por sistema
Ethernet	10/100 Mbps par trenzado (10-BaseT y 100-BaseT) o tansceiver MII por tarjeta de I/O
Canal de fibra óptica	Dos interfases por tarjeta de I/O (25 Mbps full duplex)
Unidades de Almacenamiento Interno	
Discos internos	Hasta cuatro tarjetas controladoras de disco con dos discos duros SCSI-2 de 3.5" y dos de 1" (de 2.1 GB formateados)
Cinta	Unidad de cinta de 8 mm y 20GB de capacidad, unidad de cinta DDS3 de 12-24 GB, unidad de cinta de 8 mm 14 GB ó unidad de cinta de 0.25" a 2.5 GB
CD-ROM	Unidad estándar SunCD

Tabla 4.4 Características de los servidores Ultra Enterprise 4000 (continúa).

Unidades de Almacenamiento Externo	
Discos	SPARCstorage Array Modelo 100 (Hasta 30 discos de 2.1 GB ó 4.2 GB fast/wide SCSI-2) SPARCstorage MultiPack (Hasta 54 6 GB usando discos de 2.1 GB, 4.2 GB ó 9.1 GB) SPARCstorage Array Modelo 200 serie RSM (hasta 382 GB de discos *hot-swap*[7] de 4.2 GB ó 9.1 GB con fuente de poder y ventilador redundante)
Cinta	Unidad de cinta DSS2 de 4 mm con capacidad de 4 a 8GB, unidad de cinta DDS3 de 4 mm con capacidad de 72 a 144 GB, unidad de cinta de 8 mm con capacidad de 20 a 40 GB unidad de cinta de 0.25 con capacidad de 2.5 GB, SPARCstorage Library de 140 GB, SPARCstorage Library de 400 GB, SPARCstorage DLT 4000 de 20 a 40 GB, SPARCstorage DLT 7000 de 35 a 70 GB, SPARCstorage DLT 4700
Opciones de Consola	
Monitor	Monitor a color de 17 ó 20 pulgadas
Frame Buffer	Turbo GX, Turbo GXplus, *frame buffer* Creator o Creator3D
Opciones de PCI	
Interfase serial de alta velocidad, FDDI, SunFastEthernet, fast/wide SCSI	
Fuentes de Alimentación	
Hasta cuatro fuentes de alimentación de 300 watts. Una fuente de alimentación requerida por cada dos tarjetas del sistema. Una fuente de alimentación para los periféricos de 184 watts. Completa redundancia de las fuentes y de los ventiladores.	
Software	
Sistema operativo	Solaris 2.5.1
Lenguajes de programación	C, C++, Pascal, FORTRAN, Java
Software de red	ONC, NFS, TCP/IP, SunNet OSI, MHS, X.25, DCE, NetWare
Monitoreo del sistema	Solstice SyMON
Entorno Gráfico	OpenWindows versión 3
Software de administración del sistema y de la red	Solstice SunNet Manager, Networker for Solaris, Solstice DiskSuite, SPARCstorage Volume Manager
Conectividad con IBM/DEC	Productos de conectividad SunLink
Condiciones Ambientales	
Alimentación	100-240 VAC, 47-63 Hz, 10 A
Ambiente de operación	5° C a 40° C, 20% a 80% de humedad relativa, sin condensación
Ambiente de almacenamiento	-20° C a 60° C, 5% a 93% de humedad relativa, sin condensación

Tabla 4.4 Características de los servidores Ultra Enterprise 4000.

4.2.2 Elección de la Plataforma para el Cliente

Tomando la arquitectura de la solución propuesta, puede observarse que se emplearán las terminales existentes (PC's) que dan servicio en las diversas instituciones financieras para los sistemas SPEUA y SAGA como plataforma para implantar la parte del cliente del sistema de distribución y actualización de versiones.

En este punto, no puede hablarse de un proveedor único, ya que cada institución financiera cuenta con el proveedor de equipo de cómputo de su elección, únicamente se apega a los requerimientos solicitados por el Banco de México para implantar las

[7] Se conoce como discos hot-swap, a aquellos que pueden ser remplazados sin necesidad de detener la actividad normal del sistema

terminales. A continuación se muestra la tabla 4.5 que contiene las características de equipo de cómputo requeridas para ejecutar software del Banco de México.

Tipo de computadora	PC de escritorio
Memoria	16 MB de RAM
CPU	486 a 66 MHz o superior
Floppy	3.5"
Disco Duro	500 MB como mínimo
Tarjeta de red	Compatible con 10:BaseT (Para la red financiera)
Monitor	VGA 14" o Superior
Sistema Operativo	Windows 3.11 ó Windows 95

Tabla 4.5 Requerimientos solicitados por Banco de México para las terminales en las instituciones financieras.

4.3 Costo Estimado de Implementación

Se cuenta con una licencia ilimitada de uso en cuanto usuarios para Sybase SQL Server 11, Power Builder 5.0 y Open Client de Sybase (software empleado por las herramientas de desarrollo para interactuar con bases de datos en Sybase). También se cuentan con las licencias requeridas para desarrollar en Borland C++ 5.0 y Visual C++ 4.0. Además se tiene vigente la licencia para emplear WinBatch tanto para Windows 3.11 como para Windows 95.

Como se ha mencionado en el apartado 4.2.1, se cuenta con un servidor Ultra Enterprise 4000, al cual no deben hacerse actualizaciones de hardware, ya que el sistema no demandará los recursos de forma intensiva, el incremento en el número de usuarios por el uso del sistema de distribución y actualización de versiones será mínimo, debido a que son los usuarios (ya existentes) de las terminales los que podrán ejecutar este sistema. De esta forma no se requerirá más memoria debida a conexiones de usuarios adicionales, lo mismo aplica para el uso del CPU. En cuanto al espacio en disco, tampoco se requerirá alguna actualización, debido a que el servidor Enterprise 4000 cuenta con un SPARCstorage Array modelo 100 con 18 discos de 2.1 GB.

Por lo anteriormente expuesto, sólo deben considerarse en este punto los costos asociados a los sueldos de las personas involucradas en el proyecto, el cual aproximadamente consiste de $160,000.00 MN (ver sección 2.5).

4.4 Plan de Implementación

Una vez seleccionado el software y hardware con el que trabajaremos, y de haber estimado el costo de la inversión del proyecto, es necesario planificar como se desarrollará cada una de las etapas de la implementación, de tal manera que el trabajo se lleve a cabo en una manera ordenada y planificada.

En esta etapa se detalla cada paso de la implementación, donde abarcaremos las etapas en las cuales será montado el ambiente de pruebas, así como las conexiones físicas o cableados requeridos para generar un ambiente de desarrollo, similar al de producción. Nuestro ambiente de pruebas consistirá en una pequeña red independiente en la que se conectará un servidor Unix (en el que se ejecutarán tanto el servidor de

Bases de Datos como el servidor de archivos) con algunas computadoras PC's que estarán realizando algún tipo de operación con la red para simular el tráfico habitual. Es importante mencionar que se requiere del ambiente de pruebas, porque no se puede estar probando en el ambiente real de producción. Una vez especificado lo anterior, se planearán las tareas que intervendrán en el desarrollo de los programas a implementar, y se especificarán los detalles de las pantallas que intervienen.

Posteriormente se elaborarán cada uno de los procesos, los cuales serán capturados, compilados y probados. Es importante que durante el desarrollo de los módulos sea implementada también la base de datos.

Terminado lo anterior, propiamente se cuenta con el sistema terminado, pero es preciso probarlo, corregirlo y documentarlo, para que continúe sin ningún problema.

En la tabla 4.6 se muestra el plan de implementación propuesto. La tabla 4.7 muestra a detalle la información general de cada una de las tareas mencionadas en el plan. En la tabla 4.8 definimos a cada uno de los recursos que intervienen y su costo estimado cada uno por hora (ver capitulo 2). La tabla 4.9 muestra la cantidad de horas hombre que interviene cada recurso en cada semana, así como el total de horas requeridas por cada recurso. Mostramos también en la tabla 4.10 el diagrama de la implementación. La tabla 4.11 muestra el costo de cada una de las tareas tomando en cuenta solo el salario del recurso por horas dedicada, este costo es global y semanal. En la tabla 4.12 mostramos por cada recurso las horas que intervendrá en cada tarea por semana y global. La tabla 4.13 muestra el detalle de cada recurso que interviene, las tareas en las que interviene, el costo por tarea ejecutada del recurso y el costo del recurso en la implementación. Finalmente el costo total estimando de gasto debido a la implementación es de $128,904 pesos por recursos humanos según la tabla 4.14, que describe el resumen total de la implementación.

Con la información contenida en las tablas antes mencionadas tenemos bases de tiempo, costo, tareas y recursos, que nos permiten un excelente control de la implementación del proyecto.

PLAN DE IMPLEMENTACION

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
3	Instalación del servidor de pruebas	4d	27/01/97	30/01/97		S1
4	Instalación de software red	2d	31/01/97	3/02/97	3	S1
5	Instalación PC de pruebas	2d	4/02/97	5/02/97	4	S1
6	Instalación software PC pruebas	1d	6/02/97	6/02/97	5	S1
7	Prueba de comunicaciones	1d	7/02/97	7/02/97	6	S1
9	Conexión de red física	4d	27/01/97	30/01/97		S2
10	Pruebas de comunicaciones	2d	31/01/97	3/02/97	9	S2
12	Especificación de las tablas	12d	10/02/97	25/02/97		A0[0.5]
13	Especificación de entidades y relaciones	2d	26/02/97	27/02/97	12	A0
14	Especificación de Indices	2d	28/02/97	3/03/97	13	A0
16	Especificación de programas de servidor	5d	27/01/97	31/01/97		A1
17	Especificación de programas de usuario	5d	3/02/97	7/02/97	16	A1
18	Especificación de programas de base de datos	10d	10/02/97	21/02/97	17	A0[0.5],A1
20	Pantallas de Usuario	5d	24/02/97	28/02/97	18	A1
22	Programa de identificación de terminal	10d	3/03/97	14/03/97		P0
23	Programa de verificación acceso a transferencia	7d	17/03/97	25/03/97	22	P0
24	Programa de verificación de requerimientos del sistema	7d	26/03/97	3/04/97	23	P0
25	Programa de transferencia	8d	4/04/97	15/04/97	24	P0
26	Programa de verificación de existencia del compacto	10d	17/03/97	28/03/97	22	P1
27	Programa de verificación de permisos para instalación	10d	31/03/97	11/04/97	26	P1
28	Programa de instalación	15d	14/04/97	2/05/97	27	P1
30	Especificación de programas de base de datos	20d	4/03/97	31/03/97		P2,A0
32	Pruebas generales	20d	5/05/97	30/05/97		A0[0.5],A1[0.5],P0[0.5],P1[
33	Mantenimiento de pruebas	20d	5/05/97	30/05/97		A0[0.5],A1[0.5],P0[0.5],P1[
34	Documentación	10d	2/06/97	13/06/97	33	A0,A1,P0,P1,P2

Tabla 4.7 Descripción a detalle de las tareas.

PLAN DE IMPLEMENTACION

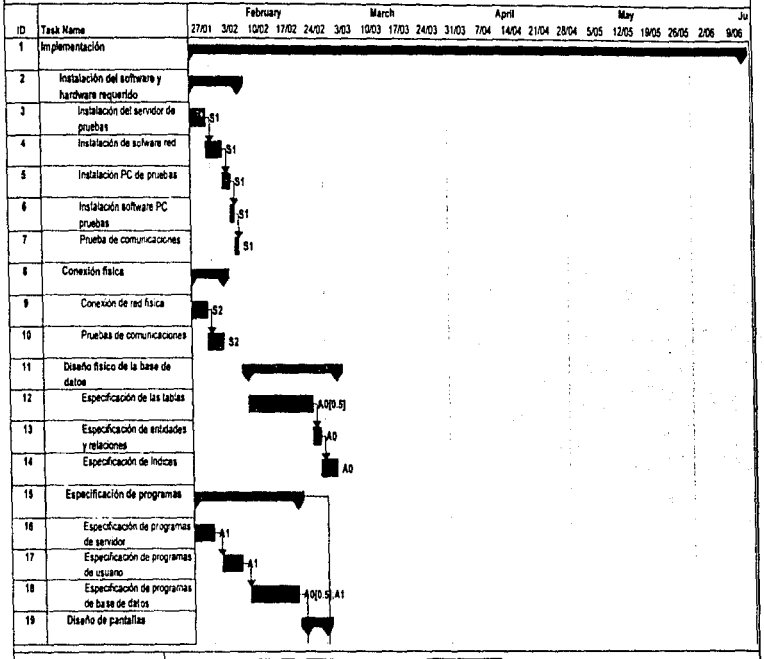


Tabla 4.6 Plan de implementación propuesto

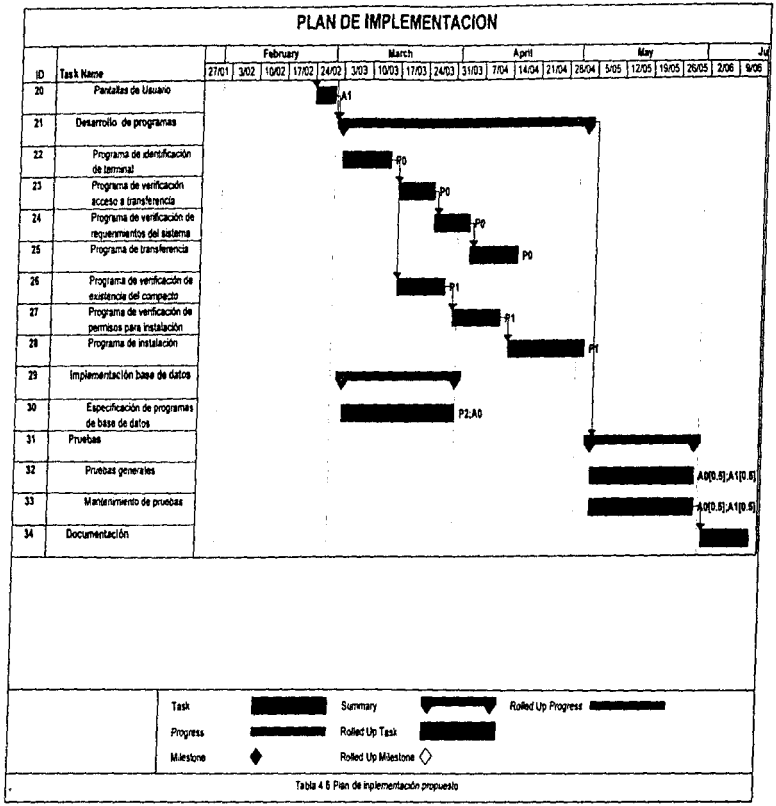


Tabla 4.6 Plan de implementación propuesto

Recurso	Inicial	Unidades	Costo Est. Por Hora	Costo Extra Por Hora	Calendario
S1	S	1	$38	0	Estandar
S2	S	1	38	0	Estandar
A0	A	1	38	0	Estandar
A1	A	1	38	0	Estandar
P0	P	1	25	0	Estandar
P1	P	1	25	0	Estandar
P2	P	1	25	0	Estandar
Lider	L	1	50	0	Estandar

Tabla 4.8 Recursos que intervienen y costo estimado por hora.

Recurso	Horas Trab	SEM1	SEM2	SEM3	SEM4	SEM5	SEM6	SEM7	SEM8	SEM9	SEM10
S1	80	40	40								
S2	48	40	8								
A0	520			40	40	32	40	40	40	40	8
A1	760	40	40	40		40	40	40	40	40	40
P0	496						40	40	40	40	40
P1	520								40	40	40
P2	400						32	40	40	40	8
LIDER	800	40	40	40	40	40	40	40	40	40	40
Recurso	Horas Trab	SEM11	SEM12	SEM13	SEM14	SEM15	SEM16	SEM17	SEM18	SEM19	SEM20
S1	80										
S2	48										
A0	520					40	40	40	40	40	40
A1	760	40	40	40	40	40	40	40	40	40	40
P0	496	40	16			40	40	40	40	40	40
P1	520	40	40	40	40	40	40	40	40	40	40
P2	400					40	40	40	40	40	40
LIDER	800	40	40	40	40	40	40	40	40	40	40

Tabla 4.9 Cantidad de horas hombre por semana y resumen total del recurso.

