UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

"CONSOLIDACIÓN DE SERVIDORES DEL AREA DE DESARROLLO DE SISTEMAS"

INFORME DE INGENIERIA

PARA OPTAR EL TITULO PROFESIONAL DE

INGENIERO DE SISTEMAS

EDWARD RAFAEL MAICELO MAICELO

LIMA – PERU
2003

Consolidación de Servidores del Area de Desarrollo de Sistemas.

DEDICATORIA

Este trabajo lo dedico a Dios y a mis padres que me han protegido y me ha guiado, en todo momento de mi vida y gracias a ellos he podido alcanzar los objetivos fijados y que son renovados cada día.

AGRADECIMIENTO

Agradezco a Dios por darme las fuerzas cada día de intentar ser mejor y a mis padres Orfelia y Mauro, que me apoyaron y creyeron en mí.

INDICE

DESCRIPTORES TEMATICOS

Consolidación de Servidores

Metodología de Sistemas.

Estándares de Programación.

Tuning de Herramientas de Programación.

CCC/Harvest.

VMWARE.

RESUMEN

Una de las cosas que uno siempre tien que hacer en todo trabajo es mirar la “Mancha en la Pared”, es decir preguntar por cosas que la mayoría supone que lo conoce, pero que en realidad no se pregunta por ser demasiado obvio. Partiendo de está premisa se elaboró el presente trabajo con la finalidad de poder mejorar la Administración de servidores de Desarrollo de sistemas, así como la Administración de fuentes de los diferentes aplicativos. Debido a que el día a día, a la premura en el desarrollo de aplicaciones, a los cambios de tecnología y la necesidad de brindar un buen servicio. Degenero que se cuenten con 63 servidores en el area de desarrollo de sistemas, los cuales estaban distribuidos en las diferentes áreas de la empresa. No teníendo ningún control sobre ellos. Los servidores de malograban continuamente, y se solicitaban nuevas instalaciones, lo cual ocasionaba gastos y lo que es peror el trabajo de desarrollo se paraba. La situación empezo a emperorar pues las solicitudes de compra de servidores continuaban, debido a lo que se menciona en líneas atrás. Similar a los problemas que teníamos con los servidores, ocurría con los fuentes de los aplicativos. Las fuentes de los aplicativos se encontraban almacenados en diferentes sitios, como servidores, PC del desarrollador y otros. Y lo peor es que existía diferentes versiones de los aplicativos y cuando se malñogrban los servidores, surgía los problemas de cual era la última versión. Todos estos problemas se intenta resolver con la consolidación de servidores, disminuyendo el número de servidores a 8 y con la Implementación de CCC/Harvest

que nos permitiría el control centralizado de las fuentes de todos los aplicativos del Area, así como la administración de todas las versiones de los mismos. Este proceso se tiene que llevar paso a paso, pues el cambio de la forma de trabajo de las personas no se puede ralizar de una forma brusca, si no que se tiene que realizar varias reuniones informandoles las ventajas y desventajas del proyecto y lo que es mejor tenemos que involucrarlos para que formen parte de nuetros equipo de proyecto recibiendo ideas y mejoras para el proyecto, con está prtenmisa se desarrollo unas encuentas en las cuales queriamos identificar los problemas que hoy en día tenemos y que soluciones podemos proponer, logrando así que las ideas de todos se vena reflejados en el proyecto. Se Realizó un piloto en el cual nos dimos cuenta de algunos problemas que se nos presentarán cuando se realice el proyecto en forma masiva y de algunas cosas que debemos de adecuar a nuetro plan general. La consolidación de servidores es un proceso lento y continuo, en la cual intervienen muchas personas.

CAPITULO I: GENERALIDADES

I. INTRODUCCIÓN

El presente trabajo se realizó con la finalidad de poder mejorar la Administración de servidores de Desarrollo de sistemas, así como la administración de fuentes de los diferentes aplicativos. Debido a que el día a día y la necesidad de brindar un buen servicio, había ocasionado que se cuenten con 63 servidores, los cuales estaban distribuidos en las diferentes áreas de la empresa. No se tenía un control sobre ellos y lo que es peor las solicitudes de compra de servidores continuaban, debido a lo que se menciona en líneas atrás. De igual forma ocurría con los fuentes de los aplicativos los cuales se encontraban almacenados en diferentes sitios, como servidores. PC del desarrollador. Y lo peor es que existía diferentes versiones de los aplicativos. Todos estos problemas se tratan de resolver con la consolidación de servidores, disminuyendo el número de servidores a 8 y con la Implementación de CCC/Harvest que nos permitiría el control centralizado de las fuentes de todos los aplicativos del Area, así como de todas las versiones de los mismos.

II. OBJETIVO

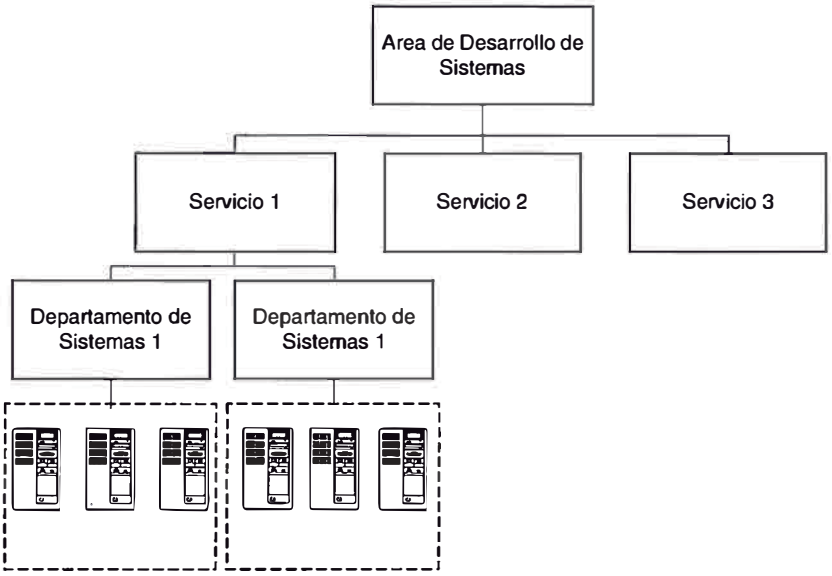
Lograr una administración eficiente de los servidores (manejo de versiones, mecanismos de seguridad, procesos de Backup, estándares, reusabilidad de código, etc.) así como de los aplicativos (fuentes, performance) del Area de Desarrollo de Sistemas (ADS). Para poder brindar el soporte técnico necesario y tener un control de los mismos.

CAPITULO II: METODOLOGIA Y PROPUESTA

I. ANTECEDENTES

El Cliente Servidor en la empresa se fue iniciando a principios del año 1996, como toda organización sufrió el proceso de crecer de golpe y de ingresar a una tecnología "nueva" la cual daba muchos logros a otras empresas. Por tal motivo se empezó desarrollando en PC's (Como servidores) y posteriormente se migraron a servidores nuevos. Con la llegada de la programación en tres capas y de la intranets/Internets, la utilización de servidores se incremento pues esta nueva forma de programación, requería tener ambientes separados para la BD, transacciones, páginas. Esto conllevo a que el Area de Desarrollo de sistemas tuviera en un momento 63 servidores de desarrollo (anexo9). Lo cual ocasionaba gastos en mantenimiento instalación y lo peor que los aplicativos requería servidores más robustos lo cual requería de repotenciar los anteriores o comprar otros, con la llegada de mecanismos como el Load Balancing produjo que el numero de servidores aumente también y todo esto trajo consigo a que la administración de servidores se tornará inmanejable, pues cada departamento contaba con sus propios servidores y los administraba de acuerdo a sus necesidades, y no estaban basadas en estándares por lo cual unir aplicativos era bastante complicado,

el gráfico a continuación muestra como eran administrados los servidores (En base a Departamentos de Desarrollo).



II. SITUACION ACTUAL

- Aumento acelerado del número de servidores del Area de Desarrollo de Sistemas.
- No existen estándares de para los Aplicativos así como para la BD, para la estructura de directorios y backups.
- No existe una metodología que permita mejorar el proceso de Desarrollo de los diferentes aplicativos.
- No existe una herramienta que permita controlar las diferentes versiones de los aplicativos, por dicho motivo el manejo de versiones de aplicativos no es la adecuada..
- No es fácil determinar donde están nuestros aplicativos y nuestros directorios de trabajo (Creación de estándares de Directorios de Trabajo).
- Los archivos de base de datos se encuentran mezclados con los archivos de base de datos y backups.
- Existen muchos backups de aplicativos en diferentes servidores (Mejor Administración).
- Muchos administradores de los servidores, lo cual ocasionan descontrol.
- No existe administración del espacio de los discos, información redundante.
- Los discos duros no se están usando correctamente
- Los directorios se encuentran compartidos para diferentes usuarios e incluso para usuarios fuera del servicio, lo que ocasiona Problemas de virus, Esto se produce porque no hay políticas de acceso a directorios, cualquier usuario puede acceder a los servidores, sin ninguna restricción.
- No existe mantenimiento ni administración de las Base de datos, componentes, fuentes de aplicativos, e información valiosa.

- Se encuentran instalados diferentes software, y no se cuenta con un documento de registro que contemple que software existe en cada servidor (No sólo BackOffice). Y Existen instaladores de software, en diferentes servidores.
- Los servidores no poseen mantenimiento.
- Los servidores están muy recargados, y tiene dificultades para realizarles un shutdown.
- Problemas frecuentes con los servidores y en especial con los discos.
- Los aplicativos cuentan con diferentes configuraciones, debido a los ServicePack de los S.O y de la Herramientas usadas. Actualmente existen 25 configuraciones de software base.

III. PROPUESTA

1. Definición de estándares para los diversos aplicativos.
2. Administración centralizada de los servidores.
3. Administradores de Servidores.
4. Rediseño de los directorios de trabajo.
5. Políticas de Backup.
6. Seguridad de los servidores.
7. Políticas de Uso de Servidores.
8. Mejoras a Servidores.

1. DEFINICION DE ESTANDARES PARA LOS DIVERSOS APLICATIVOS

Los Estándares a definir para los aplicativos C/S, Intranet, Internet son los siguientes, os cuales se detallan en los anexos:

Anexo1: Estándares de Visual Basic.
Anexo2: Estándares de PowerBuilder.
Anexo3: Estándares de ASP.
Anexo4: Estándares de SQL Server.
Anexo5: Estándares de Conexión a BD y HOST.
Anexo6: Estándares De Documentación de Programas.
Anexo7: Mejores Prácticas para VB, ASP y SQL.

2. ADMINISTACION CENTRALIZADA DE LOS SERVIDORES

La administración centralizada permitirá:

- **Revisión de estándares.**
  - ✓ Todos los aplicativos deben de cumplir con los estándares de desarrollo.
  - ✓ Deben de fijarse estándares de medición de la performance del aplicativo.
  - ✓ Revisión de la arquitectura a implementar la cual debe ser revisada desde su concepción, con la finalidad de que contemple todas las consideraciones de desarrollo.
  - ✓ Establecer estándares de componentes, así como de paquetes.
  - ✓ El uso de SourceSafe (CCC/HARVEST) es obligatorio en los aplicativos, pues nos permite tener mecanismos de control, de seguridad y respaldo de las versiones de los aplicativos.
- **Revisión estructuras a implementarse.**
  - ✓ La estructura a implementar en cada desarrollo debe de revisarse y utilizar la más eficiente, contemplando los estándares fijados por el Banco. Esto abarca tanto a nivel de cliente como de servidor.
  - ✓ Las herramientas a usarse deben de ser definidas y aprobadas.
- **Ahorro de espacio en disco, así como el número de servidores.**
  - ✓ La idea es evitar la redundancia de información.
  - ✓ Ordenar la información de acuerdo a un formato que nos sea familiar (formato de congelamiento).
  - ✓ Agrupar servidores para definir cuales son para guardar información de aplicativos y otros para los de BD.
  - ✓ Administrar la información a almacenar y de acuerdo a perfiles determinados.

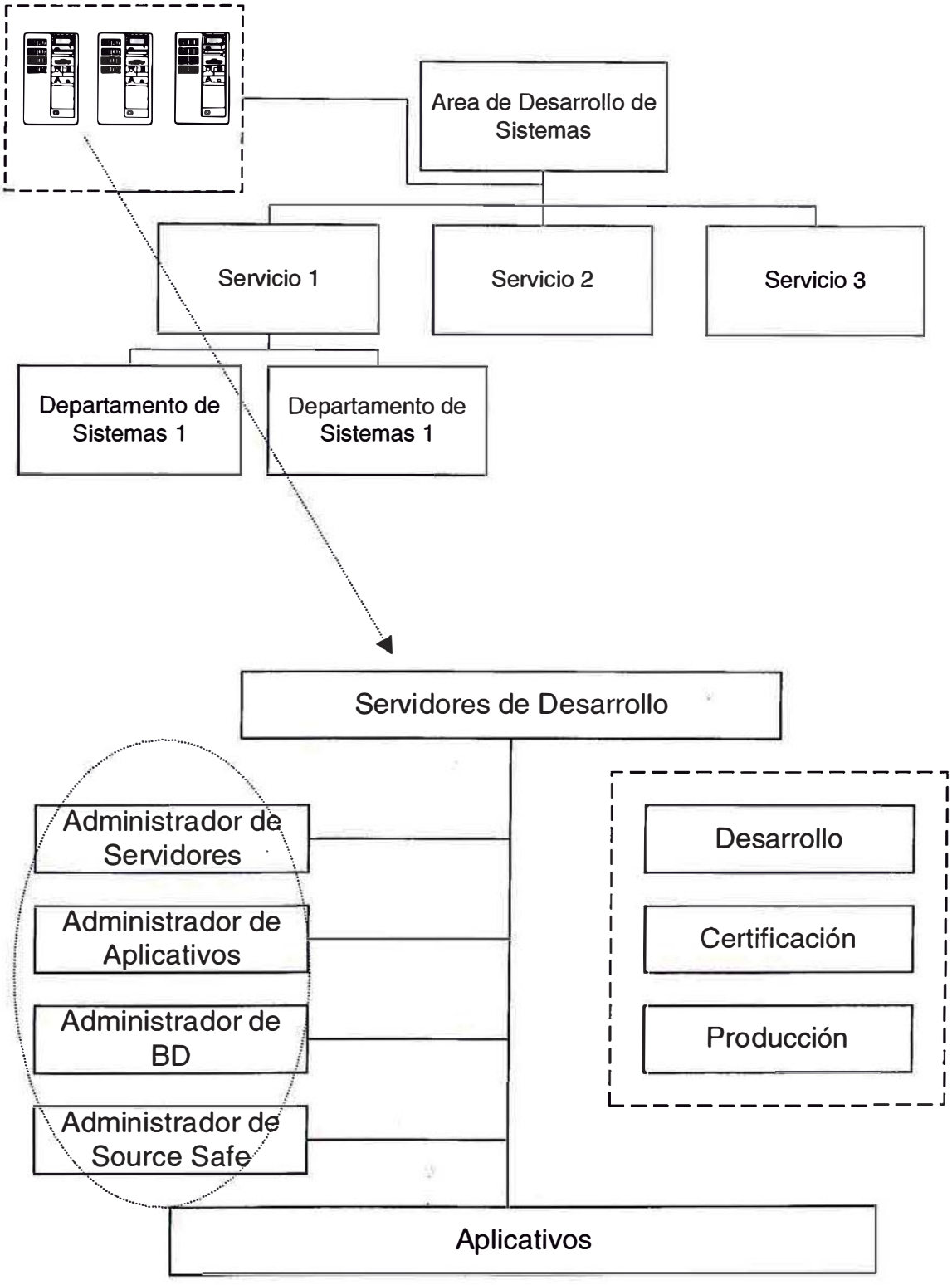
- ✓ La idea no es llenarnos de servidores, si no, repotenciar los mismos y "clonarlos" usando la herramienta VMware.
- ✓ Esta herramienta permite crear servidores virtuales que se comportan como si fuera uno nuevo, esto se comenta con mayor detalle en el informe de Implementación de ambientes.
- ✓ La administración de dichos servidores se puede realizar remotamente, para ello se pueden usar las siguientes herramientas:
  1. NetMeeting.

– **Orden y mejor administración de los servidores.**
  - ✓ Esta idea permitirá poner orden en nuestros servidores así como poder administrarlos y establecer mecanismos de seguridad.
  - ✓ La centralización de la documentación, permitirá realizar backups automáticos.
  - ✓ El uso del Visual SourceSafe permitirá centralizar en un sólo lugar las fuentes de los aplicativos y poder generar mecanismos de backup.
– **Mayor control sobre los aplicativos.**
  - ✓ Contar con un administrador nos permitirá, controlar el desarrollo de los aplicativos, en cuanto a performance, estándares, arquitectura, herramientas, etc.
  - ✓ También permitirá el desarrollo de aplicaciones más eficientes pues esta persona velara por el cumplimiento de las normas establecidas para los aplicativos.
– **La idea no es generar cuellos de botella.**
  - ✓ El presente documento no tiene la intención de generar cuellos de botella, si no más bien ordenarnos y permitir trabajar en un lenguaje que todos conozcamos. Se persigue desarrollar aplicativos con alta performance dotados de calidad y cumpliendo con las normas establecidas por el banco.

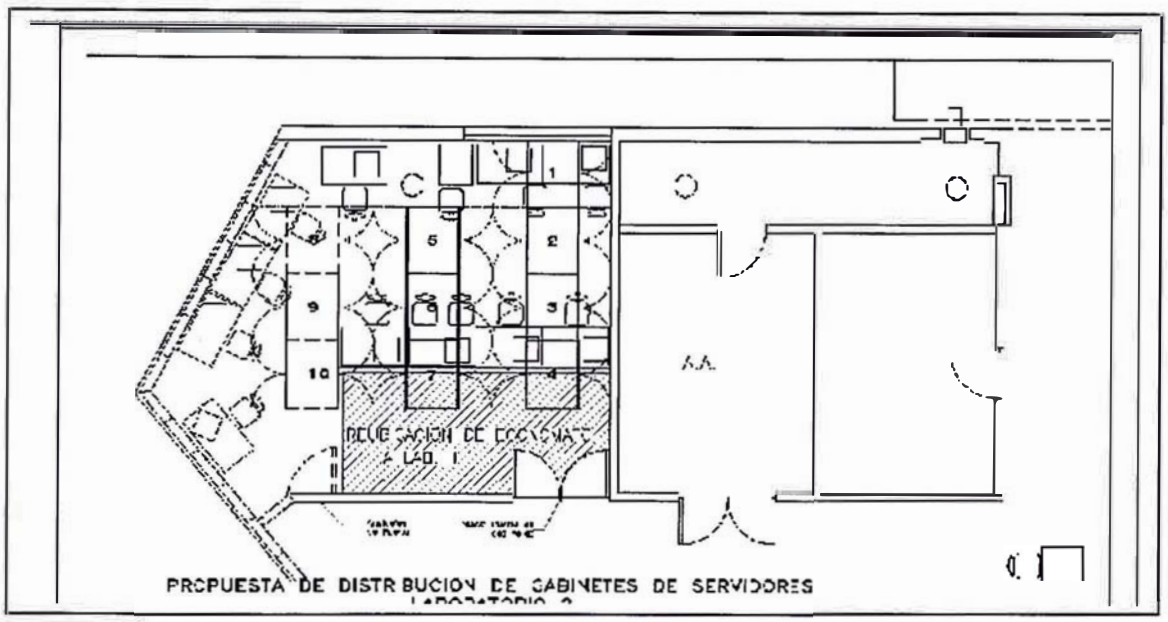
- La idea es ordenar y velar por la estructura a implementar en cada aplicativo.
- Contar con documentación detallada de los aplicativos (diagrama de procesos, entidad relación, casos de uso, etc.).
- Facilitar la carga de datos de producción al ambiente de desarrollo, con la finalidad de realizar pruebas de strees, procesos y carga de datos.
- El administrador de estos servidores hará las veces de **logistic Manager**, en los diferentes proyectos. Para lograr esta función cada responsable del proyecto debe de realizar un plan de inducción y entregar de toda la documentación concerniente al aplicativo.
- El administrador de aplicativos debe comportarse como un staff de Control de Calidad Técnico (QAT).
  - ✓ Disminuir tiempos de proceso (performance de componentes, stored procedured y aplicativos).
  - ✓ Proponer diversas estrategias de poder cumplir un objetivo, en el desarrollo de componentes, stored procedured y aplicativos.
- Conseguir un estándar Para todos el Software Base, obteniendo una única configuración para los diferentes aplicativo o disminuirlas a su mínima expresión.

ALCANCE

Los aplicativos y servidores del ADS, tendrán el siguiente, Esquema de Administración.



Para mejor administración de los servidores y poder cumplir con los objetivos trazados, se ha visto conveniente confeccionar un Centro de Computo en el cual, se alojaran todos los servidores del Area de Desarrollo de sistemas, de la siguiente manera:



El centro de Computo tendrá las siguientes Características:

	Detalle	Cantidad
1	Gabinetes de servidores de 2 cuerpos (8 server x rack)	8
2	Switch para 16 monitor de servidores	4
3	Instalaciones de Puntos de Red para Servidores	64
4	Instalaciones eléctricas para 8 rack	16
5	Tablero Eléctrico de15 circuitos para c/u de los rack	1
6	Switch Fast Ethernet 48 ports	2

7	Trabajos de acondicionamiento de ambiente	1
8	Aire Acondicionado 2 equipos 30kbtu + 1 backup	3
9	Falso Piso, en metros cuadrados	48
10	Seguridad - Sensores de Detención	10
11	Control de Acceso	1
12	Cámara Filmadora	1

En el cual se ubicaran los 63 servidores con los cuenta el Area de Desarrollo De Sistemas, posteriormente se realizará el proceso de "*consolidación*" de servidores.