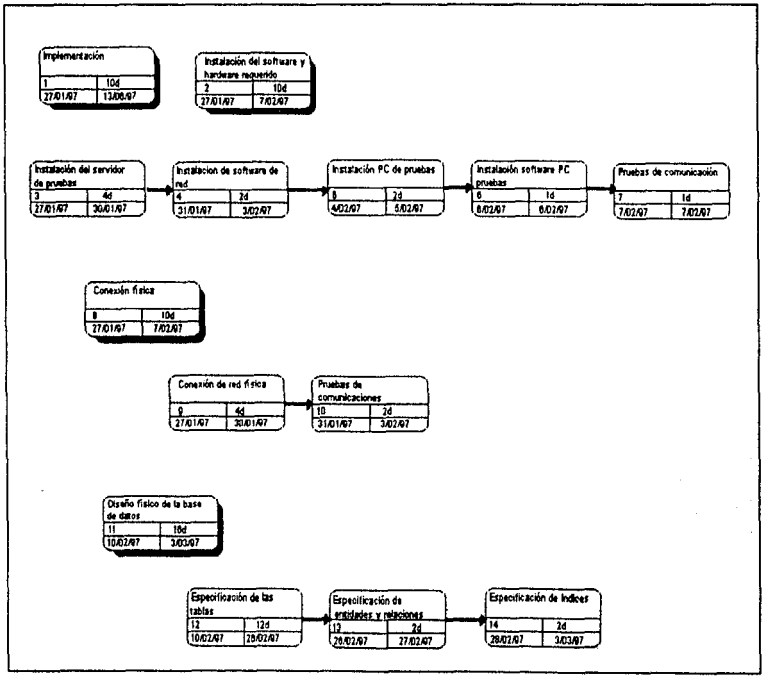


Tabla 4.10 Diagrama de la implementación

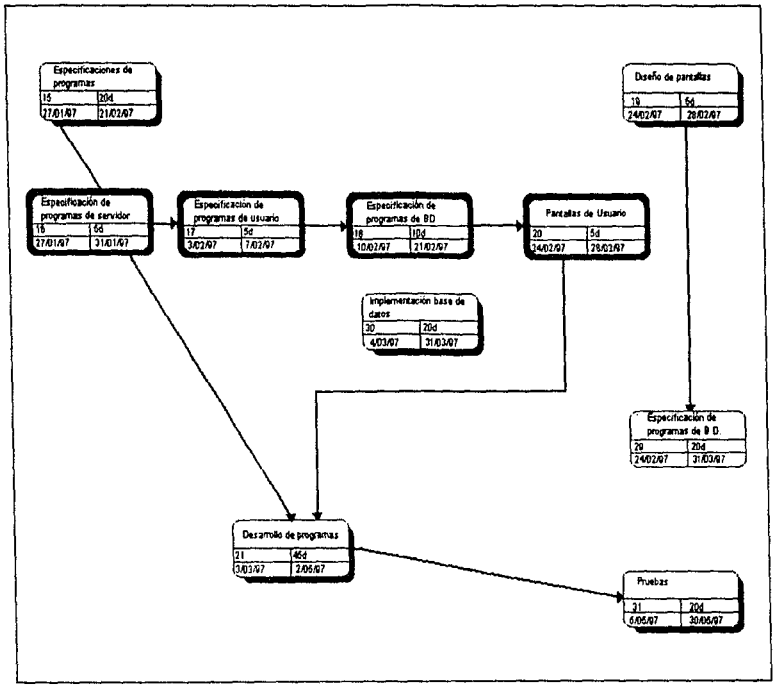


Tabla 4.10 Diagrama de la implementación (continua)

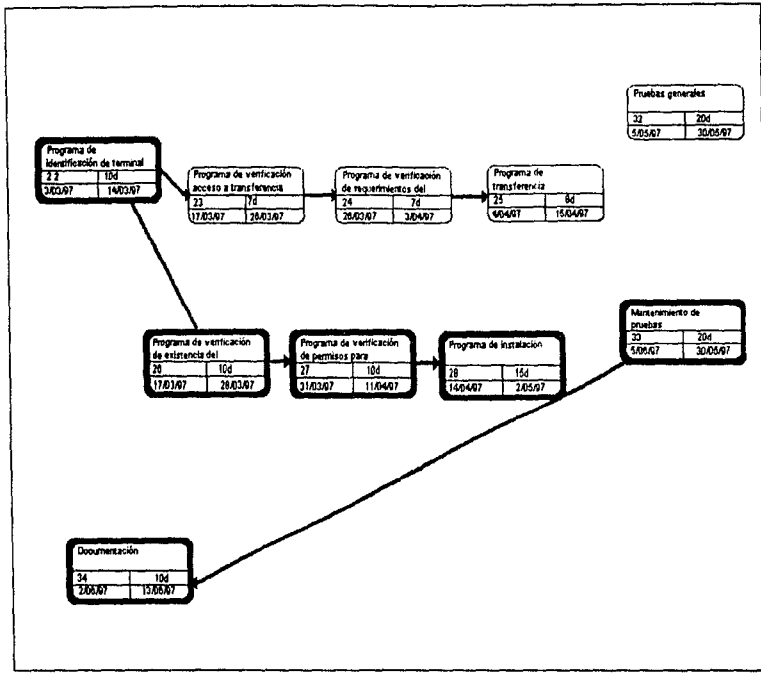


Tabla 4.10 Diagrama de la implementación (continua)

PLAN DE IMPLEMENTACION

	27/01/97	3/02/97	10/02/97	17/02/97	24/02/97	3/03/97	10/03/97	17/03/97	24/03/97
Implementación									
Instalación del software y hardware requerido									
Instalación del servidor de pruebas	$1.216								
Instalación de sofware red	$304	$304							
Instalación PC de pruebas		$608							
Instalación software PC pruebas		$304							
Prueba de comunicaciones		$304							
Conexión física									
Conexión de red física	$1.216								
Pruebas de comunicaciones	$304	$304							
Diseño físico de la base de datos									
Especificación de las tablas					$1.824				
Especificación de entidades y relaciones					$608				
Especificación de Indices						$608			
Especificación de programas									
Especificación de programas de servidor	$1.520								
Especificación de programas de usuario		$1.520							
Especificación de programas de base de datos			$1.520	$1.520					
Diseño de pantallas									
Pantallas de Usuario					$1.520				
Desarrollo de programas						$1.520	$1.520	$1.520	$1.520
Programa de identificación de terminal						$1.000	$1.000		
Programa de verificación acceso a transferencia								$1.000	$400
Programa de verificacion de requerimientos del sistema									$600
Programa de transferencia									
Programa de verificación de existencia del compacto								$1.000	$1.000
Programa de verificación de permisos para instalación									
Programa de instalación									
Implementación base de datos									
Especificación de programas de base de datos						$600	$1.000	$1.000	$1.000
Pruebas									
Pruebas generales									
Mantenimiento de pruebas									
Documentación									
Total	$4.560	$3.344	$1.520	$1.520	$3.952	$3.928	$3.520	$4.520	$4.520

Tabla 4.11 Costo por tarea, total y semanal.

PLAN DE IMPLEMENTACION

	31/03/97	7/04/97	14/04/97	21/04/97	28/04/97	5/05/97	12/05/97	19/05/97	26/05/97
Implementación									
Instalación del software y hardware requerido									
Instalación del servidor de pruebas									
Instalación de software red									
Instalación PC de pruebas									
Instalación software PC pruebas									
Prueba de comunicaciones									
Conexión física									
Conexión de red física									
Pruebas de comunicaciones									
Diseño físico de la base de datos									
Especificación de las tablas									
Especificación de entidades y relaciones									
Especificación de índices									
Especificación de programas									
Especificación de programas de servidor									
Especificación de programas de usuario									
Especificación de programas de base de datos									
Diseño de pantallas									
Pantallas de Usuario									
Desarrollo de programas	$1.520	$1.520	$1.520	$1.520	$1.520				
Programa de identificación de terminal									
Programa de verificación acceso a transferencia									
Programa de verificación de requerimientos del sistema	$800								
Programa de transferencia	$200	$1.000	$400						
Programa de verificación de existencia del compacto									
Programa de verificación de permisos para instalación	$1.000	$1.000							
Programa de instalación			$1.000	$1.000	$1.000				
Implementación base de datos									
Especificación de programas de base de datos	$6.280								
Pruebas									
Pruebas generales						$2.260	$2.260	$2.260	$5.300
Mantenimiento de pruebas						$2.260	$2.260	$2.260	$5.300
Documentación									
Total	$9.800	$3.520	$2.920	$2.520	$2.520	$4.520	$4.520	$4.520	$10.600

Tabla 4.11 Costo por tarea, total y semanal

PLAN DE IMPLEMENTACION

	2/06/97	9/06/97	Total
Implementación		$40,000	$40,000
Instalación del software y hardware requerido			
Instalación del servidor de pruebas			$1,216
Instalación de software red			$608
Instalación PC de pruebas			$608
Instalación software PC pruebas			$304
Prueba de comunicaciones			$304
Conexion fisica			
Conexión de red física			$1,216
Pruebas de comunicaciones			$608
Diseño físico de la base de datos			
Especificacion de las tablas			$1,824
Especificacion de entidades y relaciones			$608
Especificación de índices			$608
Especificación de programas			
Especificación de programas de servidor			$1,520
Especificacion de programas de usuario			$1,520
Especificacion de programas de base de datos			$3,040
Diseño de pantallas			
Pantallas de Usuario			$1,520
Desarrollo de programas			$13,680
Programa de identificación de terminal			$2,000
Programa de verificación acceso a transferencia			$1,400
Programa de verificación de requerimientos del sistema			$1,400
Programa de transferencia			$1,600
Programa de verificación de existencia del compacto			$2,000
Programa de verificación de permisos para instalación			$2,000
Programa de instalación			$3,000
Implementación base de datos			
Especificación de programas de base de datos			$10,080
Pruebas			
Pruebas generales			$12,080
Mantenimiento de pruebas			$12,080
Documentación	$4,520	$7,560	$12,080
Total	$4,520	$47,560	$128,904

Tabla 4.11 Costo por tarea, total y semanal.

PLAN DE IMPLEMENTACION

	27 Jan '97	3 Feb '97	10 Feb '97	17 Feb '97	24 Feb '97
S1	40h	40h			
Instalación del servidor de pruebas	32h				
Instalación de software red	8h	8h			
Instalación PC de pruebas		16h			
Instalación software PC pruebas		8h			
Prueba de comunicaciones		8h			
S2	40h	8h			
Conexión de red física	32h				
Pruebas de comunicaciones	8h	8h			
A1	40h	40h	40h		40h
Documentación					
Especificación de programas de servidor	40h				
Especificación de programas de usuario		40h			
Especificación de programas de base de datos			40h		
Pruebas generales					
Mantenimiento de pruebas					
Pantallas de Usuario					40h
Desarrollo de programas					
P0					
Documentación					
Programa de identificación de terminal					
Programa de verificación acceso e transferencia					
Programa de verificación de requerimientos del sistema					
Programa de transferencia					
Pruebas generales					
Mantenimiento de pruebas					
P1					
Documentación					
Programa de verificación de existencia del compacto					
Programa de verificación de permisos para instalación					
Programa de instalación					
Pruebas generales					
Mantenimiento de pruebas					
P2					
Documentación					
Especificación de programas de base de datos					
Pruebas generales					
Mantenimiento de pruebas					
A0			40h	40h	32h
Documentación					
Especificación de las tablas			20h	20h	8h
Especificación de entidades y relaciones					16h
Especificación de índices					8h
Especificación de programas de base de datos			20h	20h	
Pruebas generales					
Mantenimiento de pruebas					
Especificación de programas de base de datos					
Lider	40h	40h	40h	40h	40h
Implementación	40h	40h	40h	40h	40h
Total	160h	128h	120h	80h	112h

Tabla 4.12 Horas por recurso y tarea, semanal y global.

PLAN DE IMPLEMENTACION

	3 Mar '97	10 Mar '97	17 Mar '97	24 Mar '97	31 Mar '97
S1					
Instalación del servidor de pruebas					
Instalación de sofware red					
Instalación PC de pruebas					
Instalación software PC pruebas					
Prueba de comunicaciones					
S2					
Conexión de red física					
Pruebas de comunicaciones					
A1	40h	40h	40h	40h	40h
Documentación					
Especificación de programas de servidor					
Especificación de programas de usuario					
Especificación de programas de base de datos					
Pruebas generales					
Mantenimiento de pruebas					
Pantallas de Usuario					
Desarrollo de programas	40h	40h	40h	40h	40h
P0	40h	40h	40h	40h	40h
Documentación					
Programa de identificación de terminal	40h	40h			
Programa de verificación acceso a transferencia			40h	16h	
Programa de verificación de requerimientos del sistema				24h	32h
Programa de transferencia					8h
Pruebas generales					
Mantenimiento de pruebas					
P1			40h	40h	40h
Documentación					
Programa de verificación de existencia del compacto			40h	40h	
Programa de verificación de permisos para instalación					40h
Programa de instalación					
Pruebas generales					
Mantenimiento de pruebas					
P2	32h	40h	40h	40h	8h
Documentación					
Especificación de programas de base de datos	32h	40h	40h	40h	8h
Pruebas generales					
Mantenimiento de pruebas					
A0	40h	40h	40h	40h	8h
Documentación					
Especificación de las tablas					
Especificación de entidades y relaciones					
Especificación de Indices	8h				
Especificación de programas de base de datos					
Pruebas generales					
Mantenimiento de pruebas					
Especificación de programas de base de datos	32h	40h	40h	40h	8h
Lider	40h	40h	40h	40h	40h
Implementación	40h	40h	40h	40h	40h
Total	192h	200h	240h	240h	176h

Tabla 4.12 Horas por recurso y tarea, semanal y global.

PLAN DE IMPLEMENTACION

	7 Apr '97	14 Apr '97	21 Apr '97	28 Apr '97	5 May '97
S1					
Instalación del servidor de pruebas					
Instalación de sofware red					
Instalación PC de pruebas					
Instalación software PC pruebas					
Prueba de comunicaciones					
S2					
Conexión de red física					
Pruebas de comunicaciones					
A1	40h	40h	40h	40h	40h
Documentación					
Especificación de programas de servidor					
Especificación de programas de usuario					
Especificación de programas de base de datos					
Pruebas generales					20h
Mantenimiento de pruebas					20h
Pantallas de Usuario					
Desarrollo de programas	40h	40h	40h	40h	
P0	40h	16h			40h
Documentación					
Programa de identificación de terminal					
Programa de verificación acceso a transferencia					
Programa de verificación de requerimientos del sistema					
Programa de transferencia	40h	16h			
Pruebas generales					20h
Mantenimiento de pruebas					20h
P1	40h	40h	40h	40h	40h
Documentación					
Programa de verificación de existencia del compacto					
Programa de verificación de permisos para instalación	40h				
Programa de instalación		40h	40h	40h	
Pruebas generales					20h
Mantenimiento de pruebas					20h
P2					40h
Documentación					
Especificación de programas de base de datos					
Pruebas generales					20h
Mantenimiento de pruebas					20h
A0					40h
Documentación					
Especificación de las tablas					
Especificación de entidades y relaciones					
Especificación de Indices					
Especificación de programas de base de datos					
Pruebas generales					20h
Mantenimiento de pruebas					20h
Especificación de programas de base de datos					
Lider	40h	40h	40h	40h	40h
Implementación	40h	40h	40h	40h	40h
Total	160h	136h	120h	120h	240h

Tabla 4.12 Horas por recurso y tarea, semanal y global

PLAN DE IMPLEMENTACION

	12 May '97	19 May '97	26 May '97	2 Jun '97	9 Jun '97
S1					
Instalación del servidor de pruebas					
Instalación de software red					
Instalación PC de pruebas					
Instalación software PC pruebas					
Prueba de comunicaciones					
S2					
Conexión de red física					
Pruebas de comunicaciones					
A1	40h	40h	40h	40h	40h
Documentación				40h	40h
Especificación de programas de servidor					
Especificación de programas de usuario					
Especificación de programas de base de datos					
Pruebas generales	20h	20h	20h		
Mantenimiento de pruebas	20h	20h	20h		
Pantallas de Usuario					
Desarrollo de programas					
P0	40h	40h	40h	40h	40h
Documentación				40h	40h
Programa de identificacion de terminal					
Programa de verificacion acceso a transferencia					
Programa de verificación de requerimientos del sistema					
Programa de transferencia					
Pruebas generales	20h	20h	20h		
Mantenimiento de pruebas	20h	20h	20h		
P1	40h	40h	40h	40h	40h
Documentacion				40h	40h
Programa de verificación de existencia del compacto					
Programa de verificación de permisos para instalación					
Programa de instalacion					
Pruebas generales	20h	20h	20h		
Mantenimiento de pruebas	20h	20h	20h		
P2	40h	40h	40h	40h	40h
Documentación				40h	40h
Especificación de programas de base de datos					
Pruebas generales	20h	20h	20h		
Mantenimiento de pruebas	20h	20h	20h		
A0	40h	40h	40h	40h	40h
Documentación				40h	40h
Especificación de las tablas					
Especificación de entidades y relaciones					
Especificación de Indices					
Especificación de programas de base de datos					
Pruebas generales	20h	20h	20h		
Mantenimiento de pruebas	20h	20h	20h		
Especificación de programas de base de datos					
Lider	40h	40h	40h	40h	40h
Implementación	40h	40h	40h	40h	40h
Total	240h	240h	240h	240h	240h

Tabla 4.12 Horas por recurso y tarea, semanal y global.

PLAN DE IMPLEMENTACION

	Total
S1	80h
Instalación del servidor de pruebas	32h
Instalación de sofware red	16h
Instalación PC de pruebas	16h
Instalación software PC pruebas	8h
Prueba de comunicaciones	8h
S2	48h
Conexión de red física	32h
Pruebas de comunicaciones	16h
A1	760h
Documentación	80h
Especificación de programas de servidor	40h
Especificación de programas de usuario	40h
Especificación de programas de base de datos	40h
Pruebas generales	80h
Mantenimiento de pruebas	80h
Pantallas de Usuario	40h
Desarrollo de programas	360h
P0	496h
Documentacion	80h
Programa de identificación de terminal	80h
Programa de verificación acceso a transferencia	56h
Programa de verificacion de requerimientos del sistema	56h
Programa de transferencia	64h
Pruebas generales	80h
Mantenimiento de pruebas	80h
P1	520h
Documentacion	80h
Programa de verificación de existencia del compacto	80h
Programa de verificación de permisos para instalación	80h
Programa de instalación	120h
Pruebas generales	80h
Mantenimiento de pruebas	80h
P2	400h
Documentacion	80h
Especificación de programas de base de datos	160h
Pruebas generales	80h
Mantenimiento de pruebas	80h
A0	520h
Documentacion	80h
Especificación de las tablas	48h
Especificación de entidades y relaciones	16h
Especificación de Indices	16h
Especificación de programas de base de datos	40h
Pruebas generales	80h
Mantenimiento de pruebas	80h
Especificación de programas de base de datos	160h
Lider	800h
Implementación	800h
Total	3624h

Tabla 4.12 Horas por recurso y tarea, semanal y global.