3. ADMINISTRADORES DE SERVIDORES

Para la administración eficiente de los servidores y de los aplicativos se propone contar con los siguientes administradores:

A. Administrador de servidores de Aplicativos.
B. Administrador de Aplicativos.
C. Administrador de Visual SourceSafe (para todos los Aplicativos).
D. Administrador de servidores de BD.

Los cuales tendrán las siguientes funciones:

A. Administrador de servidores.

1. Administrar y velar por el buen funcionamiento de los servidores.
2. Definir la forma de uso de cada uno de los servidores así como roles y permisos.
3. Llevar un control del software instalado.
4. Llevar una bitácora del comportamiento de los aplicativos, cuales hacen disminuir la performance y degradan el funcionamiento del servidor.
5. Llevar un control de los errores de los aplicativos y así como cual es la causa y posibles soluciones.
6. Realizar mecanismos de backup automáticos sobre los directorios de trabajo y de aplicativos.
7. Velar por la administración del espacio físico de los servidores.

B. Administrador de Aplicativos

1. Velar por el cumplimiento de estándares de los aplicativos.
2. Definir y plantear estructura eficientes para los diferentes aplicativos.
3. Definir y desarrollar un control de calidad de aplicativo, realizando pruebas de esfuerzo y definiendo estándares de performance requeridos.

4. Definir mejores practicas para el desarrollo de aplicativos, con la finalidad de mejorar la performance y asegurarnos la buena calidad de los mismos.
5. Resolver problemas de los desarrolladores a nivel técnico, con la finalidad de mejorar el desarrollo y resolver diversos problemas.
6. Participar como logistic en los diferentes proyectos, pues es conocedor de los diversos aplicativos y de las nuevas tecnologías
7. Definir los documentos necesarios par que un aplicativo se instale en el ambiente de certificación (en desarrollo).
8. Encargarse de los procesos de carga de datos de producción a desarrollo, con la finalidad de realizar pruebas con data real, buscando minimizar tiempos de respuesta y anticiparnos a posibles problemas en la puesta a producción.
9. Ver todo lo referente a las necesidades de los aplicativos en producción, certificación y desarrollo.
10. Proponer mecanismos de mejor control de errores de aplicativos con la finalidad de poder detectar errores fácilmente.
11. Utilizar herramientas de administración remota de servidores, como el NetMeeting (con la finalidad de mejorar el proceso de administración de los aplicativos).

C. Administrador del SourceSafe

1. El administrador creara la estructura de directorios del SourceSafe (De acuerdo Al estándar de directorios de aplicativos, definidos más adelante (punto 4)).
2. Permitirá el acceso de los usuarios a dichos directorios, el cual se realizará por aplicativo.
3. Cada vez que se inicie un nuevo requerimiento, el administrador creara el proyecto y definirá cuales son los componentes a usar, por el aplicativo.

4. Los componentes serán, los últimos congelados para así poder hacer un seguimiento de modificaciones realizadas por cada proyecto.

D. Administrador de servidores de BD

1. Administrar y velar por el buen funcionamiento de las base de datos (fragmentación, tamaños, etc.).
2. Definir la forma de uso de cada uno de los servidores así como roles y permisos.
3. Llevar un control de los objetos y posible redundancia de objetos.
4. Llevar una bitácora del comportamiento de los aplicativos, cuales hacen disminuir la performance y degradan el funcionamiento de la BD.
5. Llevar un control de los errores de los aplicativos y así como cual es la causa y posibles soluciones.
6. Realizar mecanismos de backup automáticos de las BD y así como rutinas automáticas de limpieza del log de la Base de Datos.

- ✓ Los servidores seguirán un proceso de mantenimiento, definido por los administradores.
- ✓ Todas estas políticas que se acuerden estarán documentadas.

5. REDISEÑO DE LOS DIRECTORIOS DE LOS SERVIDORES

Los directorios de aplicativos guardaran una semejanza con la estructura definida para el proceso de congelamiento. Esto, con la finalidad de tener un orden y hacernos más fácil el proceso de congelamiento, administración en forma más eficiente las versiones de nuestros aplicativos, así como los directorios de trabajo.

La estructura de directorios para los aplicativos será de la siguiente manera:

A. APLICATIVOS

APLICATIVOS

1.Nemonico – Nombre Aplicativo1

1.Mnemónico - VERSION 1.1 – Nro. ticket

1.FUENTES

1. SERVIDOR

1.PAGINAS

2.COMPONENTES

2.CLIENTE

2.SCRIPTS

1.BD

1.TABLAS

2.SP

3.USUARIOS - PERMISOS

4.ACTUALIZACION DE DATOS

3.INSTALADORES

1.CLIENTE

1.WINDOWS 95

2.WINDOWS NT

2.SERVIDOR

1.APLICACIONES

2.COMPONENTES

3.FLUJOS WORKFLOW

4.ACTUALIZADOR AUTOMÁTICO DE VERSIONES (si el aplicativo lo contempla)

1.SERVIDOR

5.EJECUTABLES

1.CLIENTE

2.SERVIDOR

6.MANUALES

1.USUARIO

1.CLIENTE

2.SERVIDOR

2.INSTALACIÓN

1.CLIENTE

2.SERVIDOR

1.Mnemónico - VERSION 1.1 – Nro. ticket

2. Mnemónico - Nombre Aplicativo2

B. VISUAL SOURCESAFE/CCC-HARVETS

La estructura de directorios en el SourceSafe será de la siguiente manera:

APLICATIVOS

1.Nemonico – Nombre Aplicativo1

1.Mnemónico - VERSION 1.1 – Nro. ticket

1.FUENTES

1. SERVIDOR

1.PAGINAS

2.COMPONENTES

2.CLIENTE

2.SCRIPTS

1.BD

1.TABLAS

2.SP

3.USUARIOS - PERMISOS

4.ACTUALIZACION DE DATOS

3.INSTALADORES

1.CLIENTE

1.WINDOWS 95

2.WINDOWS NT

2.SERVIDOR

1.APLICACIONES

2.COMPONENTES

3.FLUJOS WORKFLOW

4.ACTUALIZADOR AUTOMÁTICO DE VERSIONES (si el aplicativo lo contempla)

1.SERVIDOR

5.EJECUTABLES

1.CLIENTE

2.SERVIDOR

6.MANUALES

1.USUARIO

1.CLIENTE

2.SERVIDOR

2.INSTALACIÓN

1.CLIENTE

2.SERVIDOR

1.Mnemónico - VERSION 1.1 – Nro. ticket

2. Mnemónico - Nombre Aplicativo2

C. DIRECTORIOS DE TRABAJO

La estructura de directorios para los aplicativos será de la siguiente manera:

TRABAJO

1.Nemonico – Nombre Aplicativo1

1. Mnemónico - VERSION 1.1 – DD/MM/YY

1.FUENTES

1. SERVIDOR

1.PAGINAS

2.COMPONENTES

2.CLIENTE

2.SCRIPTS

1.BD

1.TABLAS

2.SP

3.USUARIOS - PERMISOS

4.ACTUALIZACION DE DATOS

3.INSTALADORES

1.CLIENTE

1.WINDOWS 95

2.WINDOWS NT

2.SERVIDOR

1.APLICACIONES

2.COMPONENTES

3.FLUJOS WORKFLOW

4.ACTUALIZADOR AUTOMÁTICO DE VERSIONES (SI EL APLICATIVO LO CONTEMPLA)

1.SERVIDOR

5.EJECUTABLES

1.CLIENTE

2.SERVIDOR

6.MANUALES

1.USUARIO

1.CLIENTE

2.SERVIDOR

2.INSTALACIÓN

1.CLIENTE

2.SERVIDOR

2. Mnemónico VERSION 1.2 – DD/MM/YY

2. Mnemónico - Nombre Aplicativo2

Para esta estructura tendremos las siguientes consideraciones:

1. Serán administrados por el DFS (Distribución Fiel Systems), el cual permite administrar todos los directorios compartidos de diferentes servidores. Los cuales serán alojados en una ruta lógica determinada pero físicamente estarán alojados en lugares diferentes logrando así, la centralización y administración deseada.
2. Serán ubicados en un sólo lugar en lo posible (por motivos de espacio, esto puede variar).

3. La estructura de fuentes en el SourceSafe seguirá la misma estructura de directorios.
4. Los directorios de trabajo son temporales y deben ser creados con Fecha.
5. Se debe de respetar la secuencia de directorios
6. El mnemónico debe de ser de máximo 4 caracteres
7. Él número de versiones es limitado 4 versiones por aplicativos y el resto se realizaran los backups correspondientes.
8. Sólo determinados usuarios podrán tener acceso de full control y podrán crear directorios.
9. Estos permisos serán otorgados de acuerdo a los aplicativos
10. Sólo estos directorios estarán compartidos.
11. Los responsables de los aplicativos serán responsables de sus directorios.
12. No se podrá compartir ningún directorio, adicional a los ya compartidos.
13. El administrador creara la estructura de directorios del Visual SourceSafe, este realizara el acceso de los usuarios a dichos directorios. Cada vez que se inicie un requerimiento, el administrador creara el proyecto y definirá cuales son los componentes a usar, por el aplicativo y así poder hacer un seguimiento de modificaciones realiza cada proyecto.

E. INSTALADORES

La estructura de directorios para los instaladores será de la siguiente manera:

INSTALADORES

SERVIDOR

BD

CLIENTE

OTROS

Para esta estructura tendremos las siguientes consideraciones:

1. Los instaladores se ubicaran en un sólo lugar.
2. Sólo se almacenaran instaladores necesarios y de uso frecuente, los demás se eliminaran.

F. COMPONENTES

La estructura de componentes será de la siguiente manera:

1. Componentes de desarrollo
   - Para el almacenamiento de los componentes de desarrollo, se debe seguir la siguiente estructura:
     D:\Componentes
     D:\Componentes\Genéricos, el cual almacenará las "dll" que son utilizadas por todos los aplicativos
     D:\Componentes\Aplicativos\<Nombre de la aplicación>, donde se almacenará los componentes propios de la aplicación.
   - Para la exportación de paquetes se mantendrá un esquema similar al punto anterior:
     D:\Paquetes Exportados
     D:\ Paquetes Exportados\Genéricos, el cual almacenará los paquetes de los componentes genéricos que pueden ser utilizados por los diferentes aplicativos.
     D:\ Paquetes Exportados\Aplicativos\<Nombre de la aplicación>, donde se almacenará los paquetes de componentes propios de cada aplicación

2. Componentes de producción

- Para el almacenamiento de los componentes de producción, se debe seguir la siguiente estructura:

  D:\ComponentesProd

  D:\ComponentesProd\Genéricos, el cual almacenará las "dll" que son utilizadas por todos los aplicativos

  D:\ComponentesProd\Aplicativos\<Nombre de la aplicación>, donde se almacenará los componentes propios de la aplicación.

  Para la exportación de paquetes se mantendrá un esquema similar al punto anterior:

  D:\Paquetes Exportados Prod

  D:\ Paquetes Exportados Prod\Genéricos, el cual almacenará los paquetes de los componentes genéricos que pueden ser utilizados por los diferentes aplicativos.

  D:\ Paquetes Exportados Prod\Aplicativos\<Nombre de la aplicación>, donde se almacenará los paquetes de componentes propios de cada aplicación

3. Componentes de contingencia
   - Para el almacenamiento de los componentes en caso de contingencia, se debe seguir la siguiente estructura:

     D:\ComponentesConting

     D:\ComponentesConting\Genéricos, el cual almacenará las "dll" que son utilizadas por todos los aplicativos

     D:\ComponentesConting\Aplicativos\<Nombre de la aplicación>, donde se almacenará los componentes propios de la aplicación.

- Para la exportación de paquetes se mantendrá un esquema similar al punto anterior:
  D:\Paquetes Exportados Conting
  D:\ Paquetes Exportados Conting\Genéricos, el cual almacenará los paquetes de los componentes genéricos que pueden ser utilizados por los diferentes aplicativos.
  D:\ Paquetes Exportados Conting\Aplicativos\<Nombre de la aplicación>, donde se almacenará los paquetes de componentes propios de cada aplicación
  Estos directorios deberán ser eliminados una vez el servidor origen se encuentre operativo.

5. POLITICAS DE BACKUP

Los backups se dividirán en dos backups de aplicativos y directorios de trabajo y backups de BD.

Backups de aplicativos y directorios de trabajo.

1. Se realizaran una vez por semana.
2. Seguirá la siguiente estructura:
   - BK_APL_DDMMYY
   - BK_VSS_DDMMYY
3. Esto se almacenarán por un periodo determinado.
4. El proceso se realizará en forma automática y se realizara en horas fuera de trabajo.
5. Si se desean realizar backup extemporáneo se coordinara con el administrador.
6. Los backups en disco en se eliminaran.
7. Se usaran tapes de backup para realizar esta tarea.

Backups de BD.

1. Se realizaran una vez por semana, y será en forma integral.
2. Seguirá la siguiente estructura:
   - BK_NOMBREBD_DDMMYY
3. Esto se almacenarán por un periodo determinado.
4. El proceso se realizará en forma automática y se realizara en horas fuera de trabajo.
5. Si se desean realizar backup extemporáneo se coordinara con el administrador.
6. Los backups en disco en lo posible se eliminaran.
7. Se usaran tapes de backup para realizar esta tarea.

6. POLITICAS DE USO DE SERVIDORES

1. Sólo existirá un administrador para los servidores.
2. Todos los usuarios harán uso de los servidores con sus respectivos usuarios
3. Cada usuario tendrá un perfil de acuerdo a las funciones que realiza (ninguno será 0administrador).
4. Los directorios de trabajo y de aplicativo sólo estarán accesibles a un grupo determinado (responsables de aplicativos) los demás usuarios tendrán acceso sólo de lectura.
5. Los usuarios que tengan permisos de full control, tendrán que definir sus directorios de acuerdo al estándar definido anteriormente.
6. Estos serán responsables de sus directorios y de la información almacenada.
7. Los usuarios que infrinjan estas medidas de seguridad serán comunicados y amonestados.
8. Sólo los directorios de aplicativo y trabajo estarán compartidos.

9. El administrador definirá con los usuarios que posible directorios se compartirán.
10. No se podrá compartir ningún directorio, adicional a los ya compartidos.
11. No se compartirán directorios a usuarios que no pertenezcan al dpto.
12. Se tendrá registrado todos los aplicativos instalados en el servidor y sólo se instalaran de acuerdo a las políticas definidas por el administrador.
13. Las caídas de servidor serán anotadas y se analizaran las causas y posibles implicados en dichas caídas.
14. Una vez depurado los directorios de los servidores se ejercerá un control frecuente sobre los mismos.
15. No se podrán instalar aplicativos en el servidor sin la comunicación con el administrador.
16. Para la instalación de aplicativos propios se contaran con PC's de trabajo, las cuales no servirán de prueba de instaladores y aplicativos.
17. La creación de BD será coordinada con el administrador.

7. SEGURIDAD DE LOS SERVIDORES

1. Se contara con administradores quienes velaran por la integridad de los mismos.
2. Se mejorara el acceso a los servidores por parte de los usuarios.
3. Se contara con usuarios de Mts para así no depender de personas y además tener un ambiente similar al de producción.
4. Se crearan grupos de usuarios los cuales tendrán roles determinados sobre los directorios, de acuerdo a esta estructura:
   - Gro-NrombreAplicativo-Read
   - Gro-NrombreAplicativo-Change

- Gro-NrombreAplicativo-Full Control
- etc.

5. Sólo determinados usuarios podrán tener acceso de full control y podrán crear directorios.
6. Estos permisos serán otorgados de acuerdo a los aplicativos
7. Sólo estos directorios estarán compartidos.
8. Los responsables de los aplicativos serán responsables de sus directorios.
9. No se podrá compartir ningún directorio, adicional a los ya compartidos.
10. Los usuarios acezarán al servidor con perfiles determinados y no de administrador.
11. Los password de administrador y de BD estarán documentadas, estas se modificaran en un periodo de 15 días a un mes.
12. Se tendrá registrado todos los aplicativos instalados en el servidor y sólo se instalaran de acuerdo a las políticas definidas por el administrador.
13. Las caídas de servidor serán anotadas y se analizaran las causas y posibles implicados en dichas caídas.

8. MEJORAS A SERVIDORES

Para el mejor funcionamiento y administración de los mismos se realizaran lo siguiente:

1. Proceso de depuración de directorios.
2. Administración del espacio libre.
3. Repotenciación de algunos servidores (espacio en disco).
4. Limpieza de los Regedit por aplicativo.
5. Desinstalación de componentes anteriores y múltiples de versiones.
6. Desinstalación de aplicativos y software en desuso o no importantes para el dpto.

7. Contar con herramientas de monitoreo de componentes, para así detectar nuestros cuellos de botella y posibles problemas futuros.
8. Eliminación de software de administración remota de servidores.
9. Si es factible formatear el servidor el fin de semana y volverlo a Instalar con las misma configuración y volver a instalar los aplicativos nuevamente.

CAPITULO III: EVALUACION DEL VMWARE

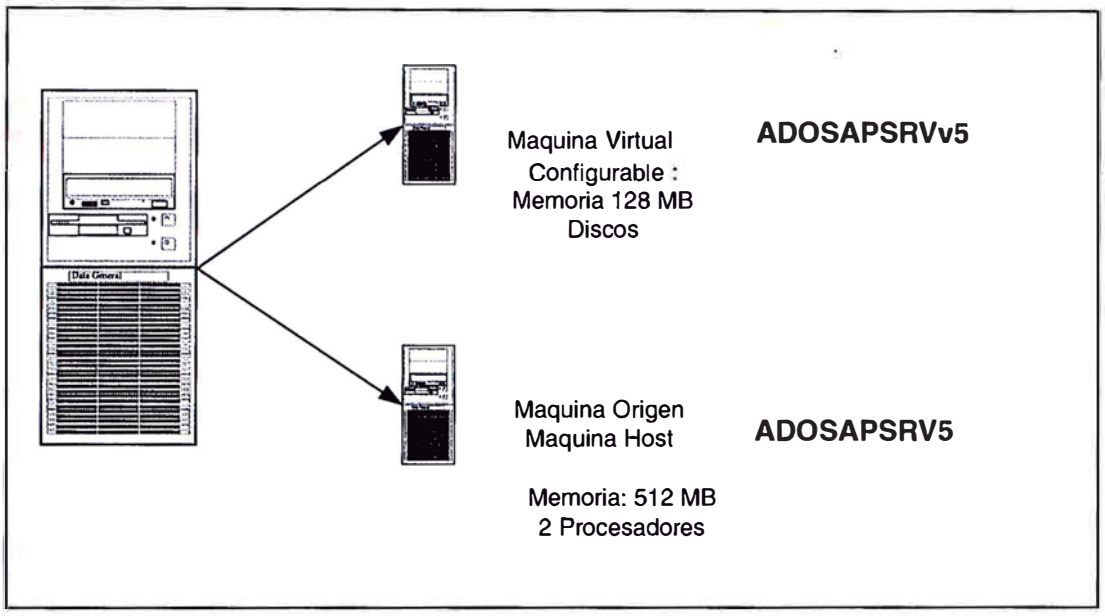
1. GENERALES

Evaluar la herramienta VMWare versión 2.0, para determinar si su uso ayudará a implementar los ambientes necesarios para El Area de Desarrollo de Sistemas, optimizando el uso de los equipos con los que contamos actualmente. Adicionalmente, como consecuencia de lo anterior, se podrían minimizar futuras compras tanto de PC's como de servidores.

2. BREVE DESCRIPCIÓN DE LA HERRAMIENTA

Esta herramienta utilizando la técnica de "virtual machines" nos permite correr **en una sola máquina y al mismo tiempo** (sin reiniciarla), dos, tres o más configuraciones de sistema operativo.

Cada una de las máquinas virtuales se comporta como una máquina física diferente, inclusive a nivel de red, donde todas están en capacidad de compartir archivos, directorios, impresoras, etc. Cabe indicar que se asigna un numero IP diferente a cada maquina virtual que se implemente.


Maquina Virtual
Configurable :
Memoria 128 MB
Discos
ADOSAPSRVv5
Maquina Origen
Maquina Host
ADOSAPSRV5
Memoria: 512 MB
2 Procesadores

Versiones de VmWare.

VMWARE Work Station

Herramienta Productividad para Profesionales Técnicos

VM Software, permite correr multiples SO en un solo computador.

Simplificación de la infraestructura, logrando flexibilidad y seguridad

Capacidad de Redes Avanzadas, T. Red Virtuales, NAT

VMWARE GSX Server

Máquinas Virtuales en Servidores Intel

VM Software para consolidación y particionamiento de servidores

Provee una plataforma segura y estable para distribuir nuevas soluciones, rápida y eficientemente

Facilidad de Integración de ambientes y múltiples servidores, con administración eficiente y reducción de costos.