PLAN DE IMPLEMENTACION

ID	Resource Name	Cost	Baseline Cost	Variance	Actual Cost	Remaining
1	S1	$3,040	$0	$3,040	$0	$3,040

ID	Task Name	Units	Cost	Baseline Cost	Act Cost	Rem Cost	Work	Ovt Work	Baseline Work	Act Work	Rem Work
3	Instalación del servidor de pruebas	1	$1,216	$0	$0	$1,216	32h	0h	0h	0h	32h
4	Instalación de software red	1	$608	$0	$0	$608	16h	0h	0h	0h	16h
5	Instalación PC de pruebas	1	$608	$0	$0	$608	16h	0h	0h	0h	16h
6	Instalación software PC pruebas	1	$304	$0	$0	$304	8h	0h	0h	0h	8h
7	Prueba de comunicaciones	1	$304	$0	$0	$304	8h	0h	0h	0h	8h

2	S2	$1,824	$0	$1,824	$0	$1,824

ID	Task Name	Units	Cost	Baseline Cost	Act Cost	Rem Cost	Work	Ovt Work	Baseline Work	Act Work	Rem Work
9	Conexión de red física	1	$1,216	$0	$0	$1,216	32h	0h	0h	0h	32h
10	Pruebas de comunicaciones	1	$608	$0	$0	$608	16h	0h	0h	0h	16h

3	A1	$28,880	$0	$28,880	$0	$28,880

ID	Task Name	Units	Cost	Baseline Cost	Act Cost	Rem Cost
16	Especificación de programas de servidor	1	$1,520	$0	$0	$1,520
17	Especificación de programas de usuario	1	$1,520	$0	$0	$1,520
18	Especificación de programas de base de datos	1	$1,520	$0	$0	$1,520
20	Pantallas de Usuario	1	$1,520	$0	$0	$1,520
21	Desarrollo de programas	1	$13,680	$0	$0	$13,680
32	Pruebas generales	0.5	$3,040	$0	$0	$3,040
33	Mantenimiento de pruebas	0.5	$3,040	$0	$0	$3,040
34	Documentación	1	$3,040	$0	$0	$3,040

ID	Task Name	Units	Work	Ovt Work	Baseline Work	Act Work	Rem Work
16	Especificación de programas de servidor	1	40h	0h	0h	0h	40h
17	Especificación de programas de usuario	1	40h	0h	0h	0h	40h
18	Especificación de programas de base de datos	1	40h	0h	0h	0h	40h
20	Pantallas de Usuario	1	40h	0h	0h	0h	40h
21	Desarrollo de programas	1	360h	0h	0h	0h	360h
32	Pruebas generales	0.5	80h	0h	0h	0h	80h
33	Mantenimiento de pruebas	0.5	80h	0h	0h	0h	80h
34	Documentación	1	80h	0h	80h	0h	80h

4	P0	$12,400	$0	$12,400	$0	$12,400

ID	Task Name	Units	Cost	Baseline Cost	Act Cost	Rem Cost
22	Programa de identificación de terminal	1	$2,000	$0	$0	$2,000
23	Programa de verificación acceso a transferencia	1	$1,400	$0	$0	$1,400
24	Programa de verificación de requerimientos del sistema	1	$1,400	$0	$0	$1,400
25	Programa de transferencia	1	$1,600	$0	$0	$1,600
32	Pruebas generales	0.5	$2,000	$0	$0	$2,000
33	Mantenimiento de pruebas	0.5	$2,000	$0	$0	$2,000
34	Documentación	1	$2,000	$0	$0	$2,000

ID	Task Name	Units	Work	Ovt Work	Baseline Work	Act Work	Rem Work
22	Programa de identificación de terminal	1	80h	0h	0h	0h	80h
23	Programa de verificación acceso a transferencia	1	56h	0h	0h	0h	56h
24	Programa de verificación de requerimientos del sistema	1	56h	0h	0h	0h	56h
25	Programa de transferencia	1	64h	0h	0h	0h	64h
32	Pruebas generales	0.5	80h	0h	0h	0h	80h
33	Mantenimiento de pruebas	0.5	80h	0h	0h	0h	80h
34	Documentación	1	80h	0h	80h	0h	80h

Tabla 4.13 Tareas de cada recurso, costo por tarea y recurso

PLAN DE IMPLEMENTACION

ID	Resource Name	Cost	Baseline Cost	Variance	Actual Cost	Remaining
5	P1	$13.000	$0	$13.000	$0	$13,000

ID	Task Name	Units	Cost	Baseline Cost	Act. Cost	Rem. Cost
26	Programa de verificación de existencia del compacto	1	$2.000	$0	$0	$2.000
27	Programa de verificación de permisos para instalación	1	$2.000	$0	$0	$2.000
28	Programa de instalación	1	$3.000	$0	$0	$3.000
32	Pruebas generales	0.5	$2.000	$0	$0	$2.000
33	Mantenimiento de pruebas	0.5	$2.000	$0	$0	$2.000
34	Documentación	1	$2.000	$0	$0	$2.000

ID	Task Name	Units	Work	Ovt. Work	Baseline Work	Act. Work	Rem. Work
26	Programa de verificación de existencia del compacto	1	80h	0h	0h	0h	80h
27	Programa de verificación de permisos para instalación	1	80h	0h	0h	0h	80h
28	Programa de instalación	1	120h	0h	0h	0h	120h
32	Pruebas generales	0.5	80h	0h	0h	0h	80h
33	Mantenimiento de pruebas	0.5	80h	0h	0h	0h	80h
34	Documentación	1	80h	0h	80h	0h	80h

6	P2	$10.000	$0	$10.000	$0	$10,000

ID	Task Name	Units	Cost	Baseline Cost	Act. Cost	Rem. Cost
30	Especificación de programas de base de datos	1	$4.000	$0	$0	$4.000
32	Pruebas generales	0.5	$2.000	$0	$0	$2.000
33	Mantenimiento de pruebas	0.5	$2.000	$0	$0	$2.000
34	Documentación	1	$2.000	$0	$0	$2.000

ID	Task Name	Units	Work	Ovt. Work	Baseline Work	Act. Work	Rem. Work
30	Especificación de programas de base de datos	1	160h	0h	0h	0h	160h
32	Pruebas generales	0.5	80h	0h	0h	0h	80h
33	Mantenimiento de pruebas	0.5	80h	0h	0h	0h	80h
34	Documentación	1	80h	0h	80h	0h	80h

7	A0	$19.760	$0	$19.760	$0	$19,760

ID	Task Name	Units	Cost	Baseline Cost	Act. Cost	Rem. Cost
12	Especificación de las tablas	0.5	$1.824	$0	$0	$1.824
13	Especificación de entidades y relaciones	1	$608	$0	$0	$608
14	Especificación de índices	1	$608	$0	$0	$608
18	Especificación de programas de base de datos	0.5	$1.520	$0	$0	$1.520
30	Especificación de programas de base de datos	1	$6.080	$0	$0	$6.080
32	Pruebas generales	0.5	$3.040	$0	$0	$3.040
33	Mantenimiento de pruebas	0.5	$3.040	$0	$0	$3.040
34	Documentación	1	$3.040	$0	$0	$3.040

ID	Task Name	Units	Work	Ovt. Work	Baseline Work	Act. Work	Rem. Work
12	Especificación de las tablas	0.5	48h	0h	0h	0h	48h
13	Especificación de entidades y relaciones	1	16h	0h	0h	0h	16h
14	Especificación de índices	1	16h	0h	0h	0h	16h
18	Especificación de programas de base de datos	0.5	40h	0h	0h	0h	40h
30	Especificación de programas de base de datos	1	160h	0h	0h	0h	160h
32	Pruebas generales	0.5	80h	0h	0h	0h	80h
33	Mantenimiento de pruebas	0.5	80h	0h	0h	0h	80h
34	Documentación	1	80h	0h	80h	0h	80h

8	Lider	$40.000	$0	$40.000	$0	$40,000

ID	Task Name	Units	Cost	Baseline Cost	Act. Cost	Rem. Cost	Work	Ovt. Work	Baseline Work	Act. Work	Rem. Work
1	Implementación	1	$40.000	$0	$0	$40.000	800h	0h	0h	0h	800h

Tabla 4.13 Tareas de cada recurso, costo por tarea y recurso

PLAN DE IMPLEMENTACION

Fechas			
Inicio	27/01/97	Termino	13/06/97
Duración			
Calendarizado	100d	Falta	100d
Trabajo			
Calendarizado	3624h	Falta	3624h
Costo			
Calendarizado	$128,904.00	Falta	$128,904.00
Tareas			
Tareas por empezar	34	Falta	34

Tabla 4.14 Resumen general del proyecto.

4.5 Especificación de Programas

A continuación se explicará el funcionamiento de los procesos que conforman el sistema de distribución y actualización de software (Versiones).

4.5.1 Identificación de Terminal

Por aspectos de seguridad es necesario identificar la terminal que se conecta al sistema de Versiones. Esto se realiza por medio de un programa que detecta la dirección IP de la terminal remota.

Para realizar este proceso se creó una librería de funciones en lenguaje C, la cuál utiliza *sockets* para poder establecer una comunicación entre la terminal remota y el servidor y de esta forma detectar las características de la máquina.

Esta librería (winftp.dll) contiene la función wsDir_IP que recibe una variable por referencia `sIPAddr` donde se almacenará la dirección IP de la terminal.

Ejemplo del código de la librería:

```
#pragma argsused
extern "C" int FAR PASCAL _export wsDir_IP(LPSTR sIPAddr)
{
      u_long lul_dir;
      int  li_Tmp;
      int  li_Sock;

      WSADATA wsaData;            // Detalles de implementación de Winsock

      if (WSAStartup(WINSOCK_VERSION, &wsaData))
      {
        MessageBeep(MB_ICONSTOP);
        MessageBox(NULL, "No se pudo cargar la librería de sockets
                                        para Windows.",
        lpszFunctionName, MB_OK|MB_ICONSTOP);
        return(-1);
      }

      lul_dir = wsGetHostID();
      if( lul_dir > 2 )
      {
        li_Tmp =  wsinet_ntoa(sIPAddr, lul_dir);
      }
      else
        li_Tmp = ( (int)lul_dir ) * (-1);
      li_Sock = WSACleanup();
      if (li_Sock == SOCKET_ERROR)
      {
        int iWinsockErr = WSAGetLastError();
        wsprintf(gszCommandBuffer,
            "¡¡ Error #%d al intentar cerrar Winsock !!",
            iWinsockErr);
        MessageBeep(MB_ICONSTOP);
        MessageBox(NULL, gszCommandBuffer, lpszFunctionName,
            MB_OK|MB_ICONSTOP);
        return(li_Sock);
      }
```

```
    return(li_Tmp);
}
```

Desde Power Builder se hace un llamado a la función wsDir_IP de la libreria winftp.dll para obtener en una variable local el valor de la dirección IP.

```
String      ls_dirIPtemp = ""
Int         li_Pos

li_Pos = wsDir_IP(ls_dirIPtemp)
```

Para poder invocar la función de la librería hecha en lenguaje C desde Power Builder, se tiene que declarar como función externa de la siguiente forma:

```
FUNCTION INT wsDir_IP(REF STRING sIPAddr) LIBRARY "winftp.dll"
```

Lo cual quiere decir que se está declarando la función wsDir_IP perteneciente a la LIBRERIA "winftp.dll" que retorna un valor INT y que recibe el parámetro STRING SIPAddr como variable de referencia.

De esta forma se logra identificar la terminal por medio de la Dirección IP que es única en toda la red.

Esta dirección queda almacenada en una variable para después hacer uso de ella.

4.5.2 Verificación de Acceso de Transferencia

Una vez identificada la terminal, el usuario introduce el sistema y la versión de sistema que le interesa actualizar. De esta forma tenemos ya tres parámetros importantes (Dirección IP, Sistema, Versión) que decidirán si la terminal tiene permisos o no para obtener el sistema.

El proceso de verificación de acceso de transferencia se realiza por medio de un *stored procedure*. Para esto necesitamos realizar la conexión desde Power Builder con el servidor que contiene la Base de Datos Sybase, la cual almacena toda la información para poder llevar un control de seguridad y administrar las versiones de los distintos sistemas disponibles. En caso de no poder establecerse la conexión por algún motivo, Versiones intenta conectarse hacia otros servidores donde se tienen replicados los datos para casos de emergencia.

El *stored procedure* verifica en distintas tablas de la Base de Datos si la terminal tiene permiso par transferir la versión del sistema especificado.

El *stored procedure* recibe como datos de entrada la Dirección IP, el Sistema y la Versión del sistema.

Si el acceso de transferencia es válido, el *stored procedure* regresa información a la terminal externa que servirá para verificar los requerimientos del sistema o aplicación solicitada, y también información para la conexión al servidor de aplicaciones:

- Clave del usuario para conexión al servidor de aplicaciones UNIX.
- Contraseña del usuario para conexión al servidor de aplicaciones UNIX.
- Dirección IP del servidor de aplicaciones UNIX.
- Tamaño del archivo compacto del sistema solicitado.
- Ruta del archivo compacto en el servidor UNIX.
- Ruta del archivo compacto en el disco duro.
- Memoria principal requerida por el sistema a transmitir.
- Memoria secundaria requerida por el sistema a transmitir.
- Espacio en disco duro necesario para el sistema.
- Versión del sistema operativo necesaria para la ejecución del sistema.

En caso de que el acceso de transferencia no sea válido, el *stored procedure* retorna un número y mensaje de error.

4.5.3 Verificación de Requerimientos del Sistema

Antes de realizar la transferencia, el sistema de Versiones verifica que la terminal cumpla con los requisitos establecidos para la transferencia e instalación del sistema solicitado, tales como espacio en disco, versión de sistema operativo, memoria principal y secundaria.

Para esto, Versiones utiliza funciones externas de la librería BMXVER.DLL realizada en lenguaje C.

Esta librería contiene funciones que permiten verificar todas las características de la terminal, así como el manejo de archivos.

Las funciones requeridas para este proceso son declaradas en Power Builder como funciones externas de la siguiente forma:

```
Function Long ESPACIO_DISCO(Int Drive) Library "BMXVER.DLL"

Function Long ESPACIO_MEMORIA( ) Library "BMXVER.DLL"

Function Int VER_DOS( ) Library "BMXVER.DLL"
```

4.5.4 Programa de Transferencia

Este programa ejecuta la transferencia de la aplicación desde el servidor UNIX hasta la terminal remota a través de FTP.

Para realizar la transferencia se vuelve a hacer referencia a las funciones de la librería winftp.dll.

La librería WinFTP.DLL realiza dos procesos : Obtención de la Dirección IP de la máquina local, y la ejecución del FTP del Cliente al Servidor (Versiones).

Para obtener la Dirección IP, se necesita llamar solamente a la función wsDir_IP. Para realizar el FTP, se necesita llamar a las siguientes funciones en orden :

```
Inicia_FTP
   Ejecuta_FTP
      AbreArch_FTP
         Recibe_FTP
      CierraArch_FTP
Termina_FTP
```

Para poder invocar las funciones de la librería hecha en lenguaje C desde Power Builder, se tienen que declarar como funciones externas de la siguiente forma:

```
FUNCTION INT Inicia_FTP(REF STRING HOST_NAME, REF STRING,
USUARIO, REF STRING CLAVE) LIBRARY "winftp.dll"

FUNCTION INT Ejecuta_FTP(UINT hConnectChannel, REF STRING FUENTE)
LIBRARY "winftp.dll"

FUNCTION INT AbreArch_FTP(REF STRING FILENAME) LIBRARY "winftp.dll"

FUNCTION INT Recibe_FTP(UINT hDataSocket, INTEGER hFile) LIBRARY
"winftp.dll"

FUNCTION INT CierraArch_FTP(UINT hDataSocket, INTEGER hFile) LIBRARY
"winftp.dll"

FUNCTION INT Termina_FTP(UINT hConnectChannel, INTEGER Codigo)
LIBRARY "winftp.dll"
```

La función Inicia_FTP se utiliza para realizar la conexión con el servidor de aplicaciones UNIX.

La función Ejecuta_FTP realiza la petición del archivo compacto del sistema que queremos transmitir.

La función AbreArch_FTP crea el archivo local a recibir.

La función Recibe_FTP se encarga de realizar la recepción de datos.

La función CierraArch_FTP cierra el archivo recibido.

La función Termina_FTP se encarga de terminar la conexión con el servidor.

Para ejecutar estas funciones se necesitan los parámetros: Dirección IP del servidor de aplicaciones, Clave y Contraseña para conectarse al servidor de aplicaciones, Ruta del destino del archivo compacto a transferir al disco duro, Ruta origen del archivo compacto en el servidor UNIX.

Estos parámetros se obtuvieron anteriormente por medio del programa que verifica el acceso de transferencia.

Una vez realizada la transferencia, el archivo compacto de la aplicación queda en el subdirectorio C:\BMX\SISTEMA\VERSION\COMPACTO.EXE

Donde SISTEMA es el nombre del sistema a instalar y VERSION es el tipo de versión del sistema transferido.

Ejemplo:

C:\BMX\SPEUA\ULTIMA\COMPACTO.EXE
C:\BMX\SAGA\BETA\COMPACTO.EXE

4.5.5 Verificación de Existencia del Compacto

Este programa verifica que el sistema solicitado se haya transferido exitosamente y se encuentre en el subdirectorio de trabajo.

Este procedimiento se realiza utilizando también las funciones externas de la librería BMXVER.DLL realizada en lenguaje C.

Las funciones requeridas para este proceso son declaradas en Power Builder como funciones externas de la siguiente forma:

```
Function Int EXISTE_DIR(String Dir) Library "BMXVER.DLL"

Function Int CREA_DIR(String Dir) Library "BMXVER.DLL"

Function Int BORRA_ARCHIVOS(String Dir, String Archivo) Library
"BMXVER.DLL"
```

Si el archivo compacto existe y fue transferido correctamente se procede a la instalación.

En caso de que no se haya transferido se despliega un número y mensaje de error.

4.5.6 Verificación de Permisos para Instalación

El proceso de verificación de permisos para instalación se realiza por medio de un *stored procedure*.

El *stored procedure* verifica en distintas tablas de la Base de Datos si el sistema especificado puede ser instalado por la terminal.

El *stored procedure* recibe como datos de entrada la Dirección IP, el Sistema y la Versión del sistema.

Si los permisos para instalación son válidos, el *stored procedure* no retorna valor.

Si la terminal no tiene permisos de instalación retorna un número y mensaje de error.

Este proceso se realiza cuando el sistema fue transmitido a la terminal con anterioridad y al ejecutar el sistema de Versiones únicamente se solicita la instalación.

Este proceso permite asegurar que la versión del sistema a instalar fue obtenida por la terminal y no por algún otro medio.

4.5.7 Programa de Instalación

Este programa descompacta la aplicación transferida en el subdirectorio de trabajo para posteriormente realizar su instalación y configurar el entorno en que se ejecutará.

El proceso de descompactar la aplicación se realiza con las funciones externas de la libreria BMXVER.DLL, las cuáles ya se mencionaron anteriormente.

Una vez descompactada la aplicación en el subdirectorio de trabajo, se ejecuta la instalación del sistema (archivo instalar.exe) y la configuración del entorno. Esto se hace por medio de un programa de procesamiento de scripts llamado Winbatch, el cuál ejecutará todo lo necesario para colocar cada archivo en el lugar especificado para la ejecución del sistema, así como la configuración de archivos.

A continuación se describe el contenido del archivo compacto.exe transferido desde el servidor de aplicaciones UNIX, así como el proceso realizado por el programa que realiza la instalación (instalar.exe).

ARCHIVO COMPACTO.EXE

El archivo **compacto.exe** debe estar formado por los siguientes elementos:

1. Archivos necesarios para ejecutar la aplicación.
2. El programa para llevar acabo la instalación de la aplicación (instalar.exe), que se realiza en winbatch.
3. Archivo install.ini en el cuál se basará el programa de instalación para obtener los parámetros del sistema o aplicación
4. Programa pg.exe en el cuál se basará el programa de instalación para la creación de grupos e iconos.
5. Bitmap xxfondo.bmp, el cuál es invocado por el programa de instalación para colocarse como tapiz durante la instalación del sistema.
6. En caso de que la aplicación utilice el archivo Hosts (esto es sólo para Windows 3.x, ya que en Windows 95 se hace el acceso a través de un DNS), se deberán incluir en este todos los servidores UNIX del Banco de México.
7. En caso de que la aplicación utilice el archivo Sql.ini, se deberán incluir en este todos los servidores Sybase del Banco de México.

Estructura del archivo INSTALL.INI

```
[Directorios]
numdirs=
dir1=c:\BMX\...\PRUA                    ejemplo: dir1=c:\BMX\PRUA\
dir2=.....
dir?=.....
```

```
[borrar]
bandera=0
icono=

[Requerimientos]
Megas=6

[Windows]
QueInstalo=Instalación de SISTEMA
NombreGrupo=BANCO DE MEXICO
numicono=1
FondoBmp=xxfondo.bmp

[dir1]
numdirf1=4
dir1file1=......
dir1file2=......
dir1file3=......
dir1file4=......

[dir2]
numdirf2=2
dir2file1=......
dir2file2=......

[dir3]
numdirf3=1
dir3file1=.....

[odbc]
numsec=1

[seccion1]
nombresec=Speua
Driver=C:\windows\system\wod40w.dll
UID=dba
PWD=sql
Database=C:\BMX\SPEUA\speua.db
Start=dbeng40w
DatabaseFile=C:\BMX\SPEUA\speua.db
DatabaseName=Speua
AutoStop=yes
```

Procesos que realiza el archivo instalar.exe

1. Elimina la versión anterior del sistema (dir1).
2. Verifica espacio en disco.
3. Copia archivos indicados a directorios indicados en el archivo install.ini.
4. Respalda ODBC.INI, archivo a modificar.
5. Modifica ODBC.INI.
6. Cambia formatos de fecha (dd/mm/aa) y numéricos (miles=coma, decimales=punto).
7. Crea grupo e iconos del sistema.
8. Durante la instalación pone el tapiz de BM Versiones (xxfondo.bmp).
9. Si existe algún error durante la ejecución lo deja en el archivo ERROR.BMX.

Todos los archivos instalar.exe son casi iguales, sólo cambian en el inicio cuando se especifica el directorio de trabajo de cada sistema, que es donde se encuentra el archivo install.ini.

Ejemplo de la única diferencia de los archivos instalar.exe:

```
; --- Identificación del archivo de datos iniciales
cr=Num2Char(13)                    ;tecla enter
lf=Num2Char(10)
dirfte="C:\BMX\SPEUA\DESCOMP\"
ArchIni=strcat(dirfte,"install.ini")
```

Si se desean hacer modificaciones a otros archivos de configuración como el autoexec.bat, se debe modificar el instalar.exe de acuerdo a los cambios deseados.

4.6 Especificaciones del Servidor de Archivos

En esta sección se mostrará como se encuentran distribuidos los archivos en el servidor. Antes de explicarlo es necesario revisar algunos conceptos referentes al manejo de *file systems* y permisos sobre los archivos y directorios.

El *file system* de Unix

En Unix, el *file system*, es la estructura nativa de organización y almacenamiento de la información de archivos y directorios en el disco. Controla al acceso de los usuarios, por tanto, se trata de la herramienta básica para implementar la seguridad del sistema operativo.

Como en muchos sistemas operativos, Unix almacena la información en una estructura de tipo árbol formada por archivos y directorios. Los directorios de Unix pueden contener archivos, nombres lógicos de dispositivos, ligas simbólicas y directorios, pero desde una perspectiva muy simple, todo aquello almacenado en un *file system* es un archivo.

Para cada archivo Unix almacena la siguiente información general:

- La localización del archivo dentro del disco.
- El tipo de archivo.
- El tamaño del archivo en bytes.
- La fecha de la última modificación del *inodo*[1] del archivo (*ctime*).
- La fecha de la última modificación del contenido del archivo (*mtime*).
- La fecha del último acceso al archivo (*atime*).
- El contador de referencias, es decir el número de nombres que tiene el archivo.

Unix también almacena la siguiente información relacionada con la seguridad:

- El dueño del archivo (UID).
- El grupo del archivo (GID).

[1] Se conoce como *inodo* a una estructura del sistema operativo que sirve para direccionar (apuntar) un espacio físico en el disco

- Los bits de modo del archivo, también llamados permisos de archivo.

A continuación se muestra la figura 4.1 que ilustra la organización de *file systems* dentro del sistema operativo Solaris.

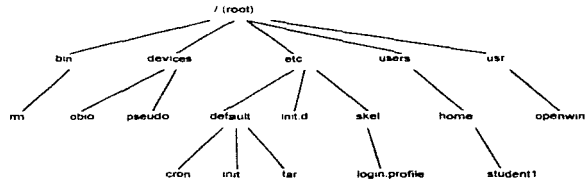


Fig 4.1 Estructura de *file systems* en el sistema operativo Solaris.

Permisos sobre archivos y directorios de Unix

Existen tres permisos que pueden ser asignados a los usuarios que empleen un archivo, los cuales son de lectura, escritura y ejecución, estos se asignan directamente al dueño del archivo, a los usuarios que pertenezcan al grupo definido en el archivo y a todos los demás usuarios.

Los términos lectura, escritura y ejecución tienen significados muy específicos para los archivos como se muestra a continuación:

- El permiso de lectura significa que el archivo puede abrirse con la llamada del sistema *open* y que puede ser leído con la instrucción *read*.

- El permiso de escritura significa que se puede sobreescribir el archivo con uno nuevo o modificar su contenido. También se permite el empleo de la instrucción *write* para agregar información al archivo o *truncate* o *ftruncate* para eliminar información.

- El permiso de ejecución tiene sentido sólo para los programas. Si un programa tiene los permisos de ejecución habilitados, entonces el programa puede ejecutarse únicamente escribiendo el nombre del archivo o ejecutándolo con alguna de las llamadas del sistema *(exec)*. La forma de ejecución del programa depende de los dos primeros bytes del archivo. Si el archivo posee el *número mágico adecuado* de dos bytes, significa que el archivo se encuentra en código de máquina y que puede cargarse en memoria y ejecutarse. Si el archivo no posee el *número mágico*, la instrucción *exec* asume que se trata de un script y emplea un shell para interpretar el contenido del archivo.

Los permisos de archivo aplican a dispositivos, *sockets* y *FIFO's* al igual que a los archivos convencionales: si se tiene permiso de escritura, entonces se puede

escribir información al archivo u objeto; si se tiene permiso de lectura, entonces puede leerse la información del archivo.

Sin embargo los permisos no aplican a las ligas simbólicas. Cuando puede leerse un archivo apuntado por una liga simbólica depende de los permisos que tenga el archivo y no de la liga. De hecho, las ligas simbólicas son siempre creadas con todos los permisos a todos los usuarios.

Existen algunas consideraciones acerca de los permisos de archivos:

- Puede tenerse permisos de ejecución sin tener permiso de lectura, en tal caso, puede ejecutarse el programa sin poderse leer el contenido. Esto es útil cuando se desea ocultar la funcionalidad del programa a los usuarios. Otra razón para tener esto es permitir a la gente ejecutar un programa sin otorgarle la posibilidad de realizar una copia de éste.

- Si se tiene permiso de lectura y no de ejecución, puede hacerse una copia y ejecutarse independientemente. La copia, sin embargo, será diferente en dos aspectos: tendrá una ruta absoluta diferente y tendrá un dueño diferente al propietario original del programa.

A diferencia de muchos sistemas operativos, Unix emplea archivos ordinarios para almacenar el contenido de los directorios. Como con los archivos ordinarios, los directorios tienen un conjunto de atributos de seguridad como son el dueño, grupo y bits de permisos. Pero debido a que los directorios son interpretados de una forma especial por el *file system*, los bits de permisos tienen un significado especial:

- Si existe el permiso de lectura, pueden emplearse las funciones *opendir* y *readdir* o el comando *ls* para encontrar que archivos se encuentran contenidos en el directorio.

- Si existe el permiso de escritura, entonces pueden agregarse o borrase archivos o ligas dentro del directorio.

- El permiso de ejecución habilita la posibilidad de inspeccionar el contenido del directorio, también permite que el usuario pueda moverse hacia ese directorio con el comando *cd*.

Si se desea prevenir que los demás usuarios lean los contenidos de los archivos pueden tomarse las siguientes acciones:

- Establecer permisos 0600 a cada archivo, de tal forma que sólo el dueño pueda leer y escribir sobre los archivos.

- Colocar los archivos en un directorio y establecer permisos 0700 sobre el directorio, de esta forma ningún usuario diferente al dueño del directorio podrá accesar los archivos.

Debe tomarse en cuenta lo siguiente:

- Se requiere permiso de ejecución para poder cambiarse al directorio (mediante el comando *cd* o *chdir*) o hacia cualquier subdirectorio contenido dentro del directorio.

- Si no se tiene permiso de ejecución en el directorio, no se podrán accesar los archivos que se encuentran dentro del directorio, aún si se es dueño de los archivos.

- Si se tienen permisos de ejecución hacia un directorio, pero no de lectura, no se pueden listar los archivos que se encuentran dentro del directorio. Sin embargo, si se tiene acceso a archivos individuales, pueden ejecutarse programas en el directorio o abrirse archivos dentro de él.

- Para borrarse un archivo de un directorio, deben tenerse permisos de escritura y ejecución sobre el directorio.

- Si se tiene permiso de lectura sobre un directorio, pero no se tiene permiso de ejecución, se podrá mostrar una lista de nombres de archivos que se encuentran en el directorio, pero no se podrá obtener ninguna información adicional.

Algunas veces es necesario que usuarios sin privilegios realicen tareas que requieran ciertos permisos. Un ejemplo es el programa *passwd*, el cual permite que se ejecute un cambio de contraseña para la cuenta que lo ejecutó. Cambiar una contraseña de usuario requiere modificar el campo de la contraseña en el archivo */etc/passwd* o */etc/shadow*. Sin embargo, es indeseable dar al usuario la posibilidad de cambiar este archivo directamente, ya que podría modificar todas las contraseñas de los usuarios. Otro ejemplo es el programa *mail* que requiere de la capacidad de colocar un archivo en el buzón de otro usuario, de otra forma sería totalmente indeseable el permitir a los usuarios acceso ilimitado a los buzones de los demás.

Para solucionar estos problemas, Unix permite a los programas una asignación especial de permisos. Tales programas asumen otro UID o GID cuando se están ejecutando. Un programa que cambia su número de usuario es llamado un programa SUID (*set-UID*); mientras que un programa que cambia su número de grupo es llamado un programa SGID (*set-GID*). Un programa puede ser SUID y SGID al mismo tiempo.

De forma convencional, cuando se ejecuta un programa, el programa se ejecuta con los privilegios del usuario que lo está ejecutando. Cuando se ejecuta un programa SUID, el UID efectivo se convierte al del usuario que creo el programa (el dueño del archivo, en vez del usuario que lo está ejecutando).

A continuación se lista el significado de los permisos especiales para los archivos:

- Un proceso que ejecuta un programa SUID, toma el UID del dueño del programa y no de la persona que está ejecutando el proceso.

- Un proceso que ejecuta un programa SGID toma el GID del programa. Los archivos que pudieran ser creados por el proceso tendrán el GID del programa también, aunque dependerá de los permisos del directorio en el que se creen los archivos.

- Un programa que tiene el *sticky* bit prendido, no será retirado del área de *swap* después de que la ejecución del programa ha concluido. Esto es útil para programas que son ejecutados frecuentemente y por múltiples usuarios, aunque esta funcionalidad se ha vuelto obsoleta con los administradores de memoria de las versiones de Unix actuales.

Aunque los permisos de SGID y sticky bits fueron originalmente destinados a programas, el Unix de Berkeley y SunOS también los usan para cambiar el comportamiento de los directorios. A continuación se explica su uso sobre directorios.

- En SunOS 4.0, el permiso SGID sobre un directorio controla la forma en la que los grupos serán asignados para los archivos creados en el directorio. Si el SGID está habilitado, los archivos creados en el directorio tendrán el mismo grupo que el del directorio. Si el SGID no está habilitado, los archivos creados dentro del directorio tendrán el mismo grupo que el del usuario que lo está creando.
- Si el *sticky bit* está habilitado, los archivos dentro del directorio pueden ser renombrados o borrados sólo por el dueño del archivo, el dueño del directorio, o el administrador (aún si los permisos del directorio indican otra cosa). Esta característica es agregada a los usuarios para el manejo de archivos en el directorio temporal de Unix /tmp.

Empleo de usuarios y grupos

En el sistema operativo Unix cada usuario tiene una cuenta (*login*) y una contraseña (*password*), sin embargo, internamente, Unix representa a cada usuario con un identificador único llamado UID. Los UID's son números de 16 bits con signo, lo que significa que pueden tener valores entre -32768 y 32767[2]. Los UID's entre 0 y 9 son destinados a los usuarios que desempeñan tareas administrativas del sistema, los UID's para los demás usuarios comienzan entre 20 y 100.

Unix mantiene la relación entre nombres de usuarios o cuentas y UID's en el archivo /etc/passwd. El UID es el número que el sistema operativo emplea para identificar a los usuarios, los nombres de usuario son proporcionados por facilidad de uso para las personas. Si a dos usuarios se les asigna el mismo UID, Unix los ve como si se tratara del mismo usuario, aún si ellos tienen diferentes cuentas y contraseñas. Como puede apreciarse, otorgar a varios usuarios el mismo UID es mala idea ya que se pierde el control sobre las acciones y permisos que tengan asignados.

Cada usuario de Unix pertenece a uno o más grupos, como en el caso de las cuentas y UID's, los grupos tienen nombre e identificador de grupo GID. El

[2] Existen algunas versiones de Unix que emplean UID's de 16 bits sin signo, lo que permite que el valor varíe entre 0 y 65535.

administrador del sistema agrega a cada usuario a uno o más grupos al momento de crear la cuenta. Los grupos permiten que el administrador asigne tareas y permisos de acceso sobre archivos y directorios a un grupo de usuarios sin la necesidad de asignarlo a cada usuario en particular.

Distribución de los archivos dentro del servidor

Para el almacenamiento de los archivos que contienen las versiones del software de Banco de México que será instalado en las terminales de las instituciones financieras se empleará un *file system* del mismo servidor que se utiliza como plataforma para soportar al RDBMS (Sybase), es decir, se usará al servidor Sun Ultra Enterprise 4000.

Para que el manejo de los archivos sea funcional, se requiere de una persona que se encargue de labores administrativas de este sistema de forma global. Esta persona contará con todos los privilegios sobre los archivos que se encuentren dentro del *file system*. Por cada sistema, tal como SPEUA, SAGA y SIAC, se tendrá una persona que se encargará de colocar las nuevas versiones de los sistemas y se encargará de mantener actualizado, este usuario tendrá todos los privilegios sobre los archivos referentes a su sistema, es decir no se le permitirá accesar a otros archivos que no sean los suyos, sólo se permitirá el acceso a su propio directorio. De la misma forma habrá una cuenta de consulta para todos los sistemas existentes, la cual será empleada por cada terminal cuando intente obtener el archivo que contenga la nueva versión del software. Como su nombre lo indica, esta cuenta sólo poseerá permisos de lectura sobre él o los archivos que contengan la versión del sistema que se desea obtener.

Para satisfacer los requerimientos de seguridad del sistema se ha destinado un *file system* para el almacenamiento de los archivos. En este *file system* se ha creado un directorio llamado versiones y dentro de él se ha creado un directorio para cada sistema. El dueño del directorio versiones será el administrador del sistema de versiones llamado admon_ver y pertenecerá al grupo version, mientras que cada subdirectorio tendrá como dueño a cada uno de los encargados de los diferentes sistemas (SAGA, SPEUA, etc.) llamados admon_speua, admon_saga, etc., y también pertenecerán al grupo version. En cuanto a los permisos, el directorio versiones tendrá todos los permisos asignados para el dueño, mientras que sólo se asignarán permisos de lectura y ejecución para los miembros del grupo (en este caso version). Todos los subdirectorios poseerán los permisos, además de tener habilitado el *sticky bit* para poder proporcionar la funcionalidad de que dos usuarios (la del administrador del sistema de versiones admon_ver y la del encargado del sistema) puedan contar con todos los accesos sobre los mismos archivos.

4.7 Diseño de Pantallas

Las pantallas que se pretenden presentar en el sistema son lo más sencillas posibles, de tal forma que se presente una interfaz amigable y fácil de operar al usuario.

Básicamente se manejarán tres tipos de pantallas que son las que se presentan a continuación.

Pantallas de Captura de Información

Estas pantallas servirán para que el usuario especifique el sistema que quiere transferir o instalar, así como el tipo de versión de la aplicación (Ultima, Anterior, Beta).

En la figura 4.2 se muestra el esquema de la pantalla de captura.

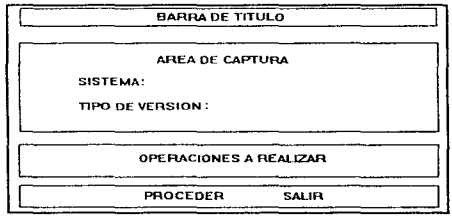


Fig. 4.2 Pantalla de Captura.

El área de captura es para especificar el sistema y la versión.

El área de operaciones a realizar es para que el usuario especifique si desea realizar una transferencia o una instalación.

Pantallas de Estado de Operación

Estas pantallas indican al usuario el estado en que se encuentra cierta operación.

Las distintas operaciones pueden ser la transferencia del sistema y la instalación del sistema.

En la figura 4.2 se muestra el esquema de esta pantalla.

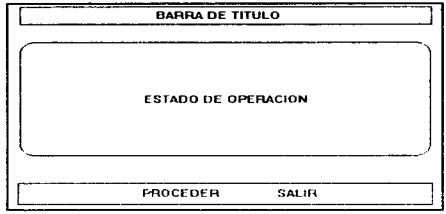


Fig. 4.3 Pantalla de estado de operación.

En el área de estado de operación el usuario podrá observar el *status* de cada actividad realizada por el sistema de Versiones.

En el caso de que se esté llevando a cabo una transferencia, en el área de estado de operación aparecerá una barra de estado que se irá recorriendo conforme al avance del FTP. Al finalizar esta operación la ventana desplegará los resultados de la transferencia.

En el caso de que se esté llevando a cabo una instalación, en el área de estado de operación aparecerá una ventana que indicará en donde se esta colocando cada archivo para poder ejecutar la aplicación, y que archivos son los que se están configurando para la correcta operación del sistema instalado.

Pantallas de Mensajes

Estas pantallas servirán para desplegarle al usuario distintos tipos de mensajes, tales como mensajes de error o mensajes de finalización de operaciones.

En la figura 4.3 se muestra el esquema de este tipo de pantallas.