VMWARE ESX Server

Máquinas Virtuales en Servidores Intel de Alta Performance

VM Software para consolidación y particionamiento de servidores en ambientes corporativos y centro de procedamiento de datos.

Distribución de una plataforma, altamente escalable, segura y uniforme con recursos avanzados de control de la administración.

VMWARE Work Station	VMWARE GSX Server	VMWARE ESX Server
Máquinas virtuales en Desktop (Intel)	Máquinas virtuales en Servidores Intel	Máquinas virtuales en Servidores Intel de Alta Performance
1 CPU	4 CPU	8 CPU
1 GB en RAM	8 GB en RAM	16 GB en RAM

3. CARACTERÍSTICAS PRINCIPALES

a. Sistemas Operativos que soporta

Microsoft MS-DOS	MS-DOS 6
Microsoft Windows	Windows 3.1 Windows for Workgroups Windows 95 Windows 98 y Windows 98 SE
Microsoft Windows NT	Windows NT 4.0 Service Pack 3, 4, 5 y 6a Workstation y Server
Microsoft Windows 2000	Windows 2000 Professional, Server y Advanced Server
FreeBSD	Free BSD 2.2.x y 3.x
Linux	Red Hat 5.x y 6.x Caldera OpenLinux 1.3 y 2.x SuSE Linux 5.3 y 6.x TurboLinux 6.0 Corel Linux OS (Debian 2.2)

b. Manejo de Discos

VMWare maneja dos tipos de discos: Físicos o virtuales en 3 modos

- Persistente (Permanente): el manejo es idéntico al de un disco duro convencional
- No persistente: Este modo nos permite descartar los cambios al finalizar una sesión de VMWare.

- Undoable: Permite al usuario guardar o descartar cambios. Mientras esta decisión se toma, todas las modificaciones se guardan en un redo-log file.

c. Suspender / Restaurar una sesión

Existe una opción para suspender momentáneamente la sesión y luego restaurarla en las mismas condiciones en que se suspendió.

d. Copy / Paste

La utilidad copy / paste está activa y se puede realizar entre las aplicaciones corriendo en una maquina virtual. Adicionalmente la herramienta nos permite realizar esta función entre aplicaciones corriendo en diferentes máquinas virtuales.

4. REQUERIMIENTOS MÍNIMOS (MÁQUINA HOST (PADRE))

Hardware

Procesador	X-86, de 266Mhz como mínimo	
	Intel	Pentium Pro, Celeron, Pentium II, Pentium III
	AMD	K6-2, K6-III, Athlon (K7)
RAM	Mínimo	96 MB
	Recomendable	128 MB

Software

- Windows NT Service Pack 3,4,5 o 6 (Workstation o Server) o Windows 2000 (Profesional o Server)
- Internet Explorer 4.0

5. PRUEBA REALIZADA

A continuación se indican los detalles de la prueba realizada, incluyendo los tipos de prueba y la configuración básica de cada una de las máquinas (reales y virtuales) que se utilizaron.

Máquina original (HOST) BCP_CP4 (servidor)

Características de HW	Memoria: 512 MB
	Disco Duro: 37.2 GB
	Procesador:Pentium III 730 Mhz
Características de SW	Windows NT 4.0 SP4
	SQL 6.5 SP5a
	VMWare 2.0
	Adicionalmente se crearon los ambientes para el aplicativo CAPS, en lo que se refiere a Base de Datos, tablas procedimientos y carga inicial de Datos.

A partir de esta instalación inicial se crearon dos máquinas virtuales con las siguientes características:

Maquina virtual 1 CP_VM1

Características de HW	Memoria: 256 MB
	Disco Duro: 14.8 GB
	Procesador: Pentium III 500 Mhz
Características de SW	Windows NT 4.0 SP3
	SQL 6.5 SP5a

Maquina virtual 2 CP_VM2

Características de HW	Memoria: 128 MB
	Disco Duro: 6 GB
	Procesador: Pentium III 500Mhz
Características de SW	Windows NT 4.0 SP6a
	SQL 7.0 SP1
	Adicionalmente se crearon los ambientes para el aplicativo ICE, en lo que se refiere a Base de Datos, tablas, procedimientos y carga inicial de datos.

Tiempo de Respuesta

Se realizaron validaciones de Tiempo de Respuesta con el aplicativo CAPS, con el aplicativo Letras Hipotecarias, y con el aplicativo ICE, con el apoyo del personal de análisis y de desarrollo, utilizando dos escenarios de prueba (en lo que se refiere a servidores), tal como se detalla a continuación:

CAPS

Servidor real CRED_COMER		Nuevo Servidor real BCP_CP4	
Procesador: Pentium III 400 Mhz		Procesador: Pentium III 730 Mhz	
Memoria: 1GB		Memoria: 512MB	
Espacio asignado a la BD: 4GB		Espacio asignado a la BD: 4GB	
Tiempo de respuesta por transacción		**Tiempo de respuesta por transacción**	
Consulta de cliente	34 seg.	Consulta de cliente	7 seg.
Consulta de SITU	20 seg.	Consulta de SITU	3 seg.
Consulta de resolución	15 seg.	Consulta de resolución	3 seg.
Monitoreo de tarea	30 seg.	Monitoreo de tarea	10 seg.
Detalle del monitoreo	50 seg.	Detalle del monitoreo	20 seg.

Letras Hipotecarias

Servidor real BCP_CP1		Servidor virtual CP_VM1	
Procesador: Pentium III 600 Mhz		Procesador: Pentium III 500 Mhz	
Memoria: 256 MB		Memoria: 256MB	
Espacio asignado a la BD: 309MB		Espacio asignado a la BD: 309MB	
Tiempo de respuesta por transacción		**Tiempo de respuesta por transacción**	
Pagos	2 seg.	Pagos	2 seg.
Pre-Pagos	2 seg.	Pre-Pagos	2 seg.

Consulta de cuentas	1 seg.	Consulta de cuentas	1 seg.
Pase a Judicial	2 seg.	Pase a Judicial	2 seg.
Pase a Extrajudicial	2 seg.	Pase a Extrajudicial	2 seg.

ICE

Servidor real BCP_CP1	**Servidor virtual CP_VM2**		
Procesador: Pentium III 600 Mhz	Procesador: Pentium III 500 Mhz		
Memoria: 256 MB	Memoria: 128 MB		
Espacio asignado a la BD: 60MB	Espacio asignado a la BD: 60MB		
Tiempo de respuesta por transacción	**Tiempo de respuesta por transacción**		
Fiscalización de Cheques	6 seg.	Fiscalización de Cheques	6 seg.
Importación de datos de rechazos	4 min.	Importación de datos de rechazos	3 min,45seg
Visualización de firmas de los cheques	5 seg.	Visualización de firmas de los cheques	5 seg.
Visualización de Reportes	5seg	Visualización de Reportes	4 seg.
Definición de usuario	3seg	Definición de usuario	3 seg.

Tal como se muestra, se obtuvieron resultados satisfactorios, dado que los tiempos obtenidos en los nuevos servidores son menores o iguales a los que se obtuvieron de los servidores originales.

Recursos del Servidor

En los siguientes cuadros se puede observar los índices de utilización de los recursos de los servidores en diferentes escenarios:

Premisas

1. Los datos se tomaron básicamente del performance Monitor del servidor principal BCP_CP4 (Host) y de las máquinas virtuales correspondientes, de acuerdo al escenario.
2. El parámetro evaluado en cada uno de los escenarios es el % de utilización del procesador. Este puede ser real o virtual.
3. Las dos sesiones de máquinas virtuales están activas en todos los escenarios.
4. Las funciones consideradas, son principalmente consultas a Base de Datos

a) Servidores sin ejecutar funciones (host, virtuales)

Perfomance Monitor Máquina Host BCP_CP4	Parámetro Procesador	% de Tiempo
	Promedio (10 minutos)	8.07%
	Máximo	34.5%

a) Servidor ejecutando funciones solo en la máquina Host

Perfomance Monitor Máquina Host BCP_CP4	Parámetro Procesador	% de Tiempo
	Promedio (10 minutos)	9.87%
	Máximo	37.5%

a) Servidor ejecutando funciones solo en la máquina virtual 1

Perfomance Monitor Máquina Host BCP_CP4	Parámetro Procesador	% de Tiempo
	Promedio (en 10 minutos)	12.3%
	Máximo	64.5%
Perfomance Monitor Máquina Virtual BCP_VM1	**Parámetro Procesador**	**% de Tiempo**
	Promedio (en 10 minutos)	9.96%
	Máximo	74%

a) Servidor ejecutando funciones en la máquina Host y en la máquina virtual 1

Perfomance Monitor Máquina Host BCP_CP4	Parámetro Procesador	% de Tiempo
	Promedio (10 minutos)	14.03%
	Máximo	51.5%

Perfomance Monitor Máquina Virtual BCP_VM1	Parámetro Procesador	% de Tiempo
	Promedio (en 10 minutos)	9.36%
	Máximo	61%

a) Servidor ejecutando funciones en la máquina Host y en las 2 máquinas virtuales

Perfomance Monitor Máquina Host BCP_CP4	Parámetro Procesador	% de Tiempo
	Promedio (en 10 minutos)	15.893%
	Máximo	76.1%
Perfomance Monitor Máquina Virtual BCP_VM1	**Parámetro Procesador**	**% de Tiempo**
	Promedio (en 10 minutos)	9.91%
	Máximo	75.7%
Perfomance Monitor Máquina Virtual BCP_VM2	**Parámetro Procesador**	**% de Tiempo**
	Promedio (en 10 minutos)	9.89%
	Máximo	73.5%

6. CONCLUSIONES DE LAS PRUEBAS

1. Después de realizadas las pruebas, podemos afirmar que esta herramienta nos permite contar con servidores virtuales capaces de soportar las tareas de los diferentes aplicativos en forma normal (como si se tratara de un servidor físico diferente).

2. Se comprobó que es posible manejar diferentes versiones de Sistemas Operativos en la máquina Host y en las máquinas virtuales. Específicamente nos referimos a todas las versiones de Service Packs que requieren los aplicativos del Banco.

3. La opción de crear discos Undoable, nos será de mucha utilidad en el caso de que necesitemos conservar baselines (Permite crear plantillas de Configuraciones de Sw Base).

4. Si por cualquier motivo, alguna de las máquinas virtuales colapsa, la máquina Host y las otras máquinas virtuales no se ven afectadas.

5. De acuerdo a los resultados obtenidos en las pruebas con un servidor con las siguientes características:
   Procesador: Pentium III 730 Mhz
   Memoria: 512 MB
   Se observó que el rendimiento del server comenzó a degradarse al instalar una tercera máquina virtual, específicamente en los que se refiere a memoria.

7. RECOMENDACIONES PARA EL USO DEL VMWARE

1. Es deseable que la máquina Host cuente con los recursos de HW suficientes para que no se degrade el rendimiento normal, tanto de

la máquina Host como de las máquinas virtuales. Esto depende mucho de la cantidad de máquinas virtuales que se implementen en un Host. Por ejemplo:

Se recomienda que para un servidor con las siguientes características:

Procesador: Pentium III 730 Mhz

Memoria: 512 MB

Se considere la instalación máxima de 2 máquinas virtuales. En el caso de que se requiera la implementación de mas máquinas virtuales, la máquina host, deberá contar con mayores recursos de Hardware.

2. En el caso que se decida la compra de la herramienta, seria deseable que además de considerar a los servidores, también se incluya a las PC's clientes. Esto facilitaría las pruebas en los casos que se requiere probar un mismo aplicativo en diferentes plataformas.

   Cabe indicar que la herramienta solo puede instalar el Sistema Operativo desde una unidad de CD local, lo que significa un gran inconveniente, dado que las máquinas del servicio no cuentan con este dispositivo.

8. ESTANDARES A USAR

Toda máquina Virtual que se cree llevará por nombre de referencia el nombre de la maquina origen seguido de una v indicando que número de máquina virtual (nombre máquina origen **v# de maquina virtual**) es:

Ejemplo. Maquina Origen BCP_ACS

Maquina Virtual BCP_ACSv1

CAPITULO IV: CONSOLIDACION DE SERVIDORES

1. DESCRIPCIÓN GENERAL

La consolidación de servidores supone la optimización de los recursos para incrementar la productividad de los departamentos y reducir las necesidades de personal, reduciendo los costos totales. Situar los sistemas en una única localización permite a los departamentos técnicos responder de un modo efectivo a los desafíos que surgen hoy en la empresas simplificando la gestión de los datos, reduciendo las necesidades de espacio y ayudando a controlar los altos costos de explotación.

El principal beneficio de la consolidación supone la reducción de costos y el incremento de la fiabilidad del acceso a los datos y a los recursos de computación. La consolidación de servidores es una tendencia de la industria que supone la optimización de los recursos físicos, la consolidación de las aplicaciones en un menor número de servidores más potentes y la gestión centralizada de las aplicaciones críticas de negocio.

Hoy en día las empresas están demandando ambientes que ofrezcan altos niveles de rendimiento, confiabilidad, disponibilidad y crecimiento. Gran cantidad de compañías para cumplir con el creciente desarrollo de su infraestructura tecnológica, añaden servidores individuales para soportar aplicaciones específicas, sin embargo al instalar un gran número de servidores de función sencilla (archivos, correo, impresión, etc.) crean una pesadilla en la administración y soporte de la red,

además de no resistir el crecimiento y demandas de usuarios en forma ágil.

La consolidación de servidores es la solución más efectiva para esta problemática, en donde la empresa combina procesos comunes y reemplaza el gran número de pequeños servidores por uno de gran rendimiento y confiabilidad.

La consolidación de servidores puede ayudar a reducir los siguientes costos:

a) Productividad del personal

La gestión centralizada de los servidores libera recursos técnicos, mejorando los tiempos de respuesta y disminuyendo los tiempos de parada del servidor.

b) Control de Datos y seguridad

La consolidación de servidores proporciona un acceso más conveniente y fiable.

c) Costos de Hardware

La estandarización de los equipos lleva a economías de escala de modo de mantener un menor número de servidores reduce los costos de hardware.

d) Cuotas por Licencia de Software

Reducir el número de servidores necesarios para dar soporte a los clientes significa un menor número de licencias por servidor.

2. VENTAJAS DE LA CONSOLIDACIÓN DE SERVIDORES

Bajos costos de operación:

El 80% del costo total de propiedad (TCO) lo representan los costos de operación. A través de la consolidación de servidores estos se ven reducidos debido a que los problemas de administración, costos por licenciamiento de software y gastos por mantenimiento de hardware disminuyen ya que la complejidad de la red es menor al tener un solo servidor consolidado.

Menor manejo de recursos:

Mediante la reducción significativa de servidores existe un excelente manejo de recursos, tanto humanos como de infraestructura de sistemas, debido a que generalmente las organizaciones poseen una gran cantidad de servidores de diferentes edades, fabricantes y sistemas operativos, que además implica una gran cantidad de personas que deben brindarle soporte. Mediante la consolidación de servidores la mayoría del personal perteneciente al departamento de sistemas pueden realizar funciones de alto nivel en vez de labores de mantenimiento y soporte, al disminuirse la complejidad de la red.

Retorno de la inversión:

En este esquema las compañías tienen grandes ahorros, permitiendo el retorno de la inversión por consolidación de servidores, a mediano plazo.

Ahorro de espacio físico:

Mediante la consolidación de servidores, se obtiene mayor espacio físico disponible, debido a que múltiples servidores son llevados a uno solo a pocos, ocupando una sola área.

3. TIPOS DE CONSOLIDACIÓN DE SERVIDORES

Lógica: Se mantiene el número y emplazamiento de los servidores pero se centraliza su gestión.

- Aumento de la productividad personal mediante la centralización y monitorización de tareas
- Reducción del coste total de propiedad (TCO)

Física: Transición desde los servidores distribuidos ampliamente a localizaciones más centralizadas o centros de datos.

- Se incrementa la seguridad física al limitar el
- número de ubicaciones y accesos.
- La productividad se incrementa al centralizar las tareas.
- Se reduce el coste total de propiedad (TCO)
- El rendimiento de la red puede ser mejorado
- El espacio físico se usa más eficientemente.

Racional: La reducción del número total de servidores. Los servidores más viejos, lentos y caros se reemplazan por menor número de servidores, más potentes y más baratos.

- Se eliminan servidores pequeños y de diferentes plataformas consolidando en servidores estándares de mayor potencia
- La productividad del personal de soporte se incremente con la estandarización de servidores y aplicaciones evitando la formación en múltiples plataformas
- Los entornos heterogéneos implican complicaciones por la convivencia de distintos

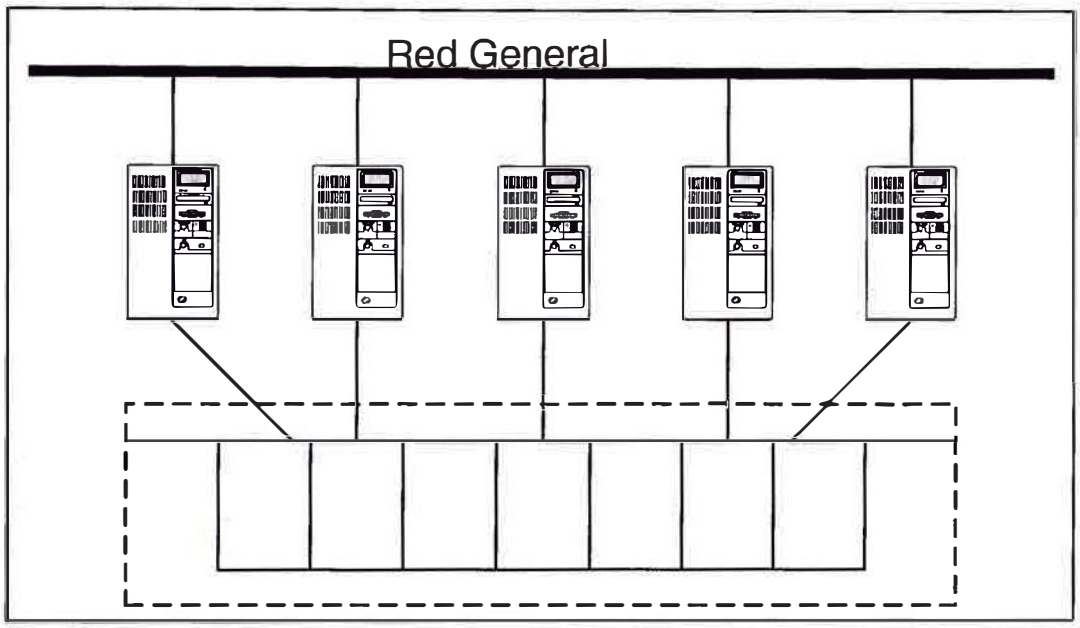
sistemas operativos

4. CONSOLIDACION DEL ALMACEAMIENTO

Las organizaciones que se pueden beneficiar de la consolidación de almacenamiento a menudo tienen un crecimiento de la capacidad de almacenamiento para cada servidor de la red rápido y dedicado. Tanto si los servidores están centralizados como distribuidos, los volúmenes de cintas y discos dedicados disuaden la compartición de datos, complicando la seguridad de la información, haciendo más difícil los backups del administrador e incrementando en gran medida el costo y la complejidad del crecimiento del conjunto de equipos de almacenamiento.

La consolidación del almacenamiento proporciona pools de lata disponibilidad de un modo flexible y con la gestión centralizada de almacenamiento, se puede distribuir para proporcionar el rendimiento y la disponibilidad demandados por las aplicaciones. Además, la consolidación del almacenamiento permite gestionar mejor el crecimiento, controlar la seguridad y el acceso a la información y proporciona una respuesta rápida a los cambios en las necesidades del negocio.

La consolidación del almacenamiento permite a múltiples servidores acceder a un repositorio de almacenamiento compartido.



5. BENEFICIOS DE LA CONSOLIDACIÓN DEL ALMACENAMIENTO

La consolidación del almacenamiento proporciona numerosos beneficios a las organizaciones, incluyendo:

a) Alta escalabilidad en el almacenamiento. Permite a los administradores gestionar el crecimiento y la respuesta rápida a las necesidades de cambio en le negocio.

b) Alta disponibilidad y almacenamiento tolerante a fallos Proporcionado acceso continuo y fiable a los datos.

c) Mejora de la gestión de los datos y de la protección del almacenamiento consolidado.

d) Reducción de los costos Administrativos y del tiempo requerido en la resolución de los problemas.

e) Independencia de la Plataforma Permite la compartición de datos y la simplificación de los procedimientos de backup.

6. IMPLEMENTACIÓN

De acuerdo a la naturaleza de la empresa, tomando en cuenta el tamaño y la complejidad de los aplicativos que posee, se ha visto necesario realizar la Implementación de una consolidación de servidores de tipo física primeramente y posteriormente racional. Estos definición de consolidación aparecieron cuando ya se había iniciado el proceso de administración centralizada de servidores y lo único que se realizo es adaptarlo a lo que mundialmente se conocía, pero nos dimos cuenta que muchas empresas al igual que nosotros sufrían el hecho de tener muchos procesos distribuidos y que eran uno de los síntomas del C/S (por el crecimiento acelerado), y esta solución era una manera de poder mejorar la administración de servidores.