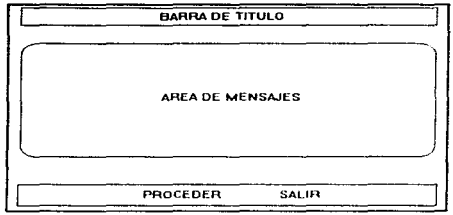


Fig. 4.4 Pantalla de Mensajes.

No sólo se podrá manejar el sistema de Versiones con el *mouse*, también se podrá hacer uso de la tecla tabular para navegar a través de los botones y el área de captura, así como del uso de combinaciones de teclas como ALT y alguna otra para no limitar al usuario al momento de hacer uso del sistema. Estas especificaciones se darán en el capítulo siguiente.

Haciendo mención a las pantallas , diremos que contarán con un fondo de color gris, los bordes de las ventanas serán lo más sencillos posibles, en todas las ventanas el usuario tendrá restringido el control de las mismas, solo podrá cerrarlas por medio del sistema. Para el manejo de todo el sistema, el usuario solo deberá hacer un *click* para ejecutar alguna tarea, salvo en el caso del área de captura donde debe especificar el sistema y la versión. Se procura desplegar mensajes de estado siempre que se esté realizando una operación para que el usuario no se desespere.

Una vez analizadas las herramientas de software y hardware, se decidió la plataforma de desarrollo del sistema. Basándonos en esta plataforma llegamos a las especificaciones de programas y distribución de archivos en el servidor.

Una vez vistos el plan de implementación, y el diseño de las pantallas, continuaremos con la presentación y descripción del mantenimiento del sistema de Versiones.

Capítulo
CINCO

IMPLANTACIÓN Y PRESENTACIÓN DEL SISTEMA

En este capítulo se analizarán todos los aspectos necesarios para realizar la implantación del sistema. Se tomarán en cuenta las características del lugar físico donde se operará, y se plantearán planes de prueba y entrenamiento, así como el procedimiento de respaldo de la información.

Al finalizar el capítulo se presentará un ejemplo de la ejecución del sistema indicando la instalación y la operación del mismo.

5.1 Preparación del Lugar Físico

En relación a la preparación del lugar físico se deben considerar las características generales de seguridad; aspectos físicos tales como las paredes y el techo, el piso falso, la iluminación, el control de temperatura, etc. El Banco de México cuenta con estas características para la instalación del servidor.

Dentro de las principales consideraciones tenemos las siguientes:

Puertas de Acceso

- Se tiene acceso restringido al personal.
- La autorización al área de cómputo será por medio de una tarjeta con un código.
- Las puertas son de doble hoja y con anchura total de 1.4 a 1.6 mts. por hoja.

- Se cuenta con una puerta de salida de emergencia.
- Se tomaron en cuenta las dimensiones máximas de los equipos, para facilitar el transporte del mismo.

Paredes y Techo

- Las paredes tienen pintura plástica lavable, para poder limpiarlas fácilmente.
- Están pintados el techo y las placas de plafón.
- Utiliza placas metálicas o de madera prensada para el piso falso con soporte y amarres de aluminio.
- La altura entre piso falso y plafón está aproximadamente entre 2.70 y 3.30 mts.

Piso Falso

- Cubre los cables de comunicación entre la unidad central de proceso y los dispositivos periféricos de conexión y cables de alimentación eléctrica.
- Permite que el espacio entre los dos suelos actúe como una cámara plena de aire.
- La altura está entre 40 a 60 cms.
- El piso es de un material sintético resistente al fuego.
- El piso está soportado por pedestales.
- El piso está cubierto con pintura antipolvo.
- Está considerada la resistencia eléctrica transversal del recubrimiento del piso falso, para evitar cargas electrostáticas.

Iluminación

- En el área de máquinas se mantiene un promedio mínimo de 450 luxes a 70 cms. del suelo
- Se evita la luz solar directa.
- La iluminación no se alimenta de la misma fuente que el equipo de cómputo.
- Del 100% de iluminación, se distribuye el 25% para iluminación de emergencia.

Vibraciones

- Se evita todo tipo de vibración.
- Si hay vibraciones superiores a las normales se estudian antes de colocar los equipos.

Acústica

- El suelo amortigua la transmisión de vibraciones a otras áreas.
- Las paredes evita que el ruido pase a los locales adyacentes.
- Las puertas estan siempre cerradas.

Aire Acondicionado

Se tiene en cuenta :

- Disipación térmica de las máquinas.
- Disipación térmica de las personas.
- Pérdida por puertas, ventanas, paredes, techos y suelos.
- Disipación de otros aparatos.
- Las cargas caloríficas del equipo de cómputo y sus periféricos las proporciona el proveedor, por lo común deben especificarse en BTU/Hora o en Kcal/Hora.
- El proveedor del equipo de cómputo también proporciona la cantidad de aire que requieren los ventiladores de los diferentes dispositivos, por lo regular en pies cúbicos por hora o en metros cúbicos por hora.
- Su alimentación eléctrica es directamente desde la Planta de Generación de Energía eléctrica para Emergencia; de ninguna manera se conecta a la salida del equipo NO-BRAKE, ya que por el encendido y apagado automático de motores y compresores ocasionaría disminución en el voltaje y ruido eléctrico al equipo de cómputo.

Condiciones de Temperatura y Humedad

- El proveedor indica las condiciones de Temperatura y Humedad, del equipo de cómputo.
- Cifras aproximadas : Rango de temperatura de 18 a 22 grados centígrados.
- Humedad relativa(HR) de 20% a 80%.
- Cuando el aire frío se inyecta directamente a los equipos, su temperatura no es inferior a 17 grados centígrados y su HR no es superior al 80%.

Filtros

- Se utilizan filtros de tipo absoluto con una eficiencia del 99% sobre partículas de 3 micrones.
- El aire de renovación o ventilación es tratado antes de ser introducido en la sala, tanto en temperatura y humedad como en filtrado.

Distribución de aire en la sala

- Los componentes de las máquinas se refrigeran mediante la circulación rápida de aire por ventiladores.
- La entrada de aire se efectúa por debajo de las máquinas, a través de rejillas.
- El aire de ventilación es en función del volumen de la sala. Se proyecta para obtener de 1.5 a 2 renovaciones por hora y para crear una sobrepresión que evita la entrada de polvo y suciedad por las puertas.
- En zonas contaminadas el aire de renovación se descontamina previamente.

- El calor disipado por los diferentes dispositivos de cómputo obliga a necesitar aire frío todo el año.

Distribución del aire por piso falso

- El espacio entre el suelo del edificio y el piso falso se utilizan como una cámara plena de aire.
- Todo el aire se descarga en la sala a través de rejillas en el suelo.
- El aire retorna a la unidad acondicionadora por rejillas en el techo.
- El sistema tiene controles de la temperatura del aire en el piso falso.
- Se cuenta con una cierta cantidad de recalentamiento para controlar la humedad relativa del aire antes de que entre en la sala.
- Se colocan cuidadosamente las rejillas y los retornos para no crear tiros de aire frío a caliente

Ductos

- Son de material que no desprendan partículas al paso del aire.
- No tienen revestimientos internos de fibras.
- Ningún objeto obstruye los ductos.

Protección contra incendios

- El equipo de cómputo está en un edificio resistente al fuego.
- Se cuenta con indicaciones de las salidas de emergencia.
- Las paredes, el techo falso y el piso son de material incombustible o resistente al fuego.
- El techo de la sala y área de almacenamiento de discos y cintas es impermeable.
- Hay de detección de humos, para aviso anticipado.
- El sistema de detección no interrumpe la corriente de energía eléctrica al equipo de cómputo.
- Existen suficientes extintores portátiles de CO_2, por todo el centro de cómputo.

Almacenamiento de información

- Las cintas, discos magnéticos y CD-ROM se almacenan en una sala aparte, y cuenta con todos los elementos de seguridad posibles.

Instalación eléctrica

- Se comprobaron, con el proveedor del equipo de cómputo, los voltajes de trabajo.

- La tolerancia en tensión no es mayor de 10% ni menor de 8% de la tensión nominal que especifica el fabricante del equipo de cómputo.
- La tolerancia en frecuencia es de ½ Hz.
- La variación de voltaje entre fase no es mayor de 2.5% de la media aritmética de las tres fases.
- Respecto al contenido de armónicas el máximo es inferior al 5%, con el equipo desconectado.
- La acometida de energía eléctrica que alimenta al equipo de cómputo está completamente independiente y a ella no se conecta ninguna otra carga, a fin de evitar interferencias.
- La acometida independiente llega desde el equipo de fuerza ininterrumpible y de ahí se alimenta un tablero de distribución que está situado en un lugar visible y accesible dentro de la sala de equipo de cómputo.
- La toma de tierra física es independiente, con una resistencia total de 3 ohms, que incluye conductor más electrodo.
- Las terminales, microcomputadoras e impresoras, que se localizan dentro del edificio de la sala de cómputo están alimentadas por energía eléctrica regulada y cuentan con una alimentación de tierra física.

5.2 Plan de Pruebas

El propósito de esta sección pretende determinar las etapas y metodología que se emplearán para que el sistema sea probado, es decir que pruebas deberán realizarse antes de que se entregue el sistema, de tal manera que el usuario cuente con una herramienta que contenga un margen de error mínimo.

La prueba de software es un elemento crítico para la garantía de calidad del software y representa un último repaso de las especificaciones, del diseño y de la codificación.

La creciente aparición de software como un elemento más de muchos sistemas y la importancia de los costos asociados a una falla de la misma están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software gaste el 40 por ciento del esfuerzo total del proyecto en la prueba.

Una serie de reglas que sirven acertadamente como objetivos de prueba son:

1. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Los objetivos anteriores suponen un cambio dramático. Nos quitan la idea que normalmente se tiene de que una prueba tiene éxito si no descubre errores. El objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Uno de los métodos en los que nos podemos basar para realizar pruebas es la metodología de prueba de la caja negra, la cual permite al analista derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales de un programa.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

1) Funciones incorrectas o correctas
2) Errores de interfaz
3) Errores en estructura de datos o accesos a bases de datos externas
4) Errores de rendimiento
5) Errores de inicialización y terminación.

Prueba de partición equivalente

La partición equivalente es un método de prueba de la caja negra, que divide el dominio de entrada de un programa en clases de datos de los que se pueden dar de entrada en la prueba. El diseño de los casos de prueba para la partición equivalente se basa en una evaluación de la clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de entradas válidas e invalidas para condiciones de entrada. Las clases de equivalencia se pueden definir de acuerdo con los siguientes directrices:

1. Si la condición de entrada especifica un rango, se define una clase de rangos válidos y dos inválidos para la prueba.
2. Si una condición de entrada requiere un valor especifico, se definen una clase de equivalencia válida y dos inválidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se definen una clase de equivalencia válida y una inválida.
4. Si una condición de entrada es booleana, se definen una clase válida y una inválida.

Para el caso de nuestra prueba se toca solamente el punto 1 anteriormente mencionado, en los cuales generaremos las siguientes pruebas:

Datos de entrada válidos	Datos de entrada Inválidos
Información de red correcta	Información de red inválida (2 casos)
Información correcta en pantalla de captura de información	Información incorrecta en pantalla de captura de información (2 casos)

Aplicando las entradas anteriores, la entrada válida deberá seguir el flujo normal del sistema, si el dato es inválido deberá indicar error de dato invalido.

Prueba de análisis de valor límite

Por razones que no están del todo claras, los errores tienden a darse más en los límites del dominio de entrada que en el centro. Es por esto que se ha desarrollado el análisis de valores límites como técnica de prueba. El análisis de valores límite nos lleva a una elección de casos de prueba que ejerciten los valores límite.

Las directrices del análisis son similares en muchos aspectos a las que proporciona la petición equivalente:

1. Si una condición de entrada especifica un rango limitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b y para valores justo por debajo y justo por encima de a y b, respectivamente.
2. Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.
3. Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, suponga que se requiere una tabla de temperatura frente a presión como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo y mínimo número permitido de entradas en la tabla.
4. Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

Para el proyecto en que estamos trabajamos, esta prueba queda sin su aplicación debido a que no contamos con datos que se rijan por cierto rango, es decir que los datos pueden tomar distintos valores y estos se verifican en la base de datos.

Prueba de técnicas de diagramas de causa y efecto

El uso de diagramas de causa-efecto es una técnica de diseño de caos de prueba que proporciona una concisa representación de las condiciones lógicas y sus correspondientes acciones. La técnica sigue cuatro pasos:

1. Se listan para un módulo las causas (condiciones de entrada) y los efectos (acciones), asignando un identificador a cada uno de ellos.
2. Se desarrolla un diagrama de causa y efecto.
3. Se convierte el diagrama en una tabla de decisión.
4. Se convierten las reglas de la tabla de decisión a casos de prueba.

El análisis de esta prueba en nuestro proyecto se establece de la siguiente manera y con los siguiente flujos.

1. La lista de causas y acciones es la siguiente

Causas

1.- Información del sistema de red
2.- Versión solicitada

Efectos

101.- Transmisión del archivo exitosa
102.- Mensaje de error de proceso
103.- Instalación de software exitosa
104.- Transmisión e instalación exitosa

2. Diagrama de causa y efecto

El diagrama esta expresado en la figura 5.1 en el cual se establecen las estradas y salidas, así como los procesos generales que intervienen:

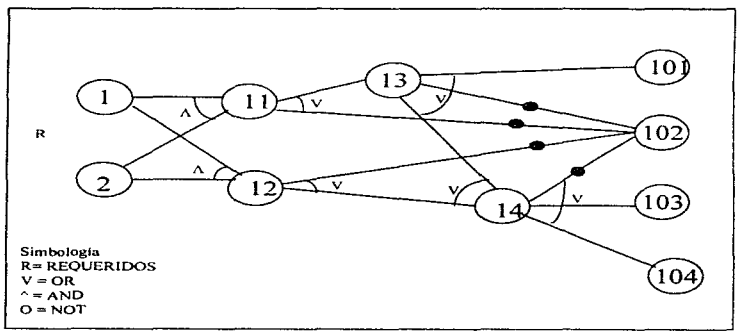


Fig. 5.1 Diagrama de causa y efecto.

En el diagrama se extraen cuatro procesos intermedios que intervienen para la toma de decisiones de nuestras salidas existentes, a estos procesos los llamaremos efectos intermedios y quedan listados de la siguiente manera:

Efectos intermedios

11.- Verificación de acceso de transferencia
12.- Verificación de requerimientos del sistema
13.- Verificación de existencia del instalador
14.- Ejecutar instalador

3.- La tabla de decisión.

La tabla de decisión 5.1 queda establecida de la siguiente manera, basándose en lógica boleana.

		PRUEBAS GENERADAS			
	TAREAS	CASO1	CASO2	CASO3	CASO4
CAUSAS	1	0	0	1	1
	2	0	1	0	1
	11	0	0	0	1
EFECTOS	12	0	0	0	1
INTERMEDIOS	13	0	0	0	1
	14	0	0	0	1
	101	0	0	0	1
EFECTOS	102	1	1	1	0
FINALES	103	0	0	0	1
	104	0	0	0	1

Tabla 5.1 Tabla de decisión.

En esta tabla hacemos referencia a los siguiente términos: posibles entradas boleanas en el diagramas se expresa como 0 = entrada incorrecta, y 1= entrada correcta. Los datos siguientes de los efectos intermedios y de los efectos finales indican 0 = salida de datos incorrecta y 1 = salida de datos correcta.

A raíz de la tabla antes descrita, se establecen varias etapas de prueba ingresando datos inválidos y validos, en donde se debe verificar que el efecto del sistema se cumpla.

4.- Efectos de prueba.

Los efectos de prueba quedarían establecidos de la siguiente manera:

A) Para datos incorrectos el efecto debe ser error.
B) Para un dato incorrecto y uno correcto debe ser error.

C) Para datos correctos el efecto debe ser el elegido en la entrada.

Prueba de validación de datos

La prueba de validación de datos conjuga un conjunto de técnicas especializadas que llenan las lagunas dejadas por otros métodos de prueba de la caja negra. La prueba de validación de datos está dirigida por la heuristica. Es decir, proporciona una serie de directrices o listas de comprobación para ayudar al encargado de la prueba, pero no presenta ningún análisis formal ni de ningún algoritmo detallado.

Cuando las órdenes son tecleadas, la interfaz del software implementa un programa de reconocimiento de cadenas que es lo que debe de ser validado. Las siguientes pruebas de validación de datos son muy apropiadas:

1. Especificar órdenes con sintaxis incorrecta, utilizar versiones de la forma correcta tanto evidentes como sutiles, es decir que ingrese datos aparentemente correctos y coherentes pero que no sean del todo correctos.
2. Proporcionar entradas sintácticamente correctas, pero que se encuentren fuera de secuencia o que estén especificadas en un momentos incorrecto.
3. Escribir una orden parcialmente correcta y a continuación terminar la entrada de ordenes.
4. Omitir todas las ordenes simplemente pulsar enter a todas las entradas.
5. Proporcionar ordenes correctas, pero con demasiados datos de parámetros.
6. Generar una interrupción del sistema inmediatamente después de haber introducido una orden, ya sea intermedia o completa.

Las reglas mencionadas se aplicarán de la misma manera en la que fueron descritas, a fin de llegar a una evaluación del sistema basado en la reglas anteriores.

5.3 Plan de entrenamiento

Una vez especificada la preparación del lugar físico y el plan de pruebas, podemos hablar del plan de entrenamiento.

Algunos elementos de documentación se pueden entregar a los usuarios antes de que el equipo esté configurado y el sistema quede instalado. Esta documentación puede ser manuales de usuario, donde se especifiquen los procesos que realiza el sistema, y tal vez una guia de conceptos y términos de computación. Los detalles técnicos se evitarán en este caso hasta que el equipo esté disponible.

Una vez que el sistema esté instalado y listo para operar, el usuario empezará a utilizarlo basándose en la documentación que recibió previamente.

Inicialmente, el analista debe mostrar al usuario como desarrollar cada tarea, y gradualmente dejarle la responsabilidad. A esto le debe seguir un periodo de interacción donde el analista esté disponible para responder preguntas y solucionar problemas, si es necesario. Gradualmente, el soporte técnico directo terminará.

El sistema de versiones será utilizado por dos tipos de usuario:

- Uno es el usuario operativo que será el que desde la terminal remota en cualquier sucursal financiera opere el sistema para transferir e instalar una aplicación
- Otro es el usuario de soporte que se encargará de la administración del sistema y brindará soporte a los usuarios operativos, desde el Banco de México.