En la empresa se penso primeramente centralizar los servidores en un lugar (consolidación física), bajo unos administradores los cuales velaban por el acceso y mantenimiento de los mismos, posteriormente se realizo un plan el cual se describe en los párrafos siguientes.

- Adquirir 8 servidores Compaq (***ProLiant DL580 G2***) con las siguientes Carácterísticas:

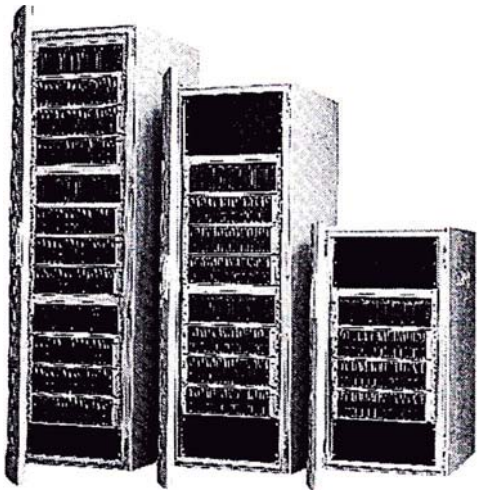
Servidores para Máquinas Virtuales (VMWare GSX 2.0)	
	Ctdad.
Servidor ProLiant DL580 G2 (Hasta 4 Procesadores)	**8**
Configuración de cada equipo:	
DL580G2 X1400-512 2P 2GB, RPS, BBWC US	1
1.4GHz 512 processor	2
2GB (4x512MB)	2
18.2-GB Pluggable Ultra3 SCSI 10,000 rpm Universal Hard Drive (1²)	2
Compaq NC3134 Fast Ethernet NIC 64 PCI Dual Port 10/100	1
Compaq NC3135 Fast Ethernet Module Dual 10/100	1

- Adquirir 1 Rack (**HP RACK 42U SERIES 10000**)

CARACTERISTICA	DESCRIPCION
Factor de Dimensión	42U de altura y 19 pulgadas de ancho
Color	Negro mate
Unidad de distribución de poder	Dos (2) Power Distribution Unit – High Voltage con regleta de tomas para todos los equipos
Bandejas	Bandeja para Teclado y Mouse. Bandeja para Monitor Estándar de 17"
Switch	Incluye Server Console Switch 1 x 8 Puertos KVM.
Teclado y Mouse	Teclado y Mouse por rack
Monitor	Cada rack incluye una pantalla de 15" (220v, 60Hz) que cumple con la norma "Energy Star" y que esta integrado al teclado y mouse (1U)
Accesorios	Incluye todos los accesorios de interconexión necesarios para instalar y administrar hasta 8 servidores dentro del rack.

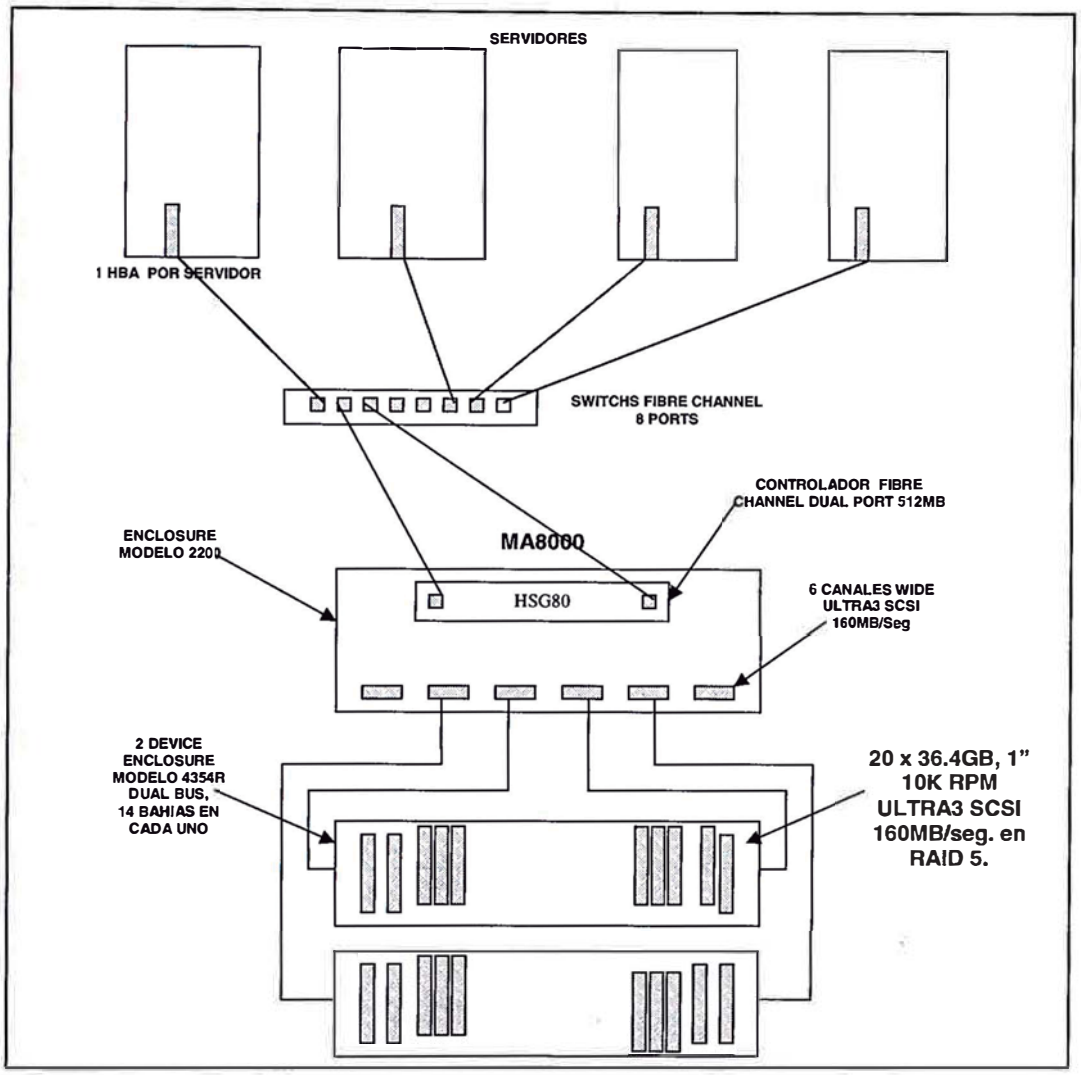
- Adquirir un Storage Area Network (SAN).
  MA 8000 soporta múltiples plataformas como:

  - Tru64 UNIX
  - Open VMS.
  - Windows NT
  - Sun Solaris.
  - HP-UX
  - Novell/Netware
  - IBM AIX.
  - SGI IRIS
  - Linux .



Solución de Almacenamiento Centralizada	Ctdad.
StorageWorks MA8000 (Incluye 720GB)	1
CONTROLADOR	
M2200 6-Channel Controller Shelf	1
StorageWorks HSG80 solution software v8.7 for Windows NT/2000	1
HSG80 Controller without ECB Cable Kit	1
256 MB Cache Upgrade for Fibre Channel or SCSI EMA/MA/RA/ESA	1
StorageWorks Enclosure M2200/M2100 ECB Battery Option	1
StorageWorks array controller software v8.7F for FC/AL & SF	1
BANDEJAS DE DISCOS	
StorageWorks Enclosure Model 4354R, Rackmount	2
1 m Ultra SCSI Cable	4
36.4 GB Pluggable Ultra3 SCSI 10,000 rpm Universal Hard Drive (1)	20
COMPONENTES STORAGE AREA NETWORK (SAN)	
2 m Multi-Mode LC/SC Fibre Channel Cable	4
2 GB Small Form Pluggable SW Transceiver Kit (2Gbit Switches)	8
StorageWorks™ SAN Switch 2/8-EL	1

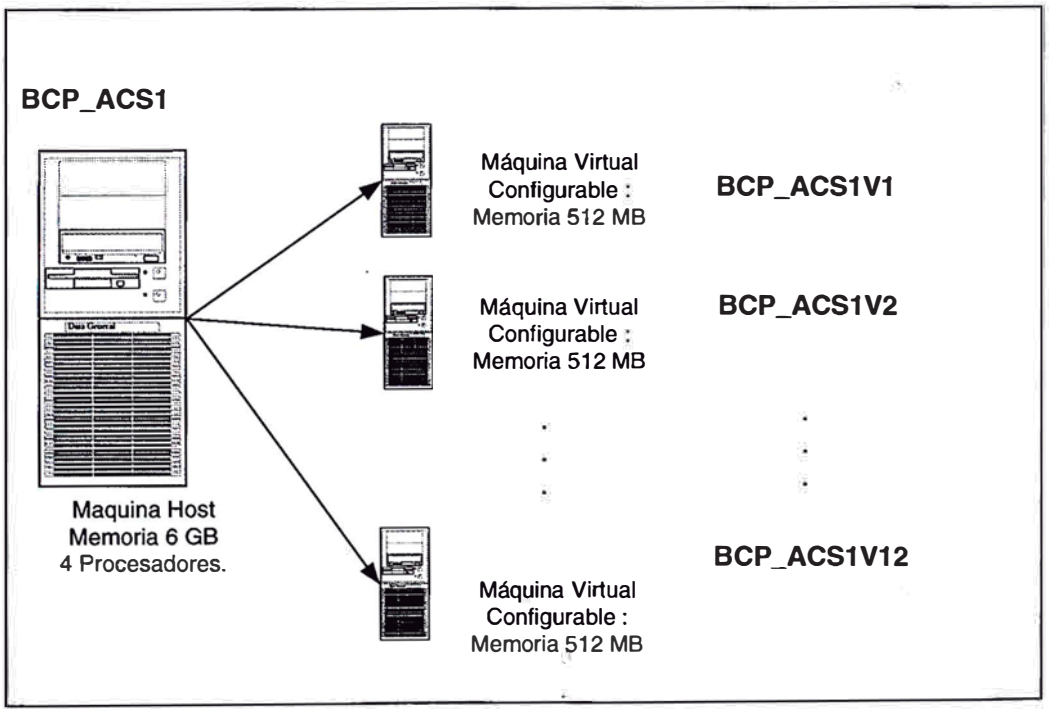
- La solución a Implementar es la siguiente:



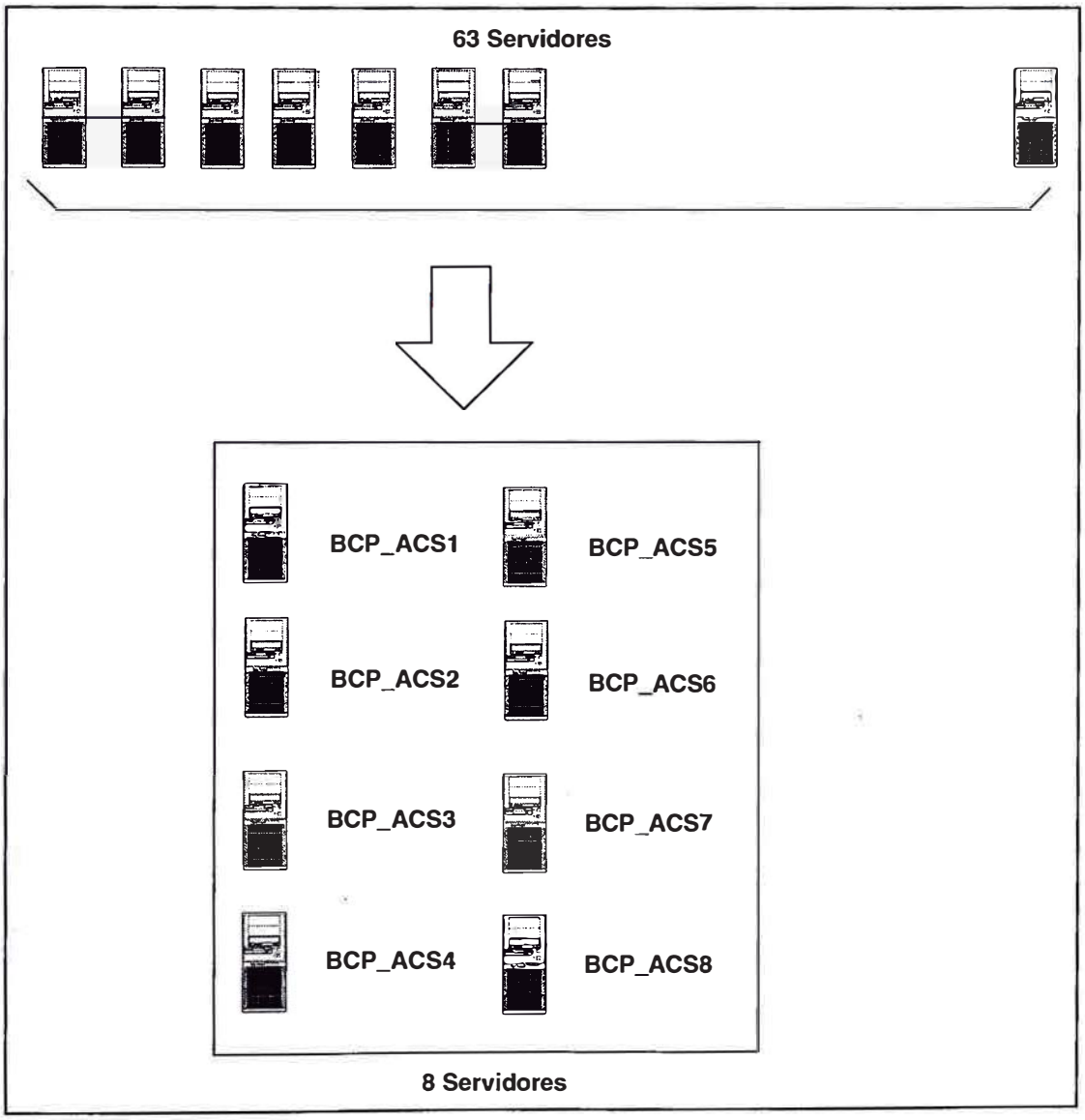
Con el apoyo del VmWare podemos obtener los ambientes que actualmente tenemos para lo cual de uno de los servidor quese van a adquirir, con las características definidas líneas arriba, podemos obtener los siguientes servidores (servidores virtuales):

0.5 GB RAM (hasta: IBM-> 8Gb HP->32 Gb)
VMWARE GSX
2 Discos 18 Gb RAID 1 (Storage)

De acuerdo alas pruebas realizadas de performance y de acuerdo a las necesidades propias de cada aplicativos se ha creído conveniente que de cada servidor físico se obtengan 12 máquinas virtuales, para lo cual se adquirido el VmWare GSX por la cantidad de memoria que posee cada servidor físico (Máquina Host), pues el WorkSatation no puede administrar dicha memoria (Tal como se definió en el parte IV).



Con esta configuración y con el número de servidores que se adquirirán obtendremos un total de 96 servidores pero la tendencia es a disminuir el número de servidores, intentando consolidar para ello los aplicativos que es una nueva etapa de este proceso. De acuerdo a lo planificado Reemplazaríamos 63 servidores Físicos por 8, tal como se muestra en la figura.



CAPITULO IV: EVALUACIÓN DE CCC/HARVEST

1. DESCRIPCION GENERAL

Es innegable que el mundo empresarial se encuentra sujeto a permanentes cambios originados por nuevos objetivos y metas. Para afrontar estos cambios con éxito es necesario que las aplicaciones tecnológicas que soportan gran parte del negocio se encuentren estandarizadas y debidamente organizadas de manera que estos cambios no representen un cuello de botella en el logro de los objetivos y metas trazadas. Ante esto surge la necesidad de contar con una Solución de Administración y Control de cambios que permita administrar y realizar los cambios de manera rápida y eficiente.

La solución para administrar y realizar estos cambios rápida y eficientemente es CCC/Harvest, herramienta que, adecuadamente implementada, es capaz de reducir los costos totales de las actividades de coordinación en el desarrollo de aplicaciones distribuidas hasta en un 30%, como lo demuestra un estudio independiente preparado por el Software Engineering Institute de la Universidad de Carneggie Mellon.

La reducción de tiempo en los procesos de desarrollo se obtiene gracias a que las tareas realizadas a lo largo del ciclo de vida de desarrollo se encuentran claramente definidas minimizando así la duplicidad de esfuerzo. De esta manera se busca encuadrar el desarrollo de aplicaciones en un marco ordenado y disciplinado que permita obtener, entre otros beneficios, una mejor distribución de

funciones de desarrollo, menor cantidad de errores puestos en producción, mantener un histórico de la evolución de las aplicaciones, reutilización de código fuente, métodos más eficientes en el control de calidad, entre otros.

El incremento en la seguridad de las aplicaciones se ve reflejado a través del almacenamiento del código fuente en un repositorio al cual solo se puede acceder si el perfil definido dentro del ciclo de vida de desarrollo así lo requiere y si existe una orden de trabajo que permita la manipulación del código fuente. Toda actividad realizada a través de CCC/Harvest queda registrada facilitando el trabajo en la auditoria de sistemas.

Para el caso de la Empresa el CCC/Harvest integra los procesos realizados a través de diferentes áreas involucradas (Desarrollo, Control de Calidad y Producción) en la atención de requerimientos de cambios de software gracias a un único y estandarizado workflow.

La empresa realiza en algunos Departamentos para el proceso de almacenamiento y Control de Fuentes usa el Visual SourceSafe, el cual tiene una serie de desventajas en comparación con el CCC/Harvest pero ahora se ve la necesidad de poder implementar está herramienta en toda el área de Desarrollo de Sistemas pues no existe una buena administración de fuentes, los cuales se encuentran dispersos en diferentes sitios y existe a veces perdida de estos fuentes por las diversas manos por las que pasa. Se cree conveniente evaluar está herramienta y compararalo con lo que actualmente usamos y tratar de buscar la mejor forma de implementarlo.

2. METAS

Los metas del piloto realizado son las siguientes:

1. Mostrar el impacto positivo de la aplicación de las funcionalidades de CCC/Harvest en las actividades del área de desarrollo de aplicaciones distribuidas de la Empresa.

2. Extender el diseño del flujo de trabajo actual, que cubre únicamente al área de Control de Calidad, hacia el área conexa de Desarrollo por medio de un diseño básico preliminar. Este diseño será posteriormente mejorado, consolidado y documentado como parte del proyecto de Implementación a ser propuesto.

Durante el piloto se demostró que estos objetivos son alcanzables mediante el desarrollo de una solución que integra estrategia, procesos, personal y tecnología basada en el uso de CCC/Harvest, sin embargo cabe mencionar que el apoyo prestado por las diferentes gerencias y personal involucrados han sido determinantes para el éxito del piloto.

3. ENFOQUE

El piloto consistió en la Implementación de una solución de *Control de Configuración y Cambios para el Ciclo de Vida del Desarrollo de Aplicaciones Distribuidas* y estuvo enfocado en cubrir las siguientes actividades de la Empresa:

a) Coordinación de las actividades propias de la atención de requerimientos de cambio (desarrollo o mantenimiento de aplicaciones) tales como la asignación de trabajos, análisis y diseño, desarrollo, pruebas funcionales, control de calidad, seguridad de información y puesta en producción.
b) Almacenamiento y administración de accesos a los archivos fuentes y demás archivos asociados al trabajo de desarrollo de las aplicaciones distribuidas.
c) Administración de la efectividad del trabajo de desarrollo y mantenimiento realizado por los programadores.
d) Administración de la configuración de las versiones de los fuentes que conforman una versión de una aplicación.
e) Establecimiento de una metodología de trabajo en el desarrollo de aplicaciones distribuidas basadas en el uso de CCC/Harvest.

4. BENEFICIOS

1. **Incremento de la Productividad y Reducción de Costos**

   Según el Capability Maturity Model (CMM) preparado por el Software Engineering Institute de la Universidad de Carnegie Mellon a través de la implementación de una solución de Administración y Control de Cambios, en este caso basada en CCC/Harvest, se obtendría un ahorro de un 30% en el costo de las siguientes actividades.

   a. **Acceso y Recuperación de Fuentes**

      Disminución en el tiempo necesario para determinar la versión de un componente de software a modificar. Típicamente las actividades de acceso y recuperación de software pueden consumir hasta un **3%** del costo del proyecto.

   b. **Coordinación a través del ciclo de vida de desarrollo**

      Disminución de las actividades de coordinación en desarrollos paralelos o concurrentes asegurando que los cambios no se sobrescriban y manteniendo la integridad de la aplicación. Típicamente estas actividades pueden llegar a consumir hasta un **10%** del costo del proyecto.

   c. **Modificación de Cambios a través del ciclo de vida de desarrollo.**

      Los cambios se mueven en sentido contrario en el ciclo de vida cuando después de ser realizadas las pruebas deben regresar a la etapa de desarrollo. Típicamente estas actividades pueden llegar a consumir hasta un **6%** del costo del proyecto

   d. **Administración de la Configuración de componentes de una aplicación**

      Incluye todos los esfuerzos requeridos para asegurar que la aplicación está completa para la generación del medio

instalador o la transferencia electrónica y la habilidad para reconocer la versión de la aplicación que está siendo utilizada por un grupo de usuarios específicos. Típicamente estas actividades pueden llegar a consumir hasta un **3%** del costo del proyecto.

e. **Obtención de Aprobaciones y Notificaciones**

En muchos proyectos las aprobaciones y/o notificaciones de conclusión de tareas son requeridas antes de que puedan continuarse con otras actividades tales como migración, distribución, etc. Localizar a los individuos apropiados de manera oportuna puede convertirse en un cuello de botella. Típicamente estas actividades pueden llegar a consumir hasta un **2%** del costo del proyecto.

f. **Administración de Requerimientos de Cambio**

La administración manual de los requerimientos de cambio es una actividad engorrosa y consumidora de tiempo. Dentro de la información típica administrada se encuentra: Desarrollador/Probador asignado al requerimiento, Fecha de Inicio, Fecha de Finalización, Requerimientos de Cambio asociados, etc. Típicamente estas actividades pueden llegar a consumir hasta un **2%** del costo del proyecto.

g. **Coordinación y Comunicación entre grupos**

En la medida que los equipos de desarrollo se vuelven más distribuidos a través de la organización, y frecuentemente geográficamente remotos, la comunicación y coordinación efectiva entre ellos se vuelve crítica. Típicamente estas actividades pueden llegar a consumir hasta un **4%** del costo del proyecto.

h. **Obtención del Estado de Avance del Proyecto**

En la medida que los ambientes de desarrollo se hacen más complejos, obtener información confiable y precisa del estado del proyecto se vuelve difícil y demandante de tiempo.

Típicamente estas actividades pueden llegar a consumir hasta un **3%** del costo del proyecto.

i. **Rastreo de Errores y Reparaciones**

Este esfuerzo usualmente involucra a grupo de usuarios diferentes a los desarrolladores. Conforme le número de cambios crece el esfuerzo en rastrear errores, reparaciones y solicitudes de cambio se vuelve enorme. Típicamente estas actividades pueden llegar a consumir hasta un **2%** del costo del proyecto.

Las actividades anteriormente mencionadas llegan a consumir hasta un 35% del costo del proyecto.

2. **Incremento de la Seguridad**

A través del almacenamiento de fuentes en un repositorio único y centralizado y del establecimiento de nuevas políticas de seguridad. De esta manera aseguramos que los estándares establecidos por Auditoria de Sistemas sean cumplidos.

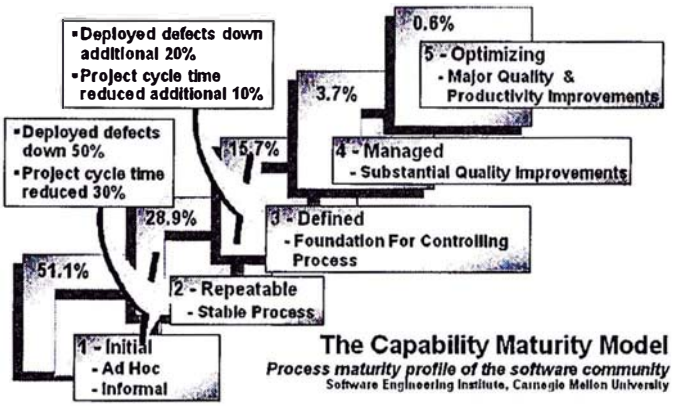
3. **Disponibilidad de la funcionalidad de CCC/Harvest a través de Web.**

Esto permitirá a los equipos de desarrollo distribuidos a través de la organización y geográficamente remotos acceder a la funcionalidad de CCC/Harvest en el mismo instante que requieran interactuar con él para funciones propias de su trabajo (Desarrollo, Control de Calidad, Auditoria, etc.).

4. **Direccionamiento de la Empresa hacia CMM (Capability Maturity Model).**

El establecimiento de una metodología de Administración y Control de Cambios implica un salto desde un primer nivel de desarrollo informal hacia un segundo nivel de desarrollo de procesos repetibles. Este salto entre niveles definidos dentro de

CMM le permitirá a la Empresa disminuir los defectos de desarrollo hasta en un 50% y disminuir el tiempo en el ciclo de vida de desarrollo hasta en un 30%.



5. PLAN DE TRABAJO DESARROLLADO

Fase I: Reconocimiento y Documentación de los Procesos de Desarrollo

Toda organización de TI es un universo particular y como tal desarrolla sus aplicaciones de forma única y distintiva. No existen empresas, por más que se dediquen al mismo negocio, que operen de similar modo. Por ello es imprescindible, para la consecución de los objetivos trazados, el reconocimiento, la documentación y posterior validación de los procesos con los responsables de los mismos.

Tareas:

- ✓ Entrevistas al personal de gerencia y jefatura para levantamiento de información a nivel gerencial.
- ✓ Entrevistas al personal operativo para levantamiento de información a nivel de detalle.
- ✓ Construcción del modelo de procesos actual.

- ✓ Validación del modelo de procesos con el personal de desarrollo de sistemas.
- ✓ Refinamiento y aprobación del modelo final.

Fase II: Diseño del Ambiente de Trabajo de la Solución de Control de Configuración y Cambios Para el Ciclo de Vida de Desarrollo de Aplicaciones Distribuidas

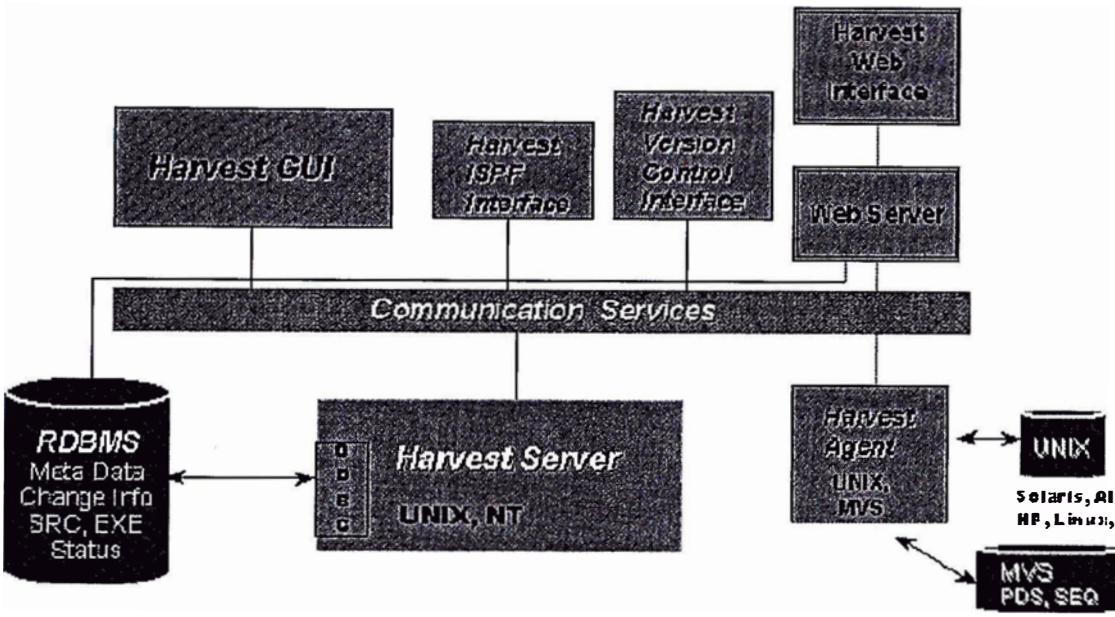
En esta fase, se cubre tanto la definición como el diseño de una serie de elementos que conformarán la solución de Control de Configuración y Cambios.

Tareas:

- ✓ Identificación de las etapas del ciclo de vida, sus procesos y actividades.
- ✓ Definición de los usuarios iniciales del piloto.
- ✓ Definición de los grupos de usuarios y su jerarquía dentro del piloto
- ✓ Definición de las solicitudes de cambio y los formularios asociados.
- ✓ Identificación y organización lógica de los fuentes y demás archivos asociados a cada proyecto de desarrollo de aplicaciones distribuidas.
- ✓ Definición de una nomenclatura estándar para los grupos de usuarios, repositorios y proyectos.

6. ARQUITECTURA DE CCC/HARVEST

Esquema de la solución de Control de Configuración y Cambios para el Ciclo de Vida del Desarrollo de Aplicaciones Distribuidas



7. INFRAESTRUCTURA

Requerimientos de Hardware

- Por cada Usuario
  - Computador IBM-Compatible con procesador Intel Pentium o estación de trabajo para IBM AIX, HP-UX, Sun Solaris, Digital UNIX.
  - Por lo menos 32 Mb de RAM, 64 recomendado.
  - Por lo menos 50 Mb de espacio en disco duro para la instalación de software cliente del CCC/Harvest.
  - Espacio en disco suficiente para la carga de los fuentes y demás archivos con los que trabaja frecuentemente el usuario en su proyecto de desarrollo.
- Para el Computador Servidor

- Computador IBM-Compatible con procesador Pentium-based de 500Mhz para Windows NT/2000 o servidor para HP-UX, Solaris, IBM AIX.
- 512 MB RAM dedicado para la aplicación.
- 15 GB espacio libre en disco mínimo (el crecimiento de la capacidad requerida dependerá del volumen de fuentes y documentos adicionales a ser almacenados*).

* Actualmente CCAL cuenta con 10 GB de información en los repositorios de CCC/Harvest.

- Para copias de seguridad de la información, se recomienda el uso de alguna de las diversas herramientas disponibles para realizar Backups.

Requerimientos de Software:

- Por cada Usuario
  - Conexión de red a un servidor basado en Unix o Windows NT usando el protocolo TCP/IP.
  - Servicios TCP/IP instalados y habilitados.
  - Sistema operativo Windows de 32 bits.
  - Microsoft Internet Explorer 5.0 o superior.
- Para el Computador Servidor
  - Plataforma Operativa: Windows NT Server 4.0 SP 5+ / Windows 2000 Advanced Server, con servicios TCP/IP instalados y habilitados
  - Licencias de acceso para los usuarios a un servidor de base de datos Oracle Standard Edition 8.x o Enterprise Edition 8.x.

8. DOCUMENTO DE DISEÑO DEL AMBIENTE DE TRABAJO DE LA SOLUCIÓN DE CONTROL DE CAMBIOS

REPOSITORIOS

Los repositorios se definieron siguiendo la nomenclatura actual adoptada por Control de Calidad (CCAL).

Ejm:

P	E	SistemaIntegradoBancaSeguros

Donde:

P	E	- Prefijo País

SistemaIntegradoBancaSeguros	- Nombre de la Aplicación

Los Repositorios definidos para el piloto son los siguientes:

PESistemaIntegradoBancaSeguros
PESistema
PECounselor

GRUPO DE USUARIOS

Los Grupo de Usuarios se definieron siguiendo la nomenclatura actual adoptada por CCAL.

Ej.:

Gestión Comercial	- Jefe de Proyecto

Donde:

Gestión Comercial	- Departamento de ADS

- Jefe de Proyecto	- Perfil de Usuario

Los Grupos de Usuarios definidos para el piloto son los siguientes:

Gestión Comercial – Jefe de Proyecto

Gestión Comercial – Analista Técnico

Gestión Comercial – Desarrollador

Control de Cambios

Supervisor Procesos Distribuidos

Probador

Supervisor Pruebas

Segurinf

Servicorp

Ingeniería de Sistemas

USUARIOS

Los Usuarios definidos para el piloto son los mismos definidos por la Empresa.

PROYECTOS

Los Proyectos se definieron siguiendo la nomenclatura actual adoptada por CCAL. Adicionalmente incluye la versión de dicha aplicación lo cual permitirá realizar administración de configuraciones.

Ejm:

P	E	SistemaIntegradoBancaSeguros	v. 1.0

Donde:

P	E

- Prefijo País

SistemaIntegradoBancaSeguros

- Nombre de la Aplicación

v. 1.0

Versión de la Aplicación

Los Proyectos definidos para el piloto son los siguientes:

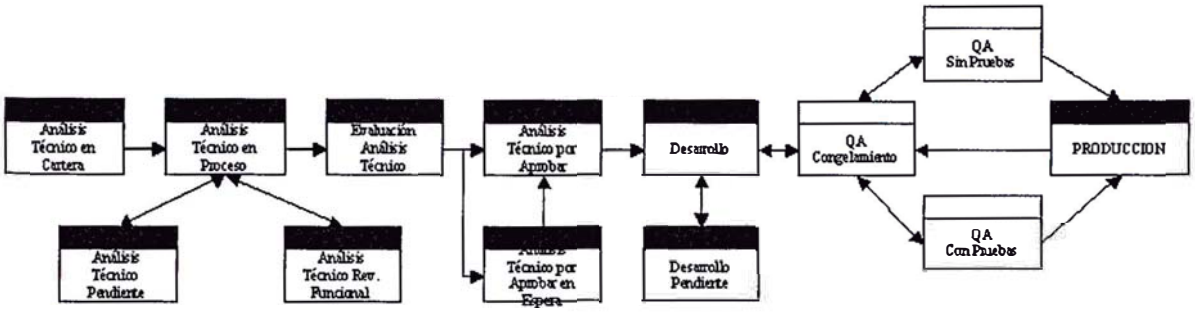
PESistemaIntegradoBancaSeguros v. 1.0
PESistema v. 1.0
PECounselor v. 1.0

Los Proyectos cuentan con un conjunto de etapas las cuales definen el workflow por el cual transitan los requerimientos de cambios. Cada etapa cuenta con procesos los cuales solo pueden ser ejecutados por el perfil asignado, de esta manera se cumplen con las políticas de desarrollo de la Empresa.

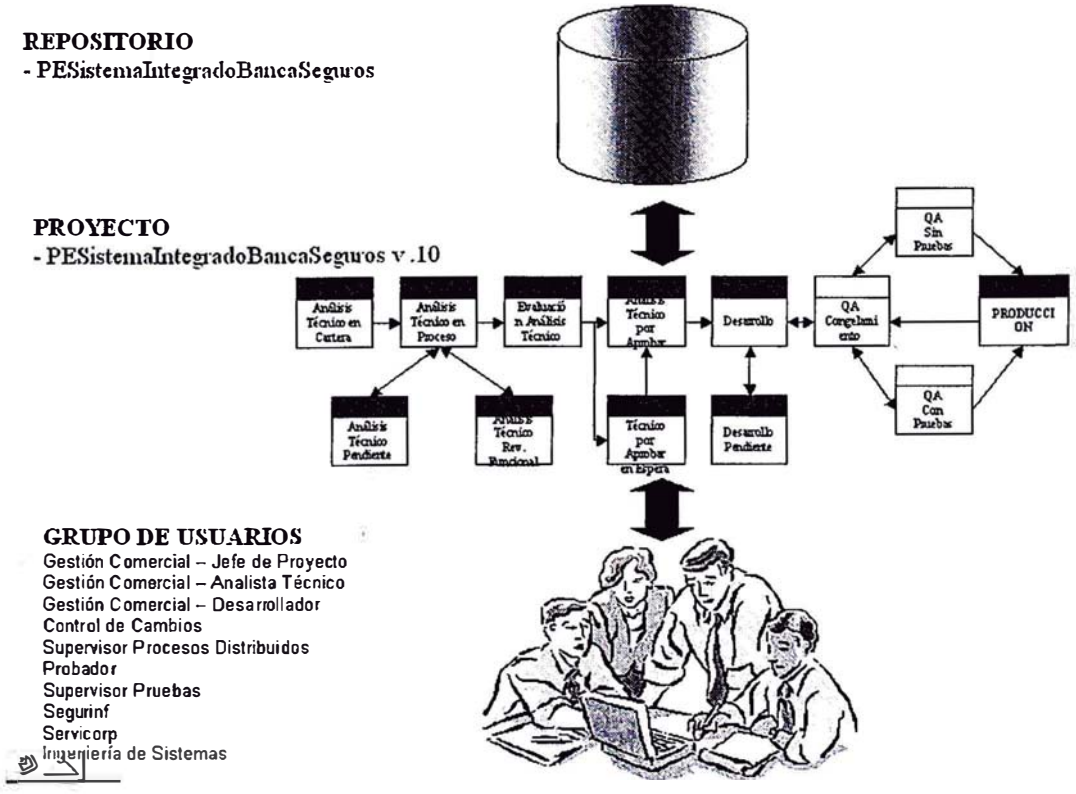
Para el caso del piloto se ha definido el siguiente workflow, según la información brindada por el personal de la Empresa, el cual es 100% personalizable. Este workflow involucra a las áreas de Desarrollo, Control de Calidad y Producción.

Ej.:

PESistemaIntegradoBancaSeguros v. 1.0



La relación que existe entre el Repositorio, el Proyecto y el Grupo de Usuario es la siguiente:



MATRIZ DEL PROYECTO PESistemaIntegradoBancaSeguros v. 1.0

Repositorio	Proyecto	Etapa	Proceso	Proceso Harvest	Acceso
PESistemaIntegradoBancaSeguros	PESistemaIntegradoBancaSeguros v.1.0	Análisis Técnico en Cartera	Crear Ticket	Create Package	- Gestión Comercial-Analista Técnico - Gestión Comercial-Jefe de Proyecto
			Enviar Ticket a Análisis Técnico en Proceso	Promote Package	- Gestión Comercial-Jefe de Proyecto
			Reporte Estado – Tickets	UDP	- Gestión Comercial-Analista Técnico - Gestión Comercial-Jefe de Proyecto
		Análisis Técnico en Proceso	Enviar a Análisis Técnico Pendiente	Promote Package	- Gestión Comercial-Analista Técnico
			Enviar a Análisis Técnico Rev. Funcional	Promote Package	- Gestión Comercial-Analista Técnico

		Enviar a Evaluación Análisis Técnico	Promote Package	- Gestión Comercial- Analista Técnico
	Análisis Técnico Pendiente	Enviar a Análisis Técnico en Proceso	Promote Package	- Gestión Comercial- Analista Técnico
	Análisis Técnico Rev. Funcional	Enviar a Análisis Técnico en Proceso	Promote Package	- Gestión Comercial- Analista Técnico
	Evaluación Análisis Técnico	Enviar a Análisis Técnico por Aprobar	Promote Package	- Gestión Comercial- Analista Técnico
		Enviar a Análisis Técnico por Aprobar en Espera	Promote Package	- Gestión Comercial- Analista Técnico
	Análisis Técnico por Aprobar	Enviar a Desarrollo	Promote Package	- Gestión Comercial- Jefe de Proyecto
	Análisis Técnico por Aprobar en Espera	Enviar a Análisis Técnico por Aprobar	Promote Package	- Gestión Comercial- Analista Técnico

		Desarrollo	Check Out Lineal	CheckOut Update	- Gestión Comercial-Analista Técnico - Gestión Comercial-Analista Programador
			Check Out Sincronización	CheckOut Synchronize	- Gestión Comercial-Analista Técnico - Gestión Comercial-Analista Programador
			Check In	CheckIn	- Gestión Comercial-Analista Técnico - Gestión Comercial-Analista Programador
			Comparar Versiones	ListVersions	- Gestión Comercial-Analista Técnico - Gestión Comercial-Analista Programador

			Enviar a Desarrollo Pendiente	PromotePackage	- Gestión Comercial-Analista Técnico - Gestión Comercial-Jefe de Proyecto
			Enviar a QA Congelamiento	PromotePackage	- Gestión Comercial-Jefe de Proyecto
		Desarrollo Pendiente	Enviar a Desarrollo	PromotePackage	- Gestión Comercial-Analista Técnico
		QA Congelamiento	Enviar a QA Pruebas	PromotePackage	Perfil Definido por CCAL
			Enviar a QA Sin Pruebas	PromotePackage	Perfil Definido por CCAL
			Regresar a Desarrollo	DemotePackage	Perfil Definido por CCAL
		QA Sin Pruebas	Procesos Definidos por CCAL	Procesos Definidos por CCAL	Perfil Definido por CCAL
		QA Con Pruebas	Procesos Definidos por CCAL	Procesos Definidos por CCAL	Perfil Definido por CCAL
		Producción	Procesos Definidos por CCAL	Procesos Definidos por CCAL	Perfil Definido por CCAL

9. METODO DE TRABAJO PARA REALIZAR EL CHECK IN Y CHECK OUT DE ITEMS EN APLICACIONES VISUAL BASIC, VISUAL INTERDEV Y POWERBUILDER

DESARROLLO EN VISUAL BASIC

1. Realizar el Check Out Synchronize de los items de la aplicación a trabajar, esto actualizará la información de que se encuentra en la PC del usuario con la información que se encuentra en el repositorio.