Plan de entrenamiento del usuario operativo

- Entrega del Manual del sistema antes de la instalación
- Entrega del Manual de conceptos y términos para el usuario operativo
- Presentación del sistema
- Explicación de la operación general del sistema
- Explicación del proceso de transferencia
- Explicación del proceso de instalación
- Asesorías temporales

Plan de entrenamiento del usuario de soporte

- Entrega del Manual del sistema antes de la instalación
- Entrega del Manual de conceptos y términos para el usuario de soporte
- Presentación del sistema
- Explicación de la operación de los procesos del sistema
- Explicación de la administración del administrador de archivos
- Explicación de la administración de la base de datos
- Explicación de la transferencia vía FTP
- Capacitación para las asesorías telefónicas

5.4 Procedimientos de respaldo

Los respaldos son un prerequisito en cualquier sistema de cómputo, implica también que se tenga un lugar seguro de almacenamiento, ya que la información guardada en cintas de respaldo es extremadamente vulnerable. Lo anterior puede observarse en la operación de cualquier centro de cómputo, mientras la información reside en la computadora, los mecanismos de seguridad del sistema operativo protegen de cualquier intento de acceso no autorizado; sin embargo, cuando la información es respaldada en cinta, cualquier persona puede adueñarse de la cinta y revisar su contenido. Por esta razón deben protegerse los respaldos al igual que como normalmente se protege el equipo de cómputo.

A continuación se listan los procedimientos que normalmente se siguen para asegurar la información que se encuentra almacenada en cintas:

- No realizar respaldos en equipos que sean accesibles a las demás personas.
- No utilizar mensajeros para la transportación de cintas de respaldos.

- Eliminar los datos de las cintas o destruir físicamente las cintas antes de desecharlas.

Además deben verificarse periódicamente los respaldos para asegurarse que ellos contienen datos válidos. Es necesario revisar los respaldos de varios meses atrás ya que la información en medios magnéticos puede algunas veces, ser borrada por las condiciones ambientales. La única forma de saber que lo anterior está ocurriendo es restaurando el respaldo para verificar que la información es válida. Al menos una vez al año debe revisarse una muestra de las cintas de respaldo.

Para maximizar las posibilidades de recuperación de la información después de un accidente o un desastre, debe considerarse la posibilidad de almacenar las cintas de respaldo en un lugar diferente al que se encuentra el equipo de cómputo.

En la figura 5.2 se muestra la política de respaldo empleada tanto para la información que reside en *file systems* como para las bases de datos. El número que aparece en la esquina inferior derecha junto a cada cinta indica su identificador dentro del esquema de respaldos, el número que se encuentra en la esquina superior izquierda muestra el día del mes. Puede observarse que se emplea la misma cinta para los lunes durante todo el mes. Lo mismo aplica para los martes, miércoles y jueves: por el contrario, en el caso de los viernes se emplea una cinta diferente para conservar una copia como se encontraba la información al final de cada semana durante el último mes. Cabe mencionar que estas cintas vuelven a emplearse en el siguiente mes, esto es, en el primer viernes del siguiente mes se volverá a utilizar la cinta 5, al segundo viernes se usará la cinta 6 y así sucesivamente.

Al final de cada mes se guarda la cinta, en este caso se trata de la cinta 2 (del día martes) la que corresponde al fin de mes. A diferencia del caso de las cintas semanales, estas cintas no vuelven a reutilizarse ya que pasan a formar parte de la cintoteca del centro de cómputo.

Las cintas, como todos los medios de almacenamiento magnético, sufren un deterioro con el tiempo y con los procesos de grabación y lectura, por tanto, es importante que se vigile el número de veces que una cinta ha sido empleada, en este caso se tiene que cada cinta ha de grabarse como máximo en 30 ocasiones. Después de este número de grabaciones debe considerar el remplazo de la cinta.

Cabe mencionar que tanto en el caso de las bases de datos como en el de los *file systems* se realizan respaldos completos, ya que para recuperar la información de respaldos incrementales se requiere que los respaldos de todas las cintas involucradas estén sin ningún tipo de error; si alguna falla puede volver inservible todo el respaldo. Por el contrario en el caso de respaldos completos la integridad de los respaldos depende de una sola cinta lo cual minimiza la posibilidad de falla en el respaldo.

Todos los respaldos se efectúan por las noches para no sobrecargar la red en las horas de operación de los sistemas, además de que en la noche se garantiza que la información permanece sin cambios, lo cual es un requisito para garantizar la integridad lógica de los respaldos. En el caso de las bases de datos todas las actividades del RDBMS están controladas y dependen de la información que se encuentra en la bitácora de transacciones (*transaction log*), en esta bitácora se guarda el estado de las

transacciones que se han realizado o se encuentran en proceso. Cuando se realiza el procedimiento de respaldo, únicamente se respaldan las transacciones que se encuentren terminadas (*comitted transactions*). .Las transacciones que aún no hayan llegado a su fin serán descartadas por el proceso de respaldo. Por tanto, es necesario que todas las transacciones se encuentren terminadas para garantizar que toda la información será guardada en el respaldo.

Lunes	Martes	Miércoles	Jueves	Viernes
1  1	2  2	3  3	4  4	5  5
8  1	9  2	10  3	11  4	12  6
15  1	16  2	17  3	18  4	19  7
22  1	23  2	24  3	25  4	26  8
29  1	30  2	1  3	2  4	3  5

Fig. 5.2 Esquema de respaldos usado en *file systems* y bases de datos

En el caso de los *file systems* también se requiere que la información se encuentre sin cambios durante el tiempo que tarda en realizarse el respaldo; esto es , porque el proceso de respaldo se realiza en dos etapas: durante la primera etapa se obtiene la información de los *inodos* que se encuentran utilizados por los archivos; en la segunda etapa se almacenan los bloques de datos en los que residen físicamente los archivos. De lo anterior puede verse que si un archivo o directorio se modifica entre la primera y segunda etapa, puede provocar que algunos archivos y/o directorios se corrompan (en el respaldo) y la información se vuelva inaccesible.

Para realizar los respaldos de las bases de datos en Sybase se emplea una herramienta proporcionada por el proveedor llamada *backup server.*, la cual se trata de un *Open Server*[1] que se conecta con el RDBMS para realizar el respaldo. Adicionalmente al proceso de respaldo se realizan verificaciones de la integridad física de las bases de datos empleando la utilería dbcc, la cual permite revisar la asignación de páginas que forman la base de datos, así como también la estructura física de las tablas tanto de sistema como de los usuarios.

En el caso de los respaldos de los *file systems*, se emplea el comando ufsdump. Tanto en este caso como en el de las bases de datos, se emplean cintas de 4mm con capacidad de hasta 8 GB. Ambas herramientas de respaldo cuentan con la capacidad de realizar respaldos multivolumen, lo cual significa que si la información sobrepasa los 8 GB de capacidad de una cinta, pueden emplearse tantas cintas como se requieran.

5.5 Instalación y Ejecución del Sistema

El sistema de Versiones permite la transferencia e instalación de aplicaciones en terminales remotas de las instituciones financieras que realizan operaciones con el Banco de México.

Estos procesos se realizan de forma práctica y oportuna sin tener que disponer de recursos humanos y materiales.

Versiones es un sistema sencillo y fácil de usar en donde el usuario no requiere de conocimientos amplios de computación para poder usarlo.

A continuación se describirán detalladamente los procesos de instalación y ejecución del sistema.

5.5.1 Instalación

El proceso de instalación se puede llevar a cabo de dos formas. Una es desde discos y otra por medio del protocolo de transferencia de archivos (FTP).

Instalación desde discos

Para la instalación del sistema se necesitan 5 diskettes con los siguientes datos :

1. Disco #0 Instalación PB4, Sybase 10 y Versiones.
2. Disco #1 Generación de Instalación PB4.
3. Disco #2 Generación de Instalación PB4.
4. Disco #3 Generación de Instalación PB4.
5. Disco #4 Generación de Instalación PB4.

[1] Se llama Open Server al proceso servidor que es construido empleando las bibliotecas de funciones de Sybase llamadas *db-libraries* y que aceptan peticiones de procesos cliente para realizar alguna tarea en particular.

El Disco #0 se llamará Disco de Instalación. Para instalar cada uno de los programas cliente se debe de realizar lo siguiente:

Antes de instalar el programa de VERSION se deberá instalar PB4 (Power Builder 4) y SYBASE 10. La instalación debe realizarse desde el Explorador de Windows.

Instalación de PB4

1. Insertar el Disco #0 de Instalación en la unidad A.
2. Copiar el archivo a:\pb4.bat a c:\.
3. Ejecutar el archivo c:\pb4.bat, y seguir las instrucciones. Los discos que irá pidiendo el programa son los etiquetados como Generación de Instalación PB4.
4. Cuando aparece el mensaje "Insert the LAST disk of the backup set. Press a key when ready", debe de insertarse el Disco #4 y presionar enter
5. Cuando aparece el mensaje "Insert disk #1 - Press a key when ready", debe de insertarse el Disco # 1 y presionar enter
6. Al terminar el proceso de instalación el título de la ventana de MS-DOS cambia a Finalizado - Pb4. En este momento hay que cerrar la ventana
7. Borrar el archivo c:\pb4.bat
8. Reinicializar la computadora (si solamente va a instalar PB4, pero si desea instalar todo el módulo de Versiones puede reinicializar la computadora al final de la instalación de todos los programas cliente)

Instalación de SYBASE 10

1. Insertar el Disco #0 de Instalación en la unidad A.
2. Copiar el archivo a:\sybase.bat a c:\.
3. Ejecutar el archivo c:\sybase.bat y seguir las instrucciones
4. Al terminar el proceso de instalación el título de la ventana de MS-DOS cambia a Finalizado - SYBASE. En este momento hay que cerrar la ventana.
5. Borrar el archivo c:\sybase.bat
6. Reinicializar la computadora (si solamente va a instalar SYBASE, pero si desea instalar todo el módulo de Versiones puede reinicializar la computadora al final de la instalación de todos los programas cliente)

Instalación de VERSION

1. Insertar el Disco #0 de Instalación en la unidad A.
2. Ejecutar el archivo a:\version.bat
3. Al terminar el proceso de instalación el título de la ventana de MS-DOS cambia a Finalizado - VERSION. En este momento hay que cerrar la ventana.
4. Reinicializar la computadora.

En el caso de que en alguno de los programas no se haya efectuado la instalación de Windows, entonces se deberán realizar los siguientes pasos :

1. Buscar que exista el archivo de instalación de Windows en :
   PB4 C:\bmx\pb4\descomp\instalar.exe
   SYBASE 10 C:\bmx\sybase10\descomp\instalar.exe
   VERSION C:\bmx\version\descomp\instalar.exe

2. Si no existe, intentar de nuevo la instalación completa.

3. Si existe, ejecutar el archivo correspondiente.

Instalación por medio del protocolo de transferencia de archivos

La instalación del sistema de Versiones se puede llevar a cabo desde cualquier terminal remota conectada a la red financiera por medio protocolo de transferencia de archivos (FTP).

De esta forma ya no es necesario disponer de recursos humanos para la instalación.

La instalación se realiza de la siguiente manera:

1. Se necesita tener la máquina conectada a la Red Financiera.

2. Se deben crear los directorios de trabajo, de la manera siguiente:

   Abrir una sesión de Sistema Operativo (MS-DOS):
   Aparecerá una ventana de MS-DOS, con la línea *c:\windows\>*
   Escribir los siguientes comandos:

   c:\windows>**cd\ <enter>**
   c:>**md bmx <enter>**
   c:>**cd bmx <enter>**
   c:\bmx>**md pb4 <enter>**
   c:\bmx>**md pb4\ultima <enter>**
   c:\bmx>**md pb4\descomp <enter>**
   c:\bmx>**md sybase10 <enter>**
   c:\bmx>**md sybase10\ultima <enter>**
   c:\bmx>**md sybase10\descomp <enter>**
   c:\bmx>**md version <enter>**
   c:\bmx>**md version\ultima <enter>**
   c:\bmx>**md version\descomp <enter>**

3.Transferencia de sistemas básicos:

Ejecutar el siguiente comando:

c:\bmx>**ftp ftpbanxico <enter>**

Teclear la clave de usuario y el password proporcionados.

Aparecerá lo siguiente:

230 User versionbm logged in.
ftp>

4. Teclear la palabra **bin**, y presionar la tecla enter.

Aparecerá lo siguiente:

200 Type set to I.
ftp>

A continuación teclear el siguiente comando:

ftp>**get pb4/compacto.exe pb4\ultima\compacto.exe**

Al finalizar la transferencia teclear el siguiente comando:

ftp> **get sybase10/compacto.exe sybase10\ultima\compacto.exe**

Al finalizar la transferencia teclear el siguiente comando:

ftp>**get version/compacto.exe version\ultima\compacto.exe**

Al finalizar la transferencia teclear el siguiente comando:

ftp>**disconnect**

Aparecerá:

221 Goodbye.
ftp>

Teclear ahora:

ftp>**bye**

5. Verificar que se tengan los siguientes archivos:

c:\bmx\pb4\ultima\compacto.exe
c:\bmx\sybase10\ultima\compacto.exe
c:\bmx\version\ultima\compacto.exe

6. Cambiarse al subdirectorio c:\bmx\pb4\ultima (*c:\bmx>**cd pb4\ultima <enter>***), y teclear el siguiente comando:

*c:\bmx\pb4\ultima>***compacto \bmx\pb4\descomp**

7. Cambiarse al subdirectorio c:\bmx\sybase10\ultima (*c:\bmx\pb4\ultima>**cd \bmx\sybase10\ultima <enter>***), y teclear el siguiente comando:

*c:\bmx\sybase10\ultima>***compacto \bmx\sybase10\descomp**

8. Cambiarse al subdirectorio c:\bmx\version\ultima (*c:\bmx\sybase10\ultima>**cd \bmx\version\ultima <enter>***), y teclear el siguiente comando:

*c:\bmx\version\ultima>***compacto \bmx\version\descomp**

Instalación de Power Builder

9. Regresar al subdirectorio c:\bmx\pb4\descomp (*c:\bmx\version\ultima>**cd \bmx\pb4\descomp <enter>***), y teclear el siguiente comando:

*c:\bmx\pb4\descomp>***instalar**

En este momento aparecerá una ventana que indica el comienzo de la instalación de Power Builder.
Presionar el botón Aceptar.
Al finalizar la instalación aparecerá una ventana que indica reinicializar la computadora.
Presionar el botón Aceptar.

Instalación de Sybase10

10. Cambiarse al subdirectorio c:\bmx\sybase10\descomp (*c:\bmx\pb4\descomp>**cd \bmx\sybase10\descomp <enter>***), y teclear el siguiente comando:

*c:\bmx\sybase10\descomp>***instalar**

En este momento aparecerá una ventana que indica el comienzo de la instalación de Sybase10.
Presionar el botón Aceptar.
Al finalizar la instalación aparecerá una ventana que indica reinicializar la computadora.
Presionar el botón Aceptar.

Instalación de Versiones

11. Cambiarse al subdirectorio c:\bmx\version\descomp (*c:\bmx\sybase10\descomp>**cd \bmx\version\descomp <enter>***). y teclear el siguiente comando:

*c:\bmx\version\descomp>***instalar**

En este momento aparecerá una ventana que indica el comienzo de la instalación de Versiones.

Presionar el botón Aceptar.

Al finalizar la instalación aparecerá una ventana que indica reinicializar la computadora.

Presionar el botón Aceptar.

En la ventana de MS-DOS teclear:

c:\bmx\version\descomp> **exit <enter>**

Apagar el Sistema y después Reiniciar el Equipo.

5.5.2 Ejecución del sistema

Una vez realizado lo anterior, ejecutar el icono de Versión directamente en el grupo de programas creado con la última instalación.

En la pantalla que presenta teclear la información que se solicita tal como se muestra en la figura 5.3.

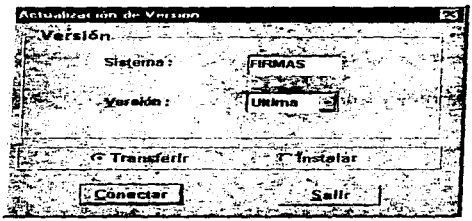


Fig. 5.3 Ventana inicial del sistema de Versiones.

En el campo de **SISTEMA** se debe digitar el nombre del sistema deseado, en el campo de versión seleccionar la versión del sistema a transferir, y marcar la opción de **Transferir.**

Presionar el botón **Conectar** para realizar la transferencia del sistema.

En este momento aparecerá una ventana con una barra de estado que irá indicando el avance del proceso de transferencia.

Es en esta etapa cuando el sistema se conectará a la base de datos para verificar que el sistema y la versión sean autorizadas para esa terminal y si es así se ejecuta la transferencia del archivo vía FTP.

Al finalizar la transferencia aparecerá una ventana como la que se muestra en la figura 5.4.

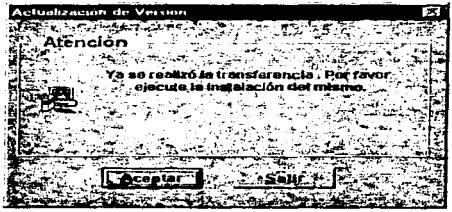


Fig. 5.4 Ventana que indica el fin de la transferencia.

Si desea realizar en ese momento la instalación debe seleccionar el botón de **Aceptar**, en caso contrario elegir el botón de **Salir** para realizar la instalación en el momento deseado.

En caso de realizar la instalación en otro momento ejecutar el icono de **Versión** y dentro de la pantalla que presenta, teclear la información que se solicita, elegir la opción de **Instalar** y presionar el botón de **Conectar**, tal como se muestra en la figura 5.5.

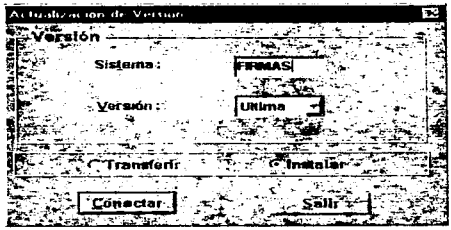


Fig. 5.5 Ventana del proceso de instalación.

En este proceso se realizará otra consulta a la base de datos para verificar si el sistema de la versión especificada pueden ser instalados en esa terminal. Esta verificación se lleva a cabo por motivos de control de seguridad, ya que el archivo compacto del sistema pudo haber sido obtenido por otros medios y no directamente desde esta terminal y con el sistema de Versiones.

Una vez verificados los derechos de instalación se realizará la descompactación de la aplicación antes transferida para poder hacer el copiado de los archivos en los subdirectorios especificados y la configuración del sistema.