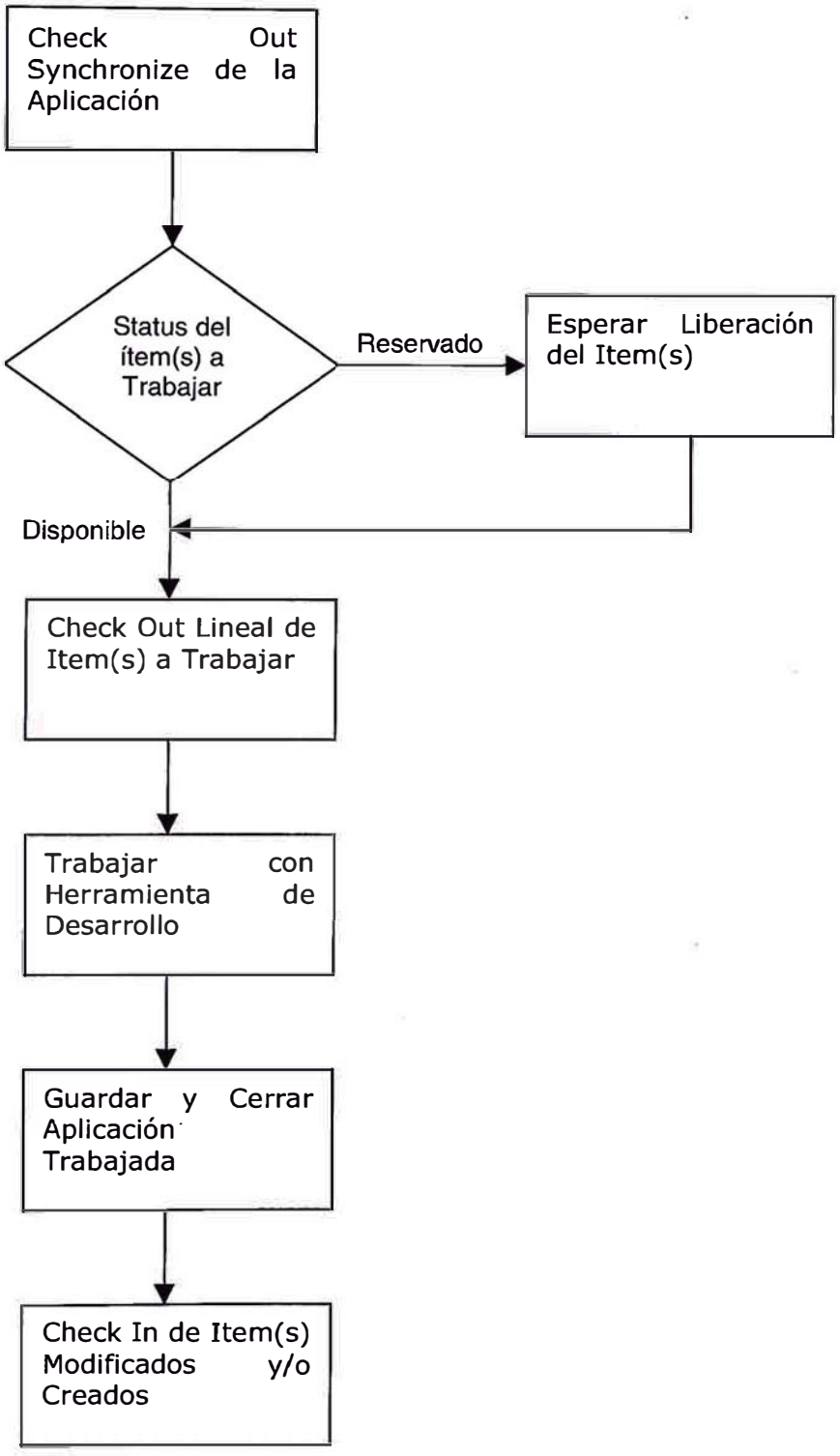
2. Verificar el status del ítem(s) a modificar, si el status es Reservado, entonces esperar hasta que esté(n) liberado(s). Sino, proceder a realizar el Check Out Lineal (esto deja al ítem(s) en modo reservado).

3. Trabajar en la herramienta de desarrollo.

4. Finalizado el trabajo de desarrollo, grabar y cerrar la aplicación trabajada.

5. Proceder a realizar el Check In de los ítems modificados y/o creados.

FLUJOGRAMA PARA DESARROLLO EN VISUAL BASIC



DESARROLLO EN VISUAL INTERDEV

1. Verificar el status del ítem(s) a modificar, si el status es Reservado, entonces esperar hasta que esté(n) liberado(s). Sino, proceder a realizar el Check Out Lineal (esto deja al ítem(s) en modo reservado).

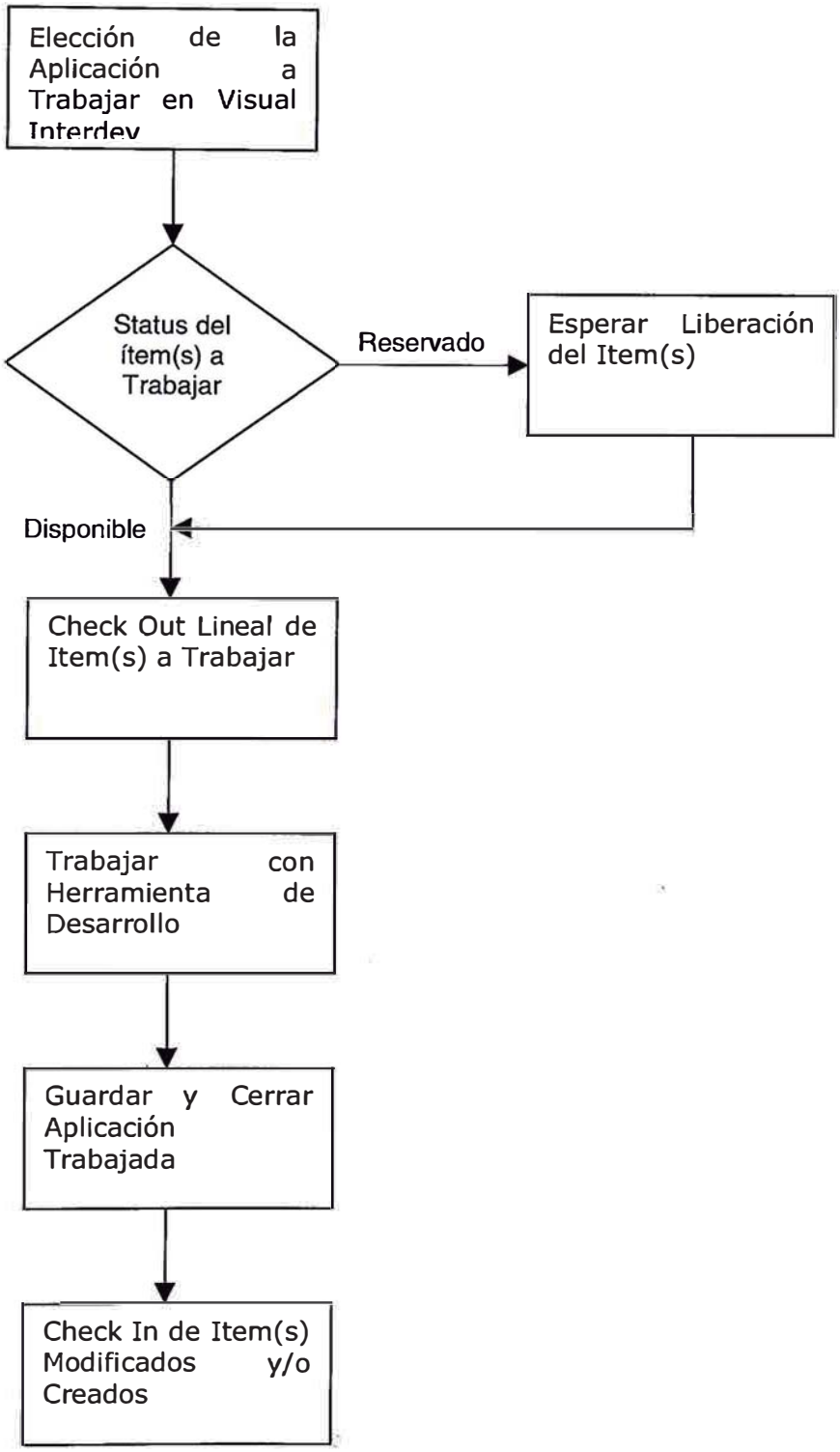
2. Trabajar en la herramienta de desarrollo.

3. Finalizado el trabajo de desarrollo, grabar y cerrar la aplicación trabajada.

4. Proceder a realizar el Check In de los ítems modificados y/o creados.

NOTA: Por ningún motivo se deberá extraer los archivos desde servidor de desarrollo, todo el trabajo se hará a través de CCC/Harvest.

FLUJOGRAMA PARA DESARROLLO EN VISUAL INTERDEV



DESARROLLO EN POWERBUILDER

ADMINISTRATOR

1. Para realizar la carga en el repositorio, copiar los archivos (PBLs, JPEG, etc.) de la aplicación en una carpeta de la PC (el nombre de dicha carpeta no deberá contener espacios en blanco).

2. Ingresar a la herramienta de desarrollo e indicar la ubicación del PBL que contiene la aplicación con la que se trabajará, luego colocar todas las librerías que están involucradas con dicha aplicación.

3. Ingresar a PowerGen (PowerGen 6.x, en caso de estar trabajando con PowerBuilder 6.x), para crear un proyecto que contenga la información de la aplicación. Para esto los pasos a seguir son:
   a) Project – New – Browse – Add (el PBL en donde se encuentre la aplicación a trabajar).
   b) Guardar el proyecto power1.gen (nombre por defecto) y guardarlo en una carpeta aparte, lo recomendable es crear una carpeta Project en el directorio PowerGen ($:\PowerGen\Project\power1.gen).
   c) También crear una carpeta Logs ($:\PowerGen\Logs\powergen.log) para almacenar los archivos .log que se generen, y direccionarlo desde Powergen a esta carpeta (Option – Log file).

4. Ingresar a PowerGen para desagregar los objetos del PBL. Para esto los pasos a seguir son:

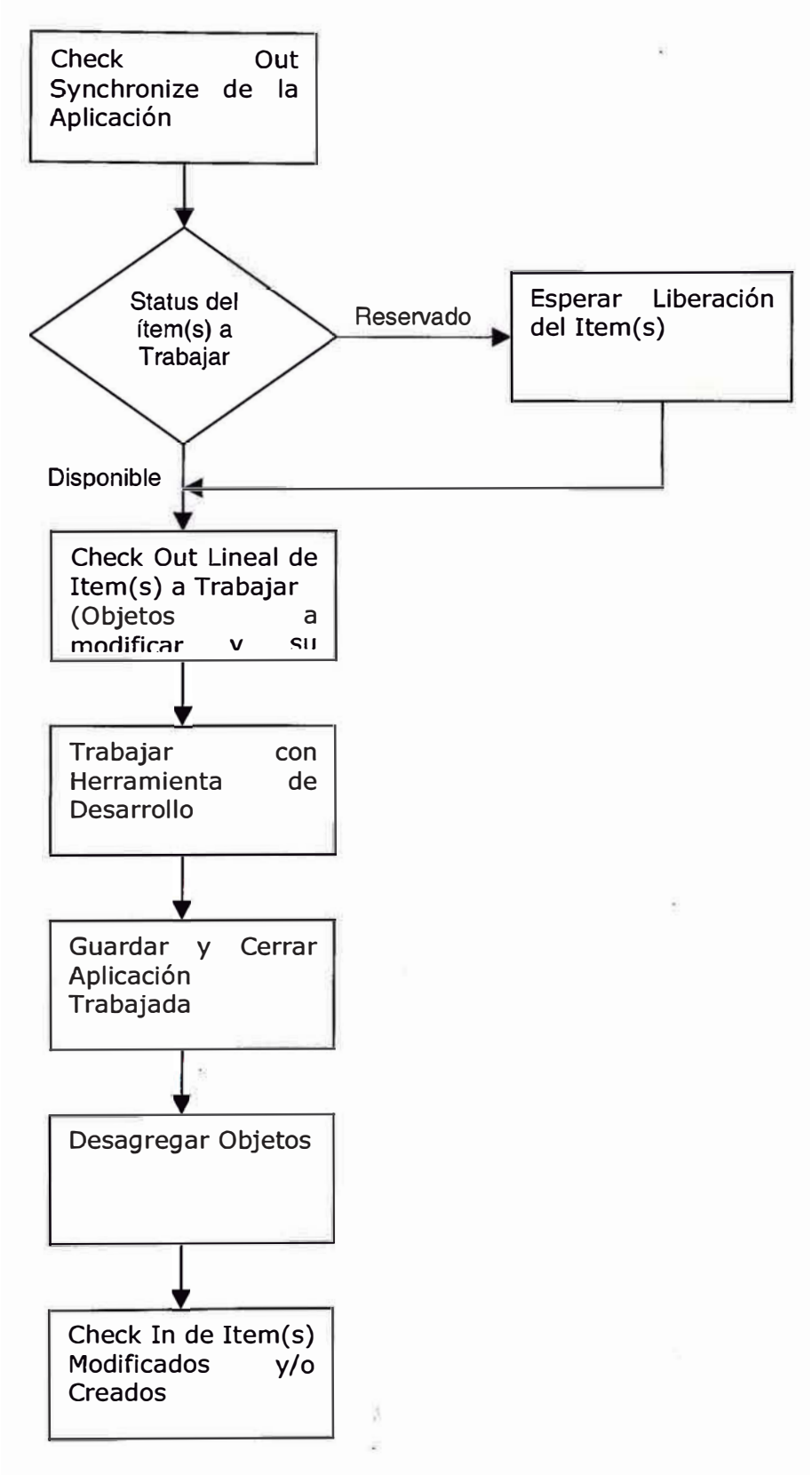
a) Application – Export objects – to directory tree (Esto originará la creación de carpetas conteniendo los objetos de cada PBL, se crea una carpeta por cada PBL desagregado, las cuales toman el nombre del PBL).
b) Señalar el directorio cliente en donde se trabajará (por ejemplo: $:\PHarvest).

5. Realizar la carga del repositorio con toda la información obtenida luego de desagregar los objetos, y borrar los archivos del directorio cliente.

WORKBENCH

1. Realizar el Check Out Synchronize de los ítems de la aplicación a trabajar, esto actualizará la información de que se encuentra en la PC del usuario con la información que se encuentra en el repositorio.
2. Verificar el status del ítem(s) a modificar, si el status es Reservado, entonces esperar hasta que esté(n) liberado(s). Sino, proceder a realizar el Check Out Lineal (esto deja al ítem(s) en modo reservado). Se debe seleccionar el PBL a modificar y los objetos del mismo PBL que se van a modificar.
3. Trabajar en la herramienta de desarrollo.
4. Finalizado el trabajo de desarrollo, grabar y cerrar la aplicación trabajada.
5. Realizar el proceso Desagregar Objetos, que actualizará los objetos del PBL con el que se esta trabajando.
6. Proceder a realizar el Check In de los ítems modificados y/o creados.

FLUJOGRAMA PARA DESARROLLO EN POWERBUILDER



10. RELACION DE PROCESOS DISPONIBLES EN CCC/HARVEST

APROVE (APROBAR)

El proceso Approve permite que usuarios designados aprueben o rechacen un paquete para que pueda ser promovido o trasladado, así que un paquete puede ser promovido o ser movido tan pronto como una aprobación sea completa.

CHECK IN (Subir Modificaciones)

El proceso Check In permite que usted copie archivos desde el sistema del directorio cliente en CCC/Harvest, creando nuevos archivos o actualizando archivos existentes que han sufrido modificaciones. Para realizar el check in de un archivo existente, debe ser reservado por un paquete.

CHECK OUT (Bajar Elementos a Modifcar)

El proceso Check Out permite que usted copie versiones de archivos bajo control de Harvest hacia directorios externos de su PC, en donde los archivos pueden ser actualizados o solamente para lectura. Alternativamente, los archivos pueden simplemente ser reservados sin copiarlos a un directorio del cliente.

COMPARE VIEWS (COMPARAR VISTAS)

El proceso Compare Views genera un reporte que muestra las diferencias entre dos vistas, ya sean de trabajo o snapshots, que existan en cualquier proyecto. Usted puede seleccionar una o todas las opciones siguientes para generar el reporte: archivos que existen solamente en la primera vista, archivos que existen solamente en la segunda vista, los artículos que existen en ambas vistas pero tienen diferente contenido, los artículos que existen en ambas vistas pero tienen contenido idéntico.

CONCURRENT MERGE (FUSION CONCURRENTE)

El proceso Concurrent Merge permite que una versión situada en una rama se convierte en parte del tronco del proyecto. Hasta que se ejecuta este proceso, los cambios realizados en la rama existen solamente en la rama. Después de que todos los archivos hayan llegado a una rama, la invocación del proceso concurrent merge para el paquete que contiene los cambios hace que las versiones en la rama se conviertan en una sola versión en el tronco. La nueva versión creada en el tronco es la última de la visión.

Si existiera una versión en el tronco más reciente de la que fue creada desde la rama, la versión creada por el proceso concurrent merge tendrá una etiqueta de Merge (M), en caso contrario esta sería la última versión del tronco.

CREATE PACKAGE (CREAR PAQUETE)

El proceso Create Package da como resultado la creación de un paquete en el estado definido como el estado inicial. Además, si la opción del formulario es seleccionada, un formulario es asociado automáticamente al crear el paquete.

CROSS PROJECT MERGE (FUSION DE VERSIONES ENTRE PROJECTOS)

El proceso Cross Project Merge se utiliza para combinar los cambios realizados a los archivos en un proyecto con los cambios realizados a los mismos archivos en otro proyecto. La fusión crea las nuevas versiones en el proyecto seleccionado que serán las más últimas de cada archivo en el tronco. Por ejemplo, en el desarrollo orientado a mantenimiento, algunos desarrolladores pueden trabajar en un lanzamiento del mantenimiento tal como Release 1.1, mientras que otros están trabajando en los cambios funcionales importantes para el Release 2.0. Varios archivos serán actualizados en ambos proyectos,

y los grupos necesitan combinar sus cambios para poner al día los archivos en ambos proyectos.

DELETE VERSIONS (BORRAR VERSIONES)

El proceso Delete Versions permite remover versiones seleccionadas del repositorio de Harvest.

DEMOTE (REGRESIÓN DE PAQUETES)

El proceso Demote permite que usted envíe unos o más paquetes en el estado actual a un estado anterior en el ciclo de vida. Cuando un paquete regresa a un estado anterior con una vista diferente, todos los cambios hechos en la vista actual por el paquete se borran y llegan a ser visibles en la vista de destino. Si se requieren aprobaciones para la promoción, el estado de aprobación del paquete cambia cuando regresa.

INTERACTIVE MERGE (FUSIÓN INTERACTIVA)

El proceso de Fusión Interactiva se utiliza para resolver los conflictos existentes en las versiones existentes en las ramas del tronco.

El proceso de Fusión Interactiva permite al usuario combinar los cambios realizados en las ramas del tronco con los cambios del tronco.

LIST VERSION (LISTAR VERSIONES)

El proceso de Listar Versiones permite que los usuarios generen informes sobre los cambios realizados a los ítems del proyecto. El listado producido por este informe está en el mismo formato que el producido por el comando del diff de UNIX.

MOVE PACKAGE (MOVER PAQUETE)

El proceso de mover paquetes mueve un paquete desde un estado en un proyecto a un estado en otro proyecto. Al igual que en el proceso promover, si los requisitos de aprobación se han definido para un

estado, entonces las aprobaciones se verifican antes de que se mueva el paquete.

Al mover un paquete solamente viaja la definición del paquete y la historia, mas no ninguna de las versiones que se pudieran haber trabajado con el paquete en el primer proyecto.

NOTIFY (NOTIFICAR)

El proceso de la notificación utiliza los servicios de correo estándar de MAPI o del smtp para permitir que usted envíe un mensaje a cualquier combinación de usuarios y de grupos de usuarios. Este proceso se puede agregar a un estado o ser ligado a otros procesos de tal manera de que los usuarios señalados automáticamente sean notificados durante la ejecución del proceso de padre.

PROMOTE (PROMOVER PAQUETES)

El proceso de promoción permite enviar uno o más paquetes del estado actual al siguiente estado del ciclo de vida

CONCURRENT UPDATE (ACTUALIZACIÓN CONCURRENTE)

El proceso de actualización concurrente permite que varios usuario modifiquen el mismo ítem a la vez utilizando diferentes paquetes.

Cada una de estas versiones viene a constituir una rama del tronco que luego deberá pasar a formar parte del tronco con la ejecución de los procesos Fusión Concurrente y Fusión Interactivo.

REMOVE ITEM (REMOVER UN ITEM)

El proceso de remover un ítem permite a los usuarios suprimir lógicamente los ítems seleccionados una vista.

El retiro de los ítems se hace parte de los cambios asociados a un paquete y pueden entonces progresar en el ciclo de vida mientras que el paquete se promueve a partir de una vista a otra.

RENAME ÍTEM (RENOMBRAR UN ÍTEM)

Permite renombrar o cambiar de nombre a un ítem.

11. NUEVAS CARACTERISTICAS CCC/HARVEST V.5.1

1. Soporte de Nuevas Plataformas

Cliente	Windows98/NT/2000/**XP***, Solaris, AIX, HP/UX, **Linux***, **Tru64***
Servidor	Windows NT/2000, Solaris, AIX, HP/UX, **Linux***
Agente	Windows 98/NT/2000/**XP***, Solaris, AIX, HP/UX, **Linux***, **Tru64***, MVS
Web	Web Server – MS IIS, iPlanet, **WebSphere***, Apache Servlet engine – Tomcat, JRUN Browser – IE 5.5+, NetScape 6.1/Mozilla
SDK	Windows NT/2000/**XP***, Solaris, AIX, HP/UX, **Linux***, **Tru64***
COM SDK	Windows NT/2000/XP
Base de Datos	Oracle 8.0.6, **8.1.7***
* - Nuevo en 5.1	

- Solaris – 2.6, 2.7, 2.8
- HPUX – 11.00

- AIX – 4.3.3
- Linux – Red Hat 7.1
- Tru64 – V5.0, V5.1

Nota: No existe soporte con Windows 95 debido a la finalización del mantenimiento con Microsoft. Netscape 4.7 no soporta tabs de formularios, y los procesos de Fusión Interactiva y Comparar Vistas.

2. Mejoras en el Trabajo con Formularios

- Los usuarios pueden anexar diversos archivos o direcciones web al formulario como por ejemplo: Un script de pruebas, un documento de cambio, resultados de pruebas, etc., es decir se puede anexar un número ilimitado de documentos, los cuales también pueden ser removidos por los usuarios. Los documentos anexados son almacenados en la Base de Datos de CCC/Harvest.
- Se ha añadido la capacidad de reingresar a la sesión de la construcción del formulario en el Form Wizard, siempre y cuando este no haya sido ingresado a la base de datos.

3. Mejoras en la Interface Web

- Soporte de nuevos procesos, como Interactive Merge (Fusión Interactiva de Versiones), UDP tipo Server (Procesos Definidos por el Usuario), Cross Project Merge (Fusión de Versiones entre Proyectos), Compare Views (Comparar Vistas).

Nota: Para ejecutar el proceso Comparar Vistas se debe contar con Internet Explorer 5.5 o Netscape 6.x / Mozilla 0.9.5.

- Soporte de documentos anexados al formulario.

- Creación de nuevas páginas para la ejecución de los procesos de CCC/Harvest.

- Los usuarios pueden modificar la ruta de las vistas en los procesos de Check In y Check Out.

- Habilidad para remover uno o más paquetes o grupo de paquetes.

- Se ha añadido un selector de directorio cliente en el proceso Check Out y en el selector de contexto.

- Mejora en el diseño de las páginas: la historia del contexto es accesible bajo la visualización del contexto actual. En la visualización del contexto actual se han agregado la opción de tener por defecto las vistas del repositorio y del directorio cliente.

- Páginas pop-up en la ejecución de cualquier proceso de CCC/Harvest.

- Mejoras en la página de edición del contexto: se ha agregado la opción de seleccionar el nombre de la PC del usuario, login y password por defecto, y el directorio cliente por defecto. Esto será usado para la conexión con el cliente.

- Los tabs de los formularios es soportado por Internet Explorer 5.5 o Netscape 6.x.

- Reportes en la Interface Harweb Administrator:

a) Interface de reportes SQL personalizable: El administrador de CCC/Harvest puede ingresar diversos queries de SQL a la base de datos y ver el resultado de estos.

b) Reportes por defecto como: Reporte de accesos, resumen de proyectos, resumen de usuarios, resumen de grupo de usuarios, resumen de repositorios. Para cada proyecto los usuarios pueden generar reportes de ciclo de vida, estados, vistas, accesos, grupo de paquetes, aprobaciones, repositorio. Para cada estado los usuarios pueden generar reportes de procesos, accesos a estados. El administrador puede obtener reportes de modificaciones de ítems, ítems reservados, grupos de usuarios, basado en cada uno de los usuarios

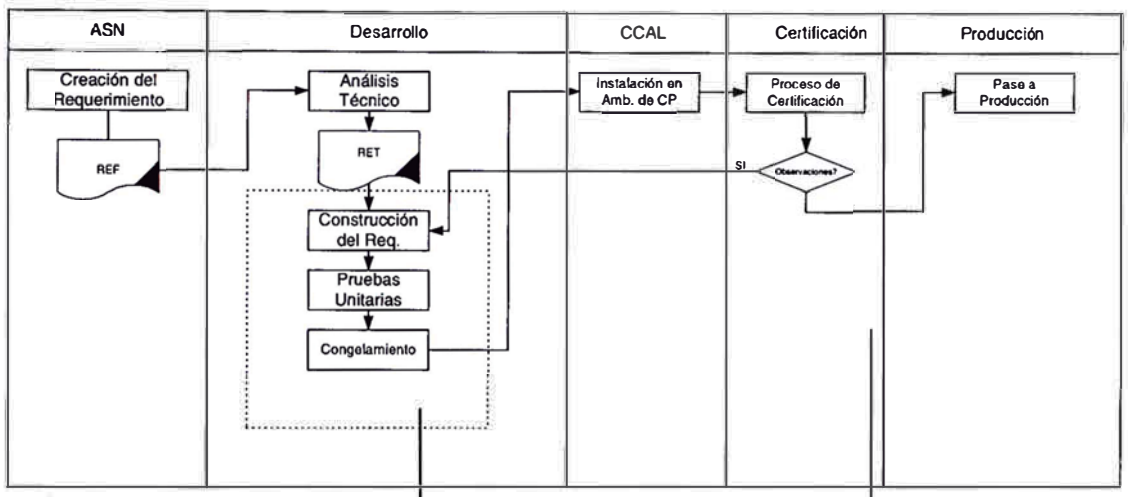
c) El administrador tiene la facultad de navegar y dar accesos a los grupos de usuarios a la estructura del repositorio.

- Reportes en la Interface Harweb Workbench: Nuevos reportes: Historial de paquetes, listado de versiones, listado de últimas versiones, ítems modificados, ítems reservados.

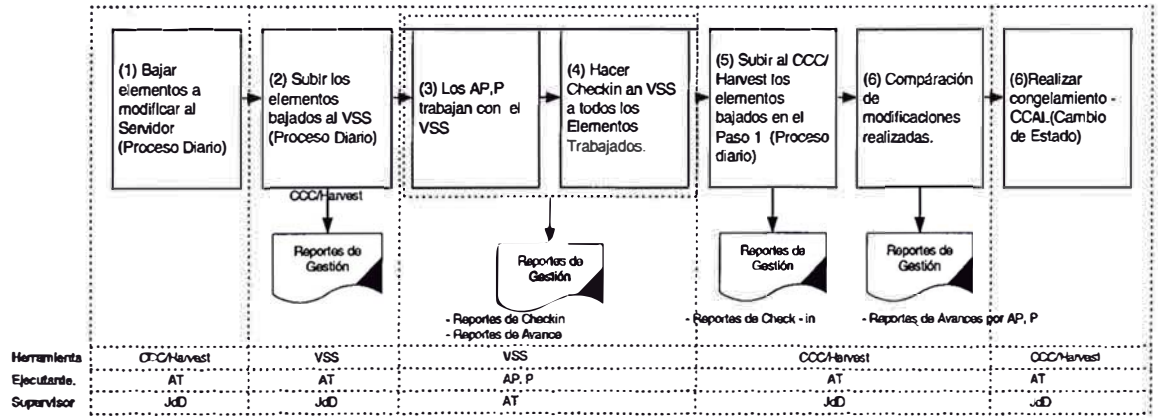
12. IMPLEMENTACION

Proceso Actual de Atención de un Requerimiento

Proceso General



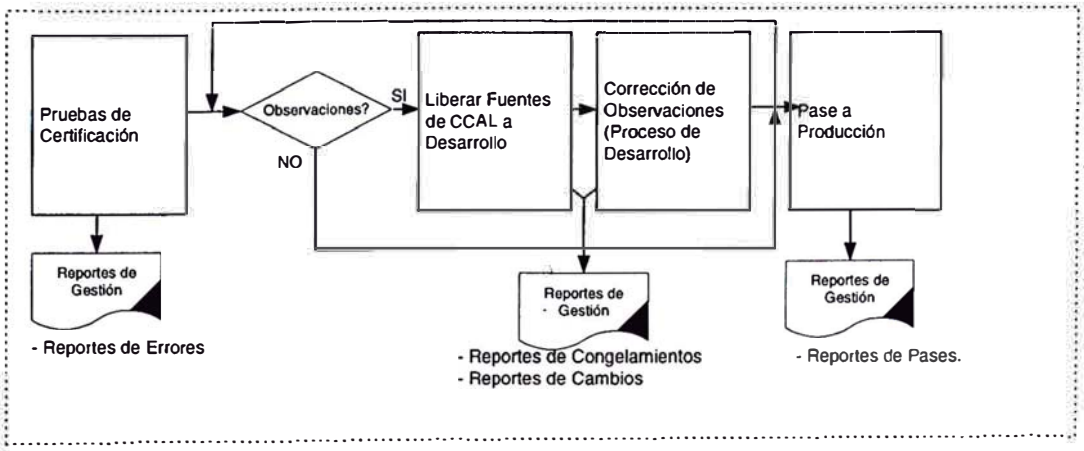
Proceso de Desarrollo



Proceso de CCAL.

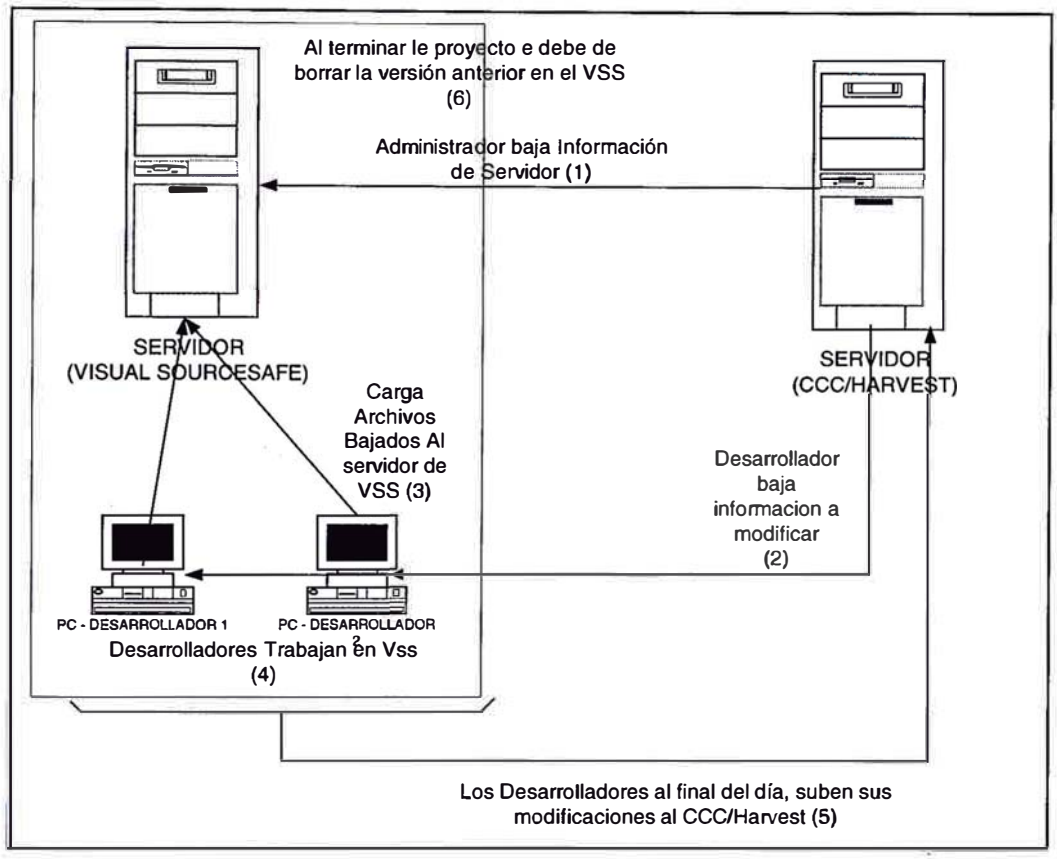
Instalación del Requerimiento realizado en ambientes de Automatización.

Proceso de Certificación y Producción



PASOS A SEGUIR PARA EL DESARROLLO DE UN REQUERIMIENTO.

Propuesta I



1. Crear el Requerimiento.
2. Aprobación del REF.
3. Análisis técnico
4. Aprobación del RET
5. Desarrollo del Requerimiento.

   5.1 Si el Requerimiento no es nuevo crear un nuevo crear un repositorio a partir del último congelado en producción, si es nuevo crear un nuevo repositorio (Con la estructura de Congelamiento).

   5.2 Se baja todos los elementos del requerimiento a un servidor, el cual controla las versiones de los fuentes por medio del VSS.

5.3 Bajar los elementos a modificar del Harvest a la PC del Desarrollador (Analistas Programadores (AP), Programadores (P)).

5.4 Subir dichos elementos al VSS de la PC de los Desarrolladores al VSS.

5.5 Los Analistas Programadores (AP), Programadores (P), comienzan a trabajar como normalmente lo realizan.

5.6 Culminar el Proceso de trabajo Realizando check in a todos los elementos modificados en el VSS.

5.7 Los desarrolladores suben los elementos bajados en le paso 1 al CCC/Harvest.

5.8 Culminado el Desarrollo se procede a congelar (Cambio de Estado en el CCC/Harvest).

6. Instalación del Aplicativo
7. Proceso de Certificación

7.1 Encontrar observaciones a Requerimiento certificado, si las hubiera se procede a:

7.1.1 Cambiar de Estado y Regresar el Requerimiento a Desarrollo.

7.1.2 Posteriormente se continúa con el paso 5.2

7.2 Culminación de la Certificación

8. Pase a Producción.

Comentarios de Está forma de Trabajo (Paso 5).

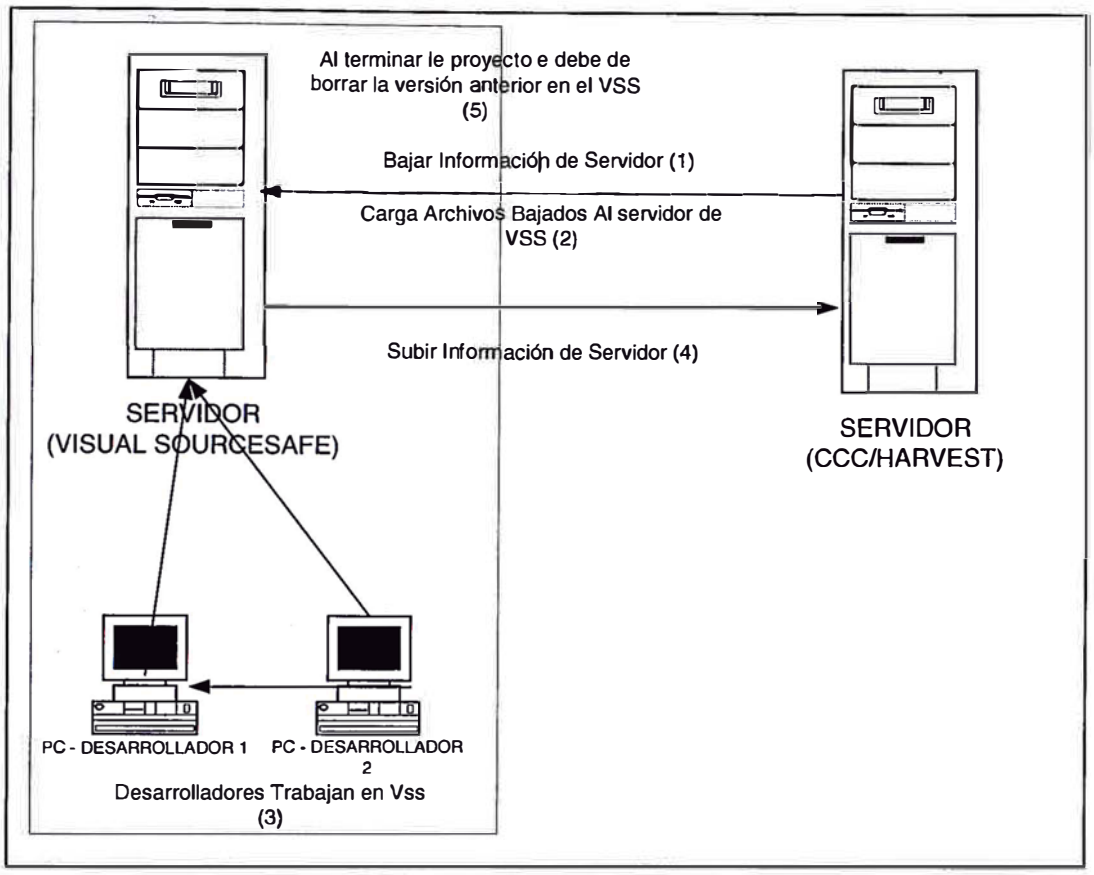
- Los reportes de seguimiento a nivel del CCC/HARVEST, son más precisos y se puede hacer control al desarrollador, pues el quien interactúa directamente con el CCC/Harvest.
- El seguimiento en cuanto al avance se realizará en forma más precisa.

- La potencialidad del CCC/Harvest en cuanto a las herramientas de Gestión sería explotadas en todo su potencial.
- El VSS se convertiría en la herramienta que nos ayuda a no perder el control del elemento bajado, cosa que ocurrirá si no nos ayudamos del VSS.
- El trabajo administrativo disminuye pues esto estaría distribuidos en varias personas (usuarios).
- Este proceso se realizaría diariamente por lo cual se volvería un poco tediosos, por la labor operativa.
- Constituye un gran avance en el proceso de consolidación de fuentes en desarrollo, pues permite completar el ciclo de desarrollo y el control de gestión de todos los desarrolladores.

Riesgos

- Aumento del proceso administrativo, pues permitirá realizar un mejor seguimiento a las tareas diarias del desarrollador (reportes de Gestión).
- Rechazo a la nueva forma de trabajo, por parte de los desarrolladores.
- Necesidad de un administrador que pueda realizar el proceso de bajar los elementos del requerimiento a un servidor.

Propuesta II



1. Crear el Requerimiento.
2. Aprobación del REF.
3. Análisis técnico
4. Aprobación del RET
5. Desarrollo del Requerimiento.
   5.1 Si el Requerimiento no es nuevo crear un nuevo crear un repositorio a partir del último congelado en producción, si es nuevo crear un nuevo repositorio (Con la estructura de Congelamiento).
   5.2 Bajar los elementos a modificar del Harvest a un servidor de Desarrollo.
   5.3 Subir dichos elementos al VSS.
   5.4 Los Analistas Programadores (AP), Programadores (P), comienzan a trabajar como normalmente lo realizan.

5.5 Culminar el Proceso de trabajo Realizando check in a todos los elementos modificados en el VSS.

5.6 Subir los elementos bajados en le paso 1 al CCC/Harvest.

5.7 Culminado el Desarrollo se procede a congelar (Cambio de Estado en el CCC/Harvest).

6. Instalación del Aplicativo
7. Proceso de Certificación

7.1 Encontrar observaciones a Requerimiento certificado, si las hubiera se procede a:

7.1.1 Cambiar de Estado y Regresar el Requerimiento a Desarrollo.

7.1.2 Posteriormente se continúa con el paso 5.2

7.2 Culminación de la Certificación

8. Pase a Producción.

Comentarios de Está forma de Trabajo (Paso 5).

- Los reportes de seguimiento a nivel del CCC/HARVEST, sólo está referido a una sola persona (Usuario), que es la persona que baja los elementos a modificar y que las vuelve a subir.
- Sólo se podrá hacer seguimiento en cuanto al avance pero a nivel de consolidado por ese usuario.
- La potencialidad del CCC/Harvest en cuanto a las herramientas de Gestión no sería explotadas en todo su potencial.
- El VSS se convertiría en la herramienta que puede ayudarnos a realizar dicha gestión pero en forma limitada pues no posee la potencialidad del CCC/Harvest en cuanto a los reportes de gestión.
- Ocasionaría mayor trabajo administrativo, porque una persona se encargaría de todo este trabajo.
- Este proceso se realizaría diariamente por lo cual se volvería un poco tediosos, por la labor operativa.

- Constituye un avance en el proceso de consolidación de fuentes en desarrollo, pues permite completar el ciclo de desarrollo.

Riesgos

- Aumento del proceso administrativo de una persona quien tendría que realizar la acción de subir y bajar los elementos a modificar.
- No usar el CCC/Harvest a todo su potencial pues, los reportes de gestión estaría limitados pues sólo se podría realizar seguimiento al usuario con el cual se bajaron los elementos del Harvest.
- No poder llevar un control de quienes no subieron sus elementos modificados al CCC/Harvest y que es lo que modificaron.
- Rechazo a la nueva forma de trabajo, por parte de los desarrolladores.
- Necesidad de un administrador que pueda realizar el proceso de bajar los elementos del requerimiento a un servidor.

Comparación de CCC/Harvest con El Visual SourceSafe.

– Interfaz CCC/Harvest – Herramienta de desarrollo no disponible (**D**).	– Se integra con la Herramientas de Desarrollo (para VB, VI y VC++) (**V**).
– Posee una Base de Datos Relacional, esto permite una mayor explotación de los datos almacenados (**V**).	– No posee una Base de Datos Relacional, esto no permite explotar o analizar los datos almacenados (**D**).
– Sólo se necesita una versión servidor instalada (**V**).	– Se necesita varias versiones de servidor instaladas (**D**).
– Permite trabajo concurrente y lineal de los desarrolladores (**V**).	– Permite trabajo concurrente y lineal de los desarrolladores (**V**).
– Puede almacenar todo tipo de archivos (**V**).	– Puede almacenar todo tipo de archivos (**V**).
– Posee un **workflow** flexible que permite adaptarse a cualquier forma de trabajo (**V**).	– No posee un **workflow** sólo sirve para almacenar fuentes y controlar el acceso a los mismos y guardar versiones (**D**).
– Incrementa la labor de Administración (**D**).	– Incrementa la labor de Administración (**D**).
– Es necesario implementar mecanismos de control para que la recuperación y actualización de fuentes se realice a través de la herramienta durante la fase de desarrollo (**D**).	– Facilita la recuperación y actualización de fuentes a través de la misma interfaz de la herramienta de desarrollo (**V**).

La empresa a implementado la Fase II, en un principio limitado principalmente por el costo de las licencias del CCC/Harvest.

CAPITULO V: PROCESO DE IMPLEMENTACION

1. Definición de las personas que desempeñaran el rol de administradores.

   a) Administrador del Servidor de Aplicativos
   b) Administrador de SourceSafe/CCC-Harvest.
   c) Administrador de los Aplicativos.
   d) Administrador de BD.

2. Capacitación de los administradores sobre los aplicativos del servicio.

La idea es que los administradores posean toda la documentación de los aplicativos.

3. Generación de Encuestas (ver Anexo 8) Para los Análistas Técnicos y Jefes de Dptos, con la finalidad de informarles de este proceso. Para realizar esta actividad se desarrollo el cronograma adjunto:

Actividad	Dura.	F. Ini	F. Fin
Administración de Servidores (Req 0400240)	20 días	01/07/2002	26/07/2002
Levantamiento de Información	17 días	01/07/2002	23/07/2002
Preparación de Encuestas	6 días	01/07/2002	08/07/2002
Encuestas	2.5 días	01/07/2002	03/07/2002
Aprobación de las Encuestas	1 día	03/07/2002	04/07/2002
Metricas	1.5 días	04/07/2002	05/07/2002
Aprobación de las Métricas	1 día	08/07/2002	08/07/2002
Hito: Encuestas y Metricas Culminadas	0 días	08/07/2002	08/07/2002
Entrevistas con JdD	6 días	09/07/2002	16/07/2002
Abram Roman	0.5 días	09/07/2002	09/07/2002
Ana Yoshikawa	0.5 días	09/07/2002	09/07/2002
Miguel Hernandez	0.5 días	10/07/2002	10/07/2002
Regina Arakaki	0.5 días	10/07/2002	10/07/2002
Elizabeth Alvarado	0.5 días	11/07/2002	11/07/2002
Juan Carlos Davila	0.5 días	11/07/2002	11/07/2002
Carlos Montaño	0.5 días	12/07/2002	12/07/2002
Alberto Timoteo	0.5 días	12/07/2002	12/07/2002
Elsa Botto	0.5 días	15/07/2002	15/07/2002
Tomas Castle	0.5 días	15/07/2002	15/07/2002
Fernando Albujar	0.5 días	16/07/2002	16/07/2002
Enrique Morey	0.5 días	16/07/2002	16/07/2002
Evaluación de la Entrevistas	1 día	17/07/2002	17/07/2002
Proceso de Evaluación	1 día	17/07/2002	17/07/2002
Análisis de los resultados	2 días	18/07/2002	19/07/2002
Proceso de Analisis	2 días	18/07/2002	19/07/2002
Concluciones de las Encuestas	1 día	22/07/2002	22/07/2002
Concluciones	1 día	22/07/2002	22/07/2002
Presentación deResultados	1 día	23/07/2002	23/07/2002
Presentación de Resultados	1 día	23/07/2002	23/07/2002
Hito: Presentación de Conclusiones	1 día	24/07/2002	24/07/2002
Elección del Departamento Piloto	1 día	25/07/2002	25/07/2002
Elección	1 día	25/07/2002	25/07/2002
Hito: Departamento Elegido	0 días	25/07/2002	25/07/2002
Implementación del Piloto	1 día	26/07/2002	26/07/2002
Departamento Piloto	1 día	26/07/2002	26/07/2002
Hito: culminación del Piloto	0 días	26/07/2002	26/07/2002

4. Ubicación Física De Los Servidores

Actividad	Dura.	F. Ini	F. Fin
Proceso de Migración de Servidores	36 días	02/09/2002	21/10/2002
Implementación del Centro de Computo	15 días	02/09/2002	20/09/2002
Movimiento de servidores	21 días	23/09/2002	21/10/2002
Tercer Piso (18)	7 días	23/09/2002	01/10/2002
Primer Piso (45)	7 días	02/10/2002	10/10/2002
RRHH (4)	7 días	11/10/2002	21/10/2002

5. Proceso de Compra De Servidores.
6. Confección de las siguientes plantillas de VmWare con las diferentes configuraciones que presentan todos los servidores de desarrollo, los cuales serán almacenados y guardados en CD's, para su mejor utilización.

Nro.	Componente	Configuración Básica	Otras Herramientas	Sustentación	Software Estándar
1	SQL	Windows NT 4.0 SP6a SQL 6.5 SP5a SNA Server 4.0 SP4		Servidor de BD SQL 6.5 similar al BCPHB2	SI
2	SQL	Windows NT 4.0 SP6a SQL 6.5 SP5a SNA Server 4.0 SP4		Servidor de BD SQL 6.5 ubicado en la Zona Transaccional	SI
3	SQL	Windows NT 4.0 SP6a SQL 7.0 SP3 SNA Server 4.0 SP4		Servidor de BD SQL 7.0 ubicado en la Zona Transaccional	SI
4	SQL	Windows NT 4.0 SP4 SQL 6.5 SP5a SNA Server 4.0 SP3	ADO 2.1 ADO 2.5	Servidor de BD SQL 6.5 ubicado en la Zona Interna	NO
5	SQL	Windows NT 4.0 SP4 SQL 6.5 SP5a	ADO 2.1	Servidor de BD SQL 6.5 similar al BCPSRV2	NO
6	SQL	Windows NT 4.0 SP4 SQL 6.5 SP5a	ADO 2.1	Servidor de BD SQL 6.5 similar al BCPSAPSRV1	NO

		SNA Server 4.0	ADO 2.5		
7	SQL	Windows NT 4.0 SP6a SQL 7.0 SP3	ADO 2.1 ADO 2.5	Servidor de BD SQL 7.0 ubicado en la Zona Interna	SI
8	SQL	Windows NT 4.0 SP6a SQL 7.0 SP3 (no estándar)	Segurinet	Servidor de BD para SEGURINET	NO
9	SQL	Windows NT 4.0 SP6a SQL 7.0 SP2	ADO 2.1	Servidor de BD SQL 7.0 similar al BCPSAPSRV6	NO
10	SQL	Windows NT 4.0 SP6a SQL 6.5 SP5a MTS 2.0 IIS 4.0 SNA Server 4.0	ADO 2.1 ADO 2.5	Servidor de BD SQL 6.5,MTS,IIS ubicado en la Zona Interna	SI
11	SQL	Windows NT 4.0 SP4 MTS 2.0	ADO 2.1	Servidor BD SQL 6.5 y MTS similar al BCPSRV3	NO
12	MTS	Windows NT 4.0 SP6a MTS 2.0	ADO 2.1	Servidor MTS similar al BCPSAPSRV3	SI
13	MTS	Windows NT 4.0 SP4 MTS 2.0 SNA Server 4.0 SP3	ADO 2.1 COMTI	Servidor MTS, SNA, COMTI similar al BCPSAPSRV4	NO
14	MTS	Windows NT 4.0 SP6a MTS 2.0 SNA Server 4.0 SP1	ADO 2.1 COMTI	Servidor MTS, SNA, COMTI similar al BCPMTS1	SI/NO
15	MTS	Windows NT 4.0 SP6a MTS 2.0 SNA Server 4.0 SP4	ADO 2.5 COMTI	Servidor MTS, SNA, COMTI ubicado en la Zona Interna	SI
16	MTS	Windows NT 4.0 SP6a MTS 2.0 SNA Server 4.0	DB2 Websphere Application Server	Servidor de BD y SNA de Pagonet	NO
17	MTS	Windows NT 4.0 SP6a MTS 2.0 SNA Server 4.0 SP3	ADO 2.5 COMTI	Servidor MTS, SNA, COMTI similar al BCPHB4	SI/NO
18	MTS	Windows NT 4.0 SP6a MTS 2.0 SNA Server 4.0 SP 4	COMTI ADO 2.1 ADO 2.5	Servidor MTS, SNA, COMTI ubicado en la Zona Transaccional	SI
19	IIS	Windows NT 4.0 SP6a IIS 4.0	ADO 2.1 ADO 2.5	Servidor Web ubicado en la Zona Interna	SI

20	IIS	Windows NT 4.0 SP6a IIS 4.0	ADO 2.1 ADO 2.5 Site server 3.0	Servidor similar al BCPINTRANET	SI
21	IIS	Windows NT 4.0 SP6a (Enterprise Edition) IIS 4.0	ADO 2.5 XML Praser 3.0	Servidor similar al BCPHB1	SI/NO
22	IIS	Windows NT 4.0 SP6a IIS 4.0	ADO2.1 ADO2.5	Servidor Web ubicado en la Zona DMZ	SI
23	IIS	Windows NT 4.0 SP6a IIS 4.0	Websphere Application Server	Servidor Web de Pagonet	NO
24	FORTE	Windows NT 4.0 SP6a SQL Server 6.5 SP5a	ADO 2.1 Forte Conductor (Workflow) Forte Runtime	Servidor Forte similar al BCPWF1	NO
25	Servidor de Excepciones y DNS	Windows NT 4.0 SP6a Software Base no estandar	Servicio DNS	Servidor DNS Servidor de Excepciones y Pilotos	NO

7. Implementación del Piloto de Administración centralizada de Servidores.

La ideas es partir con un departamento y proceder a migrar todos sus aplicativos al nuevo estándar y reducir su número de servidores, esto se plasma en el siguiente cronograma (tener presente que esto se va a realizar por los 16 Departamentos con que cuenta el Area de Desarrollo de Sistemas:

Actividad	Dura.	F. Ini	F. Fin
Administración Centralizada de Servidores	57.38 días	03/09/2002	21/11/2002
Piloto	57.38 días	03/09/2002	21/11/2002
Departamento Piloto	57.38 días	03/09/2002	21/11/2002
Reunión de Coordinación	0.5 días	03/09/2002	03/09/2002
Servidores	3 días	03/09/2002	06/09/2002
Repotenciación de Servidores	1 día	03/09/2002	04/09/2002
Inventario de Servidores en DESARROLLO	1 día	03/09/2002	04/09/2002
Instalación de VMWare GSX	0.5 días	04/09/2002	04/09/2002
Configuración de VMWARE GSX	0.5 días	04/09/2002	04/09/2002
Creación de Plantillas para VMWARE	1.5 días	05/09/2002	06/09/2002
Capacitación de Aplicativos (Transmisión de Conocimiento)	5.44 días	06/09/2002	13/09/2002
Documentación	2 días	06/09/2002	10/09/2002
Formato Estandar de Documentación	0 días	06/09/2002	06/09/2002
Definición de Documentos por Aplicativo	2 días	06/09/2002	10/09/2002
Cronograma de Capacitación Por Aplicativo (Arquitectura)	3.44 días	10/09/2002	13/09/2002
WinLeasing	2.5 horas	10/09/2002	10/09/2002
Saver +	2.5 horas	10/09/2002	11/09/2002
Archivo Negativo	2.5 horas	11/09/2002	11/09/2002
Cifito	2.5 horas	11/09/2002	11/09/2002
KeyFile	2.5 horas	11/09/2002	12/09/2002
CrediAgro	2.5 horas	12/09/2002	12/09/2002
Safic	2.5 horas	12/09/2002	12/09/2002
Raiting	2.5 horas	12/09/2002	12/09/2002
Caps	2.5 horas	13/09/2002	13/09/2002
Sistema de Alarmas	2.5 horas	13/09/2002	13/09/2002
Sistema de Desembolsos	2.5 horas	13/09/2002	13/09/2002
Proceso de Explotación de Información	3 días	13/09/2002	18/09/2002
Análisis de la información	3 días	13/09/2002	18/09/2002
Liberación de Espacio en disco de los servidores	1 día	27/09/2002	30/09/2002
Revision de Directorios a depurar	0.5 días	27/09/2002	27/09/2002
Revisión de Backups	0.5 días	27/09/2002	30/09/2002
Distribución de los Aplicativos	6.44 días	18/09/2002	27/09/2002
Ambientes	3 días	18/09/2002	23/09/2002
Servidores Desarrollo	1.5 días	18/09/2002	20/09/2002
Servidores Certificación	1.5 días	20/09/2002	23/09/2002
Aplicativos por Servidor	3.44 días	23/09/2002	27/09/2002
WinLeasing	2.5 horas	23/09/2002	24/09/2002
Saver +	2.5 horas	24/09/2002	24/09/2002
Archivo Negativo	2.5 horas	24/09/2002	24/09/2002
Cifilo	2.5 horas	24/09/2002	25/09/2002
KeyFile	2.5 horas	25/09/2002	25/09/2002
CrediAgro	2.5 horas	25/09/2002	25/09/2002
Safic	2.5 horas	25/09/2002	26/09/2002
Raiting	2.5 horas	26/09/2002	26/09/2002
Caps	2.5 horas	26/09/2002	26/09/2002
Sistema de Alarmas	2.5 horas	26/09/2002	27/09/2002
Sistema de Desembolsos	2.5 horas	27/09/2002	27/09/2002
Adecuación de la Documentación al Estandar	38 días	30/09/2002	21/11/2002
Estandares	5 días	30/09/2002	07/10/2002
Carpetas	1 día	30/09/2002	01/10/2002
Componentes	1 día	01/10/2002	02/10/2002
Scripts	1 día	02/10/2002	03/10/2002
Objetos SQL	1 día	03/10/2002	04/10/2002
Copia de Seguridad (Backup)	1 día	04/10/2002	07/10/2002
Adecuación de los Aplicativos al Nuevo Estandar	22 días	07/10/2002	06/11/2002
WinLeasing	2 días	07/10/2002	09/10/2002
Saver +	2 días	09/10/2002	11/10/2002
Archivo Negativo	2 días	11/10/2002	15/10/2002
Cifilo	2 días	15/10/2002	17/10/2002
KeyFile	2 días	17/10/2002	21/10/2002
CrediAgro	2 días	21/10/2002	23/10/2002
Safic	2 días	23/10/2002	25/10/2002
Raiting	2 días	25/10/2002	29/10/2002
Caps	2 días	29/10/2002	31/10/2002
Sistema de Alarmas	2 días	31/10/2002	04/11/2002
Sistema de Desembolsos	2 días	04/11/2002	06/11/2002
Adecuación de los Componentes al Nuevo Estandar	2 días	06/11/2002	08/11/2002
Adecuación del Source Safe/ CCC-Harvest al Estandar	4 días	08/11/2002	14/11/2002
Aplicativos	4 días	08/11/2002	14/11/2002
Adecuación de la Base de Datos	5 días	14/11/2002	21/11/2002
Modelo Entidad Relacion	3 días	14/11/2002	19/11/2002
Identificación de Objetos	2 días	19/11/2002	21/11/2002

8. Licencias De Software

7. Para llevar a cabo esta Implementación se requiere de licencias del Vnware, las cuales se deben de adquirir por servidor.

CONCLUSIONES Y RECOMENDACIONES

I. CONCLUSIONES

Con la Implementación de esta propuesta, se persigue lo siguiente:

- Mejor control sobre nuestros aplicativos, en cuanto a performance, seguridad, cumplimiento de estándares de desarrollo y otros.
- Esto nos permitirá encaminarnos con los objetivos que el ADS persigue (Control de la calidad).
- Mejor administración de nuestros servidores, consiguiendo mayor espacio en disco, menor mantenimiento y mejor utilización de los mismos.
- El uso de la herramienta VmWare, permitirá no realizar compras de servidores de desarrollo, los cuales se podrán repotenciar de acuerdo a las características, que se persigan.
- Se disminuiran los tiempos de armado de ambientes, pues ya se contará con las plantillas necesarias (reduciendo el tiempo a 1 hora).
- Los procesos de repotenciación serán solicitados por los administradores, quienes serán las personas adecuadas para realizar dichas acciones, tomando en cuenta las necesidades reales de los aplicativos.
- El uso de backups permitirán liberar espacio en disco, logrando un mejor control de los mismos, el cual se realizará de manera automática.

- Con la ayuda del CCC/HARVEST podemos administrar mejor los fuentes así como el control de las versiones, y tener un control más de tallado de las líneas de código que se desarrollan. Se podrán implementar ratios como de avance, líneas de código por día, por proceso con lo cual se conseguirá tener una BD con tiempos promedios de programación.
- Con la Ayuda del CCC/Harvest se concentrará en una sola herramienta todos los documentos concernientes a los aplicativos, como fuentes, manuales de instalación, usuario, hojas de pase, documentos de Análisis de los aplicativos, documentos de requerimientos funcionales y otra documentación propia de los mismos.
- El mantenimiento ya no se realizará a 63 servidores si no que sólo a 8, disminuyendo los gastos y mejorando la administración de los mismos.
- La administración del espacio se realizará cuidadosamente de acuerdo a las necesidades de los aplicativos y no se realizarán gastos por compres individuales de discos para los servidores.

II. RECOMENDACIONES

- Todo este proceso de consolidación reacae en el concepto de ordenamiento el cual debe de realizarse paso a paso, sin saltearse los pasos del cronograma.
- Para la mejor implementación de la solución se requiere un Administrador a tiempo completo.
- El compromiso de todo el Area de Sistemas es necesario para llevar a cabo todo esto proyecto y la única forma de lograrlo es mejorando la comunicación y haciendoles parte de nuestro equipo.
- La implementación de estas herramientas cambiara la forma de trabajo de las personas de desarrollo, pudiendo ocasionar

conflictos, los cuales tienen que contrarestarse con informar de lo que se intenta realizar.

- La administración del espacio se realizará cuidadosamente de acuerdo a las necesidades de los aplicativos, teniendoe en cuenta la capacidad de almacenamiento del storage.
- Se intentará que este proceso no sea burocrático, pues ocasionaría un cuello de botella que traería abajo la implementación de este proyecto.

BIBLIOGRAFIA

Referencias

Para seguir el enlace (lo subrayado en azul) oprima ctrl+clic.

ASP

Optimización de scripts ASP

- Developing Scalable Web Applications
- Got Any Cache? por Nancy Winnick Cluts
- Maximizing the Performance of Your Active Server Pages por Nancy Winnick Cluts
- 15 Seconds: Performance Section
- Enhancing Performance in ASP - Part I por Wayne Plourde
- When is Better Worse? Weighing the Technology Trade-Offs por Nancy Winnick Cluts
- Speed and Optimization Resources por Charles Carroll

Optimización de IIS

- The Art and Science of Web Server Tuning with Internet Information Services 5.0
- Leveraging ASP in IIS 5.0 por J.D. Meier

- Tuning IIS 4.0 for High Volume Sites por Michael Stephenson
- Tuning Internet Information Server Performance por Mike Moore
- Navigating the Maze of Settings for Web Server Performance Optimization por Todd Wanke
- Managing Internet Information Server 4.0 for Performance por Hans Hugli

ADO y SQL Server

- Top Ten Tips: Accessing SQL Through ADO and ASP por J.D. Meier
- Improve the Performance of your MDAC Application por Suresh Kannan
- Pooling in the Microsoft Data Access Components por Leland Ahlbeck y Don Willits
- SQL Server: Performance Benchmarks and Guides
- Improving the Performance of Data Access Components with IIS 4.0 por Leland Ahlbeck
- Microsoft Data Access Components (MDAC) and ActiveX Data Objects (ADO) Performance Tips por Leland Ahlbeck
- Microsoft SQL Server 7.0 Practical Performance Tuning and Optimization - The Server Perspective por Damien Lindauer

- Microsoft SQL Server 7.0 Practical Performance Tuning and Optimization - The Application Perspective por Damien Lindauer

- Accessing Recordsets over the Internet por Dino Esposito

Componentes ASP y modelos de hilamiento (threading models)

- ASP Component Guidelines por J.D. Meier

- Q243548: INFO: Design Guidelies for VB Components under ASP

- Threading Models Explained por Nancy Winnick Cluts

- So Happy Together? Using ActiveX components with Active Server Pages por Nancy Winnick Cluts

- Developing Active Server Components with ATL por George Reilly

- Agility in Server Components por Neil Allain

- Building High-Performance Middle-Tier Components with C++ por Jon Flanders

- Active Server Pages and COM Apartments por Don Box

- House of COM: Active Server Pages por Don Box

- House of COM: Contexts por Don Box

- House of COM: Performance Trade-offs of the Windows 2000 Component Execution Environment por Don Box

- Building COM Components That Take Full Advantage of Visual Basic and Scripting por Ivo Salmre

- Component Design Principles for MTS

Manejo de estados de session.

- Q175167: HOWTO: Persisting Values Without Sessions
- Q157906: HOWTO: How To Maintain State Across Pages with VBScript
- XML-based Persistence Behaviors Fix Web Farm Headaches por Aaron Skonnard
- House of COM: Stateless Programming por Don Box
- MSDN Magazine: Basic Instinct To Cache or not to Cache por Ted Pattison

Rendimiento y escalabilidad

- Blueprint for Building Web Sites Using the Microsoft Windows Platform
- Server Performance and Scalability Killers por George Reilly
- Microsoft Visual Studio Scalability Center
- Fitch & Mather Stocks 2000
- Tuning the FMStocks Application
- High-Performance Visual Basic Apps por Ken Spencer
- Duwamish Books, Phase 4
- Top Windows DNA Performance Mistakes and How to Prevent Them por Gary Geiger y Jon Pulsipher
- Building from Static HTML to High-Performance Web-Farms by Shawn Bice

Herramientas

- Microsoft Web Application Stress Tool
- I Can't Stress It Enough -- Load Test