Una vez analizados todos los aspectos necesarios para la implantación del sistema, se mostró el ejemplo de ejecución, de esta forma está todo listo para comenzar su operación.

En el siguiente capítulo se realizará un análisis de los objetivos planteados al inicio de este trabajo, para evaluar si se cumplieron o no, obteniendo así los resultados y las conclusiones que pueden ser útiles para versiones posteriores.

Capítulo
SEIS

RESULTADOS Y CONCLUSIONES

Con la realización de este trabajo se ha conseguido automatizar la distribución y actualización de versiones de software en Instituciones Financieras, si bien ésta ha sido la aplicación a la cual se ha destinado, puede observarse que es aplicable a cualquier tipo de empresa que requiera administrar y distribuir las versiones de software que esté empleando.

Esto es importante por varias razones:

- En el aspecto del soporte técnico, es más fácil resolver problemas de una versión del mismo software que de varias versiones.
- En el aspecto de la capacitación, es más simple capacitar a un usuario en el empleo de una versión de la herramienta que en el caso de contar con múltiples versiones.
- Cuando se desea evitar posibles incompatibilidades provocadas por el manejo de diversas versiones del mismo software.
- Se facilita el trabajo de los usuarios, ya que no requieren manejar al mismo tiempo distintas versiones del mismo paquete.

El sistema desarrollado emplea, en la parte del servidor de archivos, el servicio de ftp para la distribución del software, lo cual permite que pueda ser migrado a cualquier equipo de cómputo que soporte el protocolo TCP/IP y que tenga la posibilidad de levantar servicios de ftp, sin tener que realizar alguna modificación, ya que el servicio de ftp proporciona una interfaz estándar independiente del equipo y del sistema operativo en el que se esté ejecutando. Podemos encontrar una gran variedad de

equipos que cumplen con los requerimientos mencionados anteriormente, desde PC's ejecutando Windows 3.11 hasta equipos *mainframe*.

Es importante mencionar también, que el uso de una biblioteca de enlace dinámico (DLL) para la obtención de los archivos del servidor mediante el uso de ftp, posibilita que la aplicación pueda ser migrada a casi cualquier ambiente de desarrollo de aplicaciones para Windows, ya que el uso de la DLL es uno de los mecanismos fundamentales para la programación en Windows.

De forma práctica se ha observado que se han mejorado los siguientes aspectos en el servicio a las Instituciones Financieras:

- Mayor capacidad de servicio, ya que se reducen los tiempos de atención a las diversas instituciones por parte del área de soporte.

- Disminución efectiva en el tiempo de distribución de las versiones de software. Lo cual es extremadamente útil cuando se descubre algún error o falla en uno de los componentes del software. De esta forma también se estandariza el proceso de distribución a nivel nacional del software de Banco de México, sin importar la localización geográfica de las sucursales de cada Institución Financiera. También facilita que se realizen continuas mejoras al software, ya que éste puede ser rápidamente distribuido a los Bancos para realizar pruebas y en caso de ser exitosas, implantarse en el ambiente de producción.

- Incremento en la seguridad de la información, ya que se elimina casi por completo la intervención humana en el proceso de distribución e instalación de aplicaciones, lo cual evita que se haga mal uso del software que anteriormente se distribuía por medio de mensajería. También se elimina la posibilidad de corrupción del software durante su traslado, ya que anteriormente se distribuía en discos flexibles, los cuales son susceptibles a sufrir daño debido a que se trata de un medio de almacenamiento magnético.

- Actualmente se tiene un mejor control sobre las versiones de software que cada una de las Instituciones Financieras tienen instaladas.

- Se han reducido los costos administrativos, ya que ahora no se requiere elaborar un conjunto de discos para cada Institución Financiera, por el contrario, únicamente se elabora un archivo compacto, que contiene todos los archivos necesarios para instalar y ejecutar la versión de software por cada aplicación; tampoco es necesario contar con un sistema de mensajería que se encargue de la distribución del software. De la misma forma se ven reducidos los costos debidos a la papelería (discos flexibles).

- La seguridad es un elemento muy importante en las aplicaciones bancarias, se ha visto que se tiene un esquema bastante confiable que trabaja a varios niveles: en la interacción con la base de datos, se requiere de una cuenta de usuario válida para poder conectarse con el sistema; al nivel de la conexión con el servidor de archivos, se requiere igualmente una cuenta de usuario

para poder obtener los archivos que son de interés para la versión que se desea; de la misma forma, el acceso a los archivos está controlado por el esquema de usuarios y permisos diseñado para la aplicación; al nivel de la ejecución del sistema de distribución y actualización de versiones, se tiene que sólo las máquinas que cuentan con una dirección IP determinada puedan conectarse al sistema.

Las posibles mejoras que pudieran realizarse en el futuro al sistema de versiones son las siguientes:

- Se agregará una validación que permita verificar la dirección Ethernet de la tarjeta de red de cada una de las terminales; aún no se ha implantado porque este tipo de validación implica que en caso de daño de la tarjeta de red, la Institución en cuestión deba notificar al Banco de México del cambio de dirección Ethernet. Hay que recordar que esta dirección es única y es asignada desde el momento de la manufactura por el fabricante, a diferencia de la dirección IP, la cual se trata de una dirección lógica que es asignada por el administrador de la red, por lo que se trata de un excelente medio de validación; sin embargo, hay que tomar en cuenta las implicaciones operativas de realizar esta validación.

- Es probable que las siguientes versiones de este sistema puedan implementarse en una Intranet; con este esquema se tendría un cliente 'ligero' conocido comúnmente como *browser* (dentro de los más comunes podemos encontrar a Microsoft Internet Explorer, Netscape y Mosaic), y un poderoso servidor de Web desde el cual puedan ser ejecutadas las aplicaciones. Aún está lejos el día en el que esto sea realidad, ya que existen limitaciones tecnológicas para poder implantar un esquema de este tipo. Dentro de estas limitaciones podemos incluir el desempeño de las aplicaciones de Internet y las velocidades a las que trabajan los equipos de comunicaciones. Además un punto muy importante es que la seguridad aún es un tema muy discutido cuando se refiere a aplicaciones que se ejecutan en un entorno de este tipo.

- Finalmente, también cuando la tecnología de software lo permita, probablemente pueda implementarse un esquema de varias capas que a diferencia del tradicional esquema cliente/servidor, permita la existencia de varios procesos en los que se distribuya la carga de la aplicación y pueda lograrse de este modo un mejor desempeño.

La realización de este trabajo contribuyó al reforzamiento de los conocimientos adquiridos durante la carrera de Ingeniería en Computación, tales como el diseño del sistema, el desarrollo e implementación; para lo cual nos basamos en temas vistos en materias como Base de Datos, Ingeniería de Programación, Estructura de Datos, etc.

Gracias a los conocimientos adquiridos en las materias antes mencionadas, y en la experiencia laboral fué posible resolver el problema aplicando técnicas de ingeniería.

BIBILIOGRAFÍA

- **David McClanahan**
  PowerBuilder 4. A developers guide
  Ed. M&T Book
  New York 1995

- **Henry F. Korth, Abraham Silberschatz**
  Fundametos de Bases de Datos
  Ed. Mc Graw-Hill
  Noviembre 1987

- **Herbert Schildt**
  Programación en C y C++ en Windows
  Ed. Osborne McGraw-Hill
  Madrid 1995

- **James Martin**
  Security Accuracy and Privacy in Computer systems
  Ed. Prentice-Hall
  New Jersey. 1973

- **John C Dvorak y Nick Anis**
  Telecomunicaciones para PC
  Ed. Mc Graw Hill
  México 1992

- **Tere Parnell**
  Guía de redes de alta velocidad
  Ed. Osbornw / Mc Gaw Hill
  España 1997

- **Timothy Parker, Ph D.**
  Aprenda TCP/IP en 14 días
  Ed. Prentice Hall
  México 1994

- **Roger S. Pressman**
  Ingeniería del software un enfoque práctico
  Ed. Mc Graw Hill
  España 1990.

- **Ruben Adad, Miguel Angel Medina**
  Fundamentos de las Estructuras de Datos Relacionales
  Ed. Megabyte
  México, 1992

- **Solaris 2.4 TCP/IP Network Administration Guide**
  Sunsoft
  Pag. 1-24, 13
  USA,1994

- **SYS ADMIN. The Journal for UNIX Systems Administrators**
  'Interprocess Communications Between UNIX and MVS Applications Using Sockets'
  Volumen 5, Número 8
  Ed. Miller Freeman Inc.
  Agosto,1996

- **Unisys**
  FTP reference manual
  USA 1993

- **Kris Jamsa, Ken Cope**
  Programación en Internet
  Ed. Mc Graw-Hill
  México 1995

- **William S. Davis**
  Systems analysis and Design a structured Approach
  Oxford Ohio, 1982

- **DBMS and Internet Systems**
  http ://WWW.dbmsmag.com
  Driving Development
  DBMS- Abril 1997

- **Sistema instructor sobre economia de la empresa agricola**
  Julio Berbel Vecino
  http ://apolo.lcc uma.es/tea/cap13/capit13.html

- **Modelo OSI**
  http ://www.itam.mx/Info/Extuni/sat/comp/redes/conceptos.html

- **Datapro Client/Server analyst, client-server computing : emerging trends, solutions and strategies**, 1994
  http ://hidra.uniandes.edu.co/articulos/cliser.html

Apéndice
"A"

SELECCIÓN DEL TIPO DE INVERSIÓN PARA PROYECTOS

En este apéndice tomaremos como ejemplo la situación de un agente que cuenta con ahorros y quiere decidir donde colocarlos, como apoyo para ejemplificar el uso de las herramientas de calculo de inversión. Es decir, comparar inversiones alternativas y seleccionar la más adecuada. La selección y evaluación de inversiones exige estimar recursos, entradas, costos y beneficios a lo largo de la vida de una inversión a medio y largo plazo.

Comprar papelería de computo no es invertir, ya que este será consumido a lo largo del ciclo productivo, por tanto es un gasto que se emplea dentro del corto plazo.

Hay otro concepto equívoco que es llamar invertir a comprar una casa o unos activos financieros, joyas, oro, cuadros, etc. En sentido estricto esto es invertir, si bien la componente especulativa y de colocar los ahorros en algún refugio es la dominante en estos casos, por tanto, se aparta un poco de la inversión empresarial que es la que se verá a continuación.

Flujos de Caja

Cuando se hace una inversión obviamente se espera recuperar la misma cuanto antes mejor, esto se consigue cuando los flujos de caja, que son la diferencia entre los cobros anuales y los pagos anuales, son positivos y cubren pronto el pago de inversión.

A lo largo de la vida útil de toda inversión agraria se generan dos corrientes de signo opuesto: Cobros y Pago.

Pago de Inversiones

El pago de inversión es una cifra única que recoge todo lo que debe el inversor pagar el año inicial para conseguir la puesta a punto del proyecto.

Supongamos que nos encontramos con un proyecto de mejora de un centro de computo. Este pago de inversión recoge, por ejemplo:

a) Construcción de almacenes y centros de computo.
b) Adquisición de maquinaria y equipos.
c) Compra de equipo de computo.
d) Red de alimentacion eléctrica.
f) Gastos de estudios, proyectos e ingeniería del proyecto.

A veces, el pago del proyecto, que llamaremos "K", se descompone en los 2 ó 3 primeros años del proyecto, llamándolos K0, K1, K2, K3. Esto ocurre en proyectos de grandes dimensiones o en casos de grandes centros de computo.

El problema del tiempo

Cuando hacemos una inversión, el desembolso inicial es hoy, mientras que los pagos se irán produciendo en los próximos años. Obviamente el dinero de hoy vale más que la promesa de un dinero en el futuro. No podemos comparar pesos de 1992 con pesos de 1999, ya que no son unidades homogéneas.

Cuando sabemos que en el mercado de capitales podemos recibir un interés "i" por nuestro dinero, nos dará igual recibir hoy Q pesos o el año próximo Q(1+i) pesos. Este concepto es la base de lo que llamaremos actualización de los flujos. En la mayor parte de los métodos de evaluación de inversiones el año inicial es el que sirve de referencia.

De esta forma, si tenemos una serie de flujos R1, R2, R3... la actualización al momento actual se obtiene dividiendo cada año por el interés compuesto al tipo "i" durante "n" años.

Criterios de evaluación de inversiones

En un proyecto tenemos identificadas todas las variables. Los flujos de caja, Rj, el Pago de Inversiones K=10.000.000, y la Vida del Proyecto, n=5. Vamos a ver la forma más sencilla de evaluar la inversión. Para ello tenemos que definir una serie de criterios que resuman toda la serie de números que definen los flujos de caja de un proyecto en un sólo índice de la rentabilidad del mismo.

El Valor Actual Neto

Según la columna 8 de la tabla A.1 restamos el pago de inversión de los flujos actualizados que el proyecto genera, el resultado lo llamamos Valor Actual Neto (VAN). En nuestro ejemplo la fórmula queda establecida en la tabla A.2.

Los Flujos de Caja

1	2	3	4	5	6	7	8	9
0	0	0	10	0	-10	1,00	-10,00	-10,00
1	0	5	0	2	3	1,10	2,73	-7,27
2	0	5	0	2	3	1,21	2,48	-4,79
3	0	5	0	2	3	1,33	2,25	-2,54
4	0	5	0	2	3	1,46	2,05	-0,49
5	1	5	0	2	4	1,61	2,48	1,99

1=AÑO 2=COBRO EXTRA 3=COBRO ORDINARIO
4=PAGO EXTRA 5=PAGO ORDINARIO 6=FLUJO CAJA
7=COEFIC 8=FLUJO ACTUALIZ 9=SUMA

Tabla A.1 Flujos de caja durante un periodo de tiempo.

VALOR ACTUAL NETO

$$VAN = \frac{R1}{(1+i)} + \frac{R2}{(1+I)^2} + \ldots + \frac{Rn}{(1+i)^n} - K$$

$$VAN = \frac{3}{(1,1)} + \frac{3}{(1,1)^2} + \frac{3}{(1,1)^3} + \frac{3}{(1,1)^4} + \frac{4}{(1,1)^5} - 10$$

VAN= 2,73+2,48+2,25+2,05+2,48-10=1,99

Tabla A.2 Fórmula del cálculo del valor actual neto.

El VAN del proyecto es 1,990,000.00 pesos. Este concepto significa la Ganancia Neta del proyecto.

Si el inversor mete su dinero (10.000.000) en el banco al 10% de interés tendrá una rentabilidad 1.990.000 inferior a la que el proyecto genera. Visto de otra forma, si el VAN de un proyecto es cero, será indiferente dejar 10 millones de pesos en un banco y son remunerados al 10% de interés o invertir en un proyecto, en el momento en el que el VAN es mayor que cero, es más rentable el proyecto.

El plazo de recuperación

Este criterio no nos da un índice de la rentabilidad sino de la rapidez con la que se recupera la inversión. Se entiende por plazo de recuperación el número de años que pasan hasta que la suma de cobros actualizados se iguala a la de pagos actualizados.

Para calcular el plazo de recuperación basta con ir acumulando año por año los flujos de caja actualizados. De esta manera, se obtiene la tabla A.3.

Los Flujos de Caja

1	2	3	4	5	6	7	8	9
0	0	0	10	0	-10	1,00	-10,00	-10,00
1	0	5	0	2	3	1,10	2,73	-7,27
2	0	5	0	2	3	1,21	2,48	-4,79
3	0	5	0	2	3	1,33	2,25	-2,54
4	0	5	0	2	3	1,46	2,05	-0,49
5	1	5	0	2	4	1,61	2,48	1,99

1 = AÑO 2= COBRO EXTRA 3= COBRO ORDINARIO
4= PAGO EXTRA 5= PAGO ORDINARIO 6= FLUJO CAJA
7= COEFIC 8= FLUJO ACTUALIZ 9= SUMA

Tabla A.3 Periodo de recuperación.

Cuando el flujo acumulado y actualizado es mayor que cero, tenemos el plazo de recuperación, que en nuestro ejemplo son 5 años.

La Tasa Interna de Rendimiento

Hemos visto dos sencillos sistemas de evaluar una inversión: el VAN y el Plazo de Recuperación. Hemos visto cómo en ambos intervenía la tasa de interés como dato fundamental para calcular cada uno de los criterios. La tasa de interés varía anualmente

y depende del país o persona afectada. Por tanto, es conveniente contar con un criterio que no dependa del interés; ese criterio es la Tasa Interna de Rendimiento (TIR).

El inversor presta al proyecto K unidades monetarias y el proyecto le devuelve R1, R2,... Rn durante n años de vida del proyecto. Vamos a calcular los VAN para cada tipo de interés. A medida que el interés sube va descendiendo el Valor Actual Neto, hasta que toma valores negativos.

Cuando un VAN toma un valor negativo para un tipo determinado de interés, significa que es más rentable para el inversor colocar su dinero a ese tipo de interés que dejarlo en el proyecto. En el punto exacto en el que el interés hace que el VAN sea cero, el proyecto está en el límite de rentabilidad.

A ese interés que hace el VAN igual a cero lo llamamos Tasa Interna de Rendimiento (TIR).

La Vida del Proyecto

La vida del proyecto suele fijarse un poco arbitrariamente en 10, 20 ó 30 años. A veces la vida tiene un periodo más realista y determinado por los condicionantes biológicos del proyecto. Si pudiéramos aconsejar una cifra, nos parece que un horizonte de 10 años es el más adecuado a no ser que haya razones muy importantes que nos hagan variar este plazo.

Resumen

Este tema ha tratado de dar una introducción a la selección de inversiones. Para ello es necesario comprender en qué consiste el proceso de Actualización de una serie de Cobros y Pagos distribuidos en distintos momentos del tiempo.

Una vez entendemos el concepto de actualización, debemos comprender por qué se usan cobros y pagos para construir los flujos de un proyecto de inversión.

Con estos antecedentes pasamos a estudiar tres medidas de la rentabilidad de un proyecto: el VAN, el Plazo de Recuperación y el TIR.

CÓDIGO DE LA DLL UTILIZADA EN LA TRANSFERENCIA DE ARCHIVOS

```
#include <owl.h>

#include <stdio.h>
#include <io.h>
#include <string.h>

#include <ftpapi.h>
#include <ftpstat.h>

#define WM_INICIO                       WM_FIRST+300
#define WM_CHECALOGIN                   WM_FIRST+301
#define WM_RECEPCION                       WM_FIRST+304

char SERVIDOR_FTP[80]; //                  /*Dirección    IP    del
servidor FTP */
char ID_USUARIO[80];    //                    /*Login del usuario
*/
char CONTRASENA[80];    //                       /*Contraseña del
usuario */
char Archivo[120];      //
char Legaria[80];

//#define ORIGEN                                "resumen.txt"
//#define REGRESO
     "resumen.ftp"
char argv[100]="";
```

```
static FTPHANDLE ftpHnd;                        /*FTP handle */

//extern "C" {
// int far pascal _export Transmite(void);
//}
//PTModule TheModule;
//int TheStatus;

_CLASSDEF(TMyWindow)
class TMyWindow : public TWindow {
public:
  TMyWindow(PTWindowsObject  AParent,  LPSTR  ATitle,  PTModule
Module=NULL);
  ~TMyWindow();
  virtual void WMCreate(RTMessage Msg)
    = [WM_CREATE];
  virtual void WMDestroy(RTMessage Msg)
    = [WM_DESTROY];
  virtual void WMInicio(RTMessage Msg)
    = [WM_INICIO];
  virtual void WMChecaLogin(RTMessage Msg)
    = [WM_CHECALOGIN];
  //virtual void WMTransferencia(RTMessage Msg)
  //  = [WM_TRANSFERENCIA];
  virtual void WMRecepcion(RTMessage Msg)
    = [WM_RECEPCION];
  void ERROR_QUIT(LPSTR mensaje);
  void ERRORN(LPSTR mensaje);
  void LeeParametros(LPSTR Server,  LPSTR  User,  LPSTR  Password,
LPSTR Archivo, LPSTR Legaria);
};

TMyWindow::TMyWindow(PTWindowsObject AParent, LPSTR ATitle,
                     PTModule Module )
  : TWindow(AParent, ATitle, Module)
{
}

TMyWindow::~TMyWindow()
{
}


void TMyWindow::WMCreate(RTMessage Msg) {
 TWindow::WMCreate(Msg);
 Show(SW_HIDE);
 LeeParametros(SERVIDOR_FTP,   ID_USUARIO,   CONTRASENA,   Archivo,
Legaria);
 if(!PostMessage(HWindow,WM_INICIO,0,0)) {
   ERRORN( "FTP:No se pudo Mandar Mensaje" );
   PostQuitMessage(0);
 }
}
```

```
void TMyWindow::WMInicio(RTMessage) {
 if (FtpGetHandle(&ftpHnd) == FALSE) {
    ERRORN("FTP:Falla General !");
    PostQuitMessage(0);
 }
 /* FTP Login procedure */
 if (AsyncFtpLogin(ftpHnd,SERVIDOR_FTP,ID_USUARIO,CONTRASENA, "",
                          HWindow, WM_CHECALOGIN) == FALSE)
    ERROR_QUIT("FTP:Usuario o contraseña invalida");
}


void TMyWindow::WMChecalogin(RTMessage Msg) {
 if (Msg.WParam != Success)
    ERROR_QUIT("FTP:Falla de Registro");
 if( !PostMessage(HWindow,WM_RECEPCION,0,0) )
    ERROR_QUIT("FTP:No se pudo Mandar Mensaje");
}



void TMyWindow::WMRecepcion(RTMessage) {
 //char sauxl[80], sauxr[80];
 //if( access( , 0) ) {
 //  return;
// }else {
    /* se recibe al archivo  */
    if (AsyncFtpRecvFile(ftpHnd, Legaria, Archivo, TYPE_I,
                         HWindow,WM_DESTROY) == FALSE)
      ERROR_QUIT("FTP:Problemas en la Recepcion");
// }
}


/*void TMyWindow::WMOtra_transferencia(RTMessage) {
 if( !PostMessage(HWindow,WM_DESTROY,0,0) )
    ERROR_QUIT("FTP:No se pudo Mandar Mensaje");
} */

void TMyWindow::WMDestroy(RTMessage) {
 FtpCloseConnection(ftpHnd,0);
 PostQuitMessage(0);
}

void TMyWindow::ERROR_QUIT(LPSTR mensaje) {
 MessageBox(HWindow, mensaje, "ERROR", MB_ICONSTOP);
 PostMessage(HWindow, WM_DESTROY, 0, 0);
}

void TMyWindow::ERRORN(LPSTR mensaje) {
 MessageBox(HWindow, mensaje, "ERROR", MB_ICONSTOP);
```

```
}

void TMyWindow::LeeParametros(LPSTR Server, LPSTR User, LPSTR
Password, LPSTR Archivo, LPSTR Legaria) {

// MessageBox(HWindow, argv, "Linea de comando", MB_ICONSTOP);
sscanf(argv, "%s %s %s %s %s", Server, User, Password, Archivo,
Legaria);

/*
 GetPrivateProfileString("Server", "Server", "hola", Server, 80,
"miftp.ini");
 GetPrivateProfileString("User", "User", "sys_valores", User, 80,
"miftp.ini");
 GetPrivateProfileString("Password",      "Password",      "lprinc",
Password, 80, "miftp.ini");
 GetPrivateProfileString("File", "File", "resumen", File, 80,
"miftp.ini");
*/
// MessageBox(HWindow, Server, "Server", MB_ICONSTOP);
}

/* ********************************************** */


class TMyApp : public TApplication {
 public:
  TMyApp(LPSTR AName, HANDLE hInstance, HANDLE hPrevInstance,
         LPSTR lpCmdLine, int nCmdShow)
    : TApplication(AName, hInstance, hPrevInstance,
          lpCmdLine, nCmdShow)
      { strcpy(argv, lpCmdLine);};
  virtual void InitMainWindow();
};

void TMyApp::InitMainWindow() {
  MainWindow = new TMyWindow(NULL, Name);
}


int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow) {

 TMyApp     MyApp("Transferencia     de     archivos",     hInstance,
hPrevInstance,
                  lpCmdLine, nCmdShow);
 MyApp.Run();
 return MyApp.Status;
}



/*
```

```
int far pascal _export Transmite() {
  PTWindow MainWindow;
  MainWindow = new TMyWindow(NULL, "Transferencia de archivos",
TheModule);
  return MainWindow != NULL;
}

int FAR PASCAL LibMain(HANDLE hInstance, WORD /*wDataSeg*/,
  WORD /* cbHeapSize */, LPSTR lpCmdLine)
{

  TheModule = new TModule("CDLG DLL", hInstance, lpCmdLine);
  TheStatus = TheModule->Status;
  if ( TheStatus != 0 )
  {
    delete TheModule;
    TheModule = NULL;
  }
  return (TheStatus == 0);
}

int FAR PASCAL WEP(int /*bSystemExit*/)
{
  if ( TheModule )
    delete TheModule;
  return 1;
}

*/
```

GLOSARIO

3GL
Lenguaje de tercera generación, por ejemplo C ó Pascal.

4GL
Lenguaje de cuarta generación, usualmente se emplea como sinónimo de *front-end*.

ActiveX
Es un subconjunto de la tecnología denominada *Active Platform* de Microsoft. Proporciona interoperabilidad entre plataformas a través de la red. Por ejemplo, la tecnología ActiveX puede ser usada para almacenar datos, tales como una hoja de cálculo de Microsoft Excel en una página de Web, también pueden 'bajarse' aplicaciones desde un servidor de Web y estonces ejecutarse en una computadora cliente.

ANSI SQL-92
Es el estándar formal para SQL publicado en 1992 por el American National Standards Institute. Consiste de tres niveles: *entry*, *intermediate* y *full*.

API
(Application Program Interface). Interfaz de programa de aplicación. Lenguaje y formato utilizados por un programa para comunicarse con otro programa. También puede incluir los comandos utilizados para interrumpir a la computadora con el fin de llamar la atención a otro programa. Un API utilizado en comunicaciones se llama protocolo.

ARP
Protocolo de Resolución de Direcciones
Convierte las direcciones IP de 32 bits en direcciones físicas de red.

BLOB
(Bynary Large Object). Es un gran bloque de bits almacenados en una base de datos, tales como una imagen o un archivo de sonido.

breakpoint
Se trata de puntos de interrupción que se colocan en un programa para propósitos de depuración.

Browser
Cualquier programa utilizado para visualizar material contenido en el World Widw Web; Mosaic,MacWeb,Netscape son ejemplos de Browsers.

CASE
(Computer Aided Software Engineering). Se trata de una herramienta de software que implementa y automatiza alguna de las etapas del desarrollo de software.

C2
Es un nivel de seguridad que garantiza que el hardware no puede ser comprometido fácilmente, y que los usuarios deben conectarse al sistema mediante un identificador de usuario (*login*) y una contraseña. Con esta combinación pueden determinarse los derechos de acceso a programas e información que tiene cada usuario. Estos accesos están controlados no sólo por permisos sino también por niveles de autorización. Este nivel requiere que se audite el sistema.

CGI
(Common Gateway Interface). Es un estándar para ejecutar programas externos desde un servidor de Web. CGI especifica como pasar argumentos al programa en ejecución como parte de una solicitud HTTP. También define un conjunto de variables de entorno. Usualmente el programa generará una página HTML, la cual será mostrada en el *browser*. Un programa en CGI permite accesar la información en una base de datos y formatear el resultado como HTML.

compilador
Es un traductor que convierte un programa escrito en un lenguaje de alto o medio nivel como C ó Pascal en lenguaje de máquina o ensamblador. Posteriormente este programa objeto es procesado para su ejecución.

constraint
Un constraint es una restricción que es agregada a la definición de una tabla para limitar los valores de los datos que van a ser insertados.

cursor
Es un nombre simbólico para una instrucción select. Proporciona la capacidad de recorrer el contenido de una tabla, renglón por renglón.

data window
Son objetos desarrollados en los *front-end's* que presentan, manipulan, actualizan e imprimen informes de datos. Son empleados para automatizar la interfase entre una aplicación en ejecución y el usuario.

DDE
(Dinamic Data Exchange). Se trata al igual que OLE de una herramienta para intercambiar información entre aplicaciones.

DLL
(Dynamic Link Library). Biblioteca de enlaces dinámicos. Un conjunto de rutinas de programa que están disponibles para las aplicaciones en tiempo de ejecución.

DNS
(Domain Naming System). Es el segundo mecanismo del que se dispone para proporcionar el nombre de host para la resolución de dirección IP.

driver
Se denomina driver a cualquier controlador ya sea de un dispositivo de hardware o que permite la conexión hacia alguna aplicación, tal como una base de datos.

Ethernet
Protocolo estándar que funciona a 10 Mbps para una red de área local

file system
Se designa de esta manera al sistema de archivos y directorios de los sistemas operativos como Unix y MS-DOS.

Firewall
Es usado para separar una red local del mundo exterior. en general una red local es conectado al mundo exterior por una computadora llamada "gateway". Esta máquina puede ser convertida en un firewall, mediante la instalacion de un software especial que no permite que los paquetes de TCP/IP no autorizados fluyan desde el interior al exterior y viceversa.

front-end
Se trata de la herramienta para el desarrollo de aplicaciones cliente.

gateway
Es una aplicación que actúa como intermediaria para clientes y servidores que no pueden comunicarse directamente. Actúa como cliente y servidor, una aplicación gateway pasa peticiones desde un cliente hacia un servidor y regresa resultados desde el servidor hacia el cliente.

host
La computadora central o la computadora controladora en un entorno de procesamiento en tiempo compartido o distribuido.

hotkey
Se trata de alguna tecla o combinación de teclas asignada a alguna acción en una aplicación de Windows.

HTML
(Hypertext Markup Language) Se trata de un lenguaje sencillo que permite la distribución de una gran variedad de tipos de medios y vínculos hacia datos de muchas ubicaciones en Internet. Los vínculos de hipertexto, encontrados dentro del contexto de documentos formateados, dirigen los visualizadores de Web hacia varios recursos Internet. Los documentos WWW están en formato HTML.

HTTP
(Hypertext Transfer Protocol). Es el protocolo de transferencia de recursos del WWW, el cual permite que un servidor maneje más conexiones y distribuya datos de una forma más rápida que la mayoría de los protocolos existentes.

ICMP
Protocolo de Mensajes de Control Internet. Permite que los routers IP envíen mensajes de error y control a otros routers y anfitriones IP. Los mensajes ICMP viajan en los campos de datos de los datagramas IP y son una parte necesaria de todas las implementaciones IP.

IDE
(Integrated Development Environment). Se trata del entorno de programación que viene incluido en la mayoría de los lenguajes de programación, usualmente consta de un editor, compilador o intérprete, depurador y alguna otra herramienta.

Inodo
En el *file system* de Unix, se trata de un apuntador que asocia a un nombre de archivo con localidades físicas que se encuentran en los discos, estas localidades almacenan propiamente el contenido del archivo.

Integridad declarativa
Es la integridad que se agrega a la base de datos usando las instrucciones *create table* o *alter table*.

Integridad referencial
La integridad referencial asegura que la información entre las llaves primaria y foránea se mantiene consistente.

Internet
Es la mayor red mundial de computadoras, emplea el protocolo de comunicaciones TCP/IP.

Intérprete
Es un programa que traduce y ejecuta al mismo tiempo un programa escrito en un lenguaje de alto nivel como Basic ó Lisp, no produce ningún código de máquina o programa objeto, por lo que la ejecución es más lenta que en el caso de un programa compilado.

IP
Protocolo Internet. Un protocolo de bajo nivel que enruta paquetes de datos a traves de redes separadas vinculadas por routers para formar Internet o una Intranet.

ISAPI
Es la implementación de Microsoft para ejecutar tareas similares a las realizadas por CGI.

Java
Es un lenguaje de programación orientado a objetos de propósito general desarrollado por la compañía Sun Microsystems; entre sus principales características se encuentran: distribuido, interpretado, seguro, portátil y soporta el *multithread*. Su principal aplicación es en la programación para Internet en la forma de applets. Contiene una extensa colleción de rutinas para los protocolos TCP/IP, tales como HTTP y FTP. Las aplicaciones desarrolladas en Java pueden accesar objetos a través de la Internet por medio de URL's tan fácilmente como si estos se encontraran localmente.

keystroke
Se trata de una acción realizada sobre alguna tecla, tal como presionarla o 'liberarla'.

macro
Es un conjunto de instrucciones en Windows que realizan alguna tarea en particular. Tales instrucciones pueden incluir secuencias o combinaciones de teclas y acciones del ratón.

NFS
(Network File System). Sistema de archivo en red. Un sistema de archivo distribuido de Sun Microsystems, Inc., que permite que múltiples usuarios compartan datos en una red. NFS permite a los usuarios compartir los datos sin tener en cuenta el tipo de procesador, sistema operativo, arquitectura de red o protocolo.

NSAPI
Es la implementación de Netscape para ejecutar tareas similares a las realizadas por CGI.

ODBC
(Open Database Connectivity). Es un estándar definido por Microsoft para el acceso a datos almacenados en bases de datos relacionales, no relacionales e incluso formatos no de bases de datos para entornos Windows. El objetivo de ODBC es el acceso transparente desde un cliente a un servidor de base de datos.

OLE
(Object Linking and Embedding). Se trata de una de las herramientas más poderosas de Windows para compartir datos entre las aplicaciones.

ORB
(Object Request Broker). Se trata de un estándar para objetos distribuidos que está siendo desarrollado por el Object Management Group (OMG), el cual es un consorcio

de vendedores de software y usuarios finales. El nombre formal del estándar es CORBA (Common Object Request Broker Arquitecture), proporciona mecanismos en los cuales los objetos de forma transparente hacen solicitudes y reciben respuestas. Esta arquitectura soporta interoperabilidad entre objetos, construidos en diferentes lenguajes, corriendo en diferentes plataformas en un ambiente totalmente heterogéneo.

outer join
Es una operación join que muestra adicionalmente, los renglones que no satisfacen a la operación join, de una de las tablas sobre las que se especifica la operación.

Protocolos
Es el "lenguaje" que las computadoras de un red "hablan"

query
Es una solicitud de recuperación de información; usualmente se trata de una instrucción *select*, aunque a veces se le denomina *query* a cualquier instrucción SQL que manipule a los datos.

RDBMS
(Relational Data Base Management System). Se trata de un sistema manejador de bases de datos que emplea el modelo relacional para representar y almacenar la información.

RISC
(Reduced Instruction Set Code). Tecnología aplicada a los micropocesadores, de manera que contengan un conjunto reducido de instrucciones, para realizar software especializado para este tipo de arquitectura.

rpc
Se trata de una de las dos formas en la cual una aplicación cliente puede ejecutar un *stored procedure* en un servidor de base de datos. También se refiere a un *stored procedure* que es ejecutado en un servidor diferente al servidor en el que el usuario se encuentra conectado.

shell
Se trata de la capa más externa de una aplicación que proporciona la interfase con el usuario o con otras aplicaciones.

SNMP
(Simple Network Management Protocol). Es una parte del conjunto de protocolos de TCP/IP. Fué originalmente desarrollado en la comunidad de Internet para monitorear y resolver problemas en ruteadores y bridges. SNMP proporciona la habilidad de monitorear y comunicar información respecto al estado de estos dispositivos entre equipos que cuenten con este protocolo. Emplea una arquitectura distribuída que consiste de sistemas de administración y agentes.

SQL
(Structured Query Language). Lenguaje de Consulta Estructurado. Lenguaje utilizado para interrogar y procesar datos en una base de datos relacional.

stored procedure
Es una colección de instrucciones SQL precompiladas, almacenadas y ejecutadas bajo un mismo nombre.

TCP
Protocolo de Control de Transmisiones. Un protocolo orientado a conexiones que transmite información en flujo de bytes. La información se transmite en paquetes llamados segmentos TCP, que contiene encabezados TCP y datos.

trigger
Los *triggers* son un tipo especial de *stored procedure* que son específicos a una tabla y son manejados por eventos. Las instrucciones en el *trigger* son ejecutadas automáticamente por el servidor de base de datos cuando una instrucción de modificación de datos es realizada sobre la tabla a la cual se le definió el *trigger*.

two-phase-commit
Se trata de un método de alta consistencia para distribuir datos en diferentes servidores de base de datos. Se trata esencialmente de realizar un *commit* en dos servidores distintos para cada transacción realizada.

UDP
Protocolo de Datagramas de Usuario. Un protocolo libre de conexiones que transmite la información en paquetes llamados datagramas UDP. UDP es un protocolo "no confiable" ya que el transmisor no recibe información que indique si un datagrama fue en realidad recibido.

UNIX
Sistema Operativo con capacidad de multitarea y multiusuarios, comunmente usado en los servidores conectados a Internet.

vistas
Se trata de una instrucción *select* almacenada que se comporta como una tabla.

WWW
(World Wide Web). Es un sistema que proporciona acceso a una amplia variedad de recursos de Internet, bases de datos, documentos y archivos. El Web presenta la información en forma de documentos basados en hipertexto, que pueden incluir texto formateado, gráficos y elementos de audio y video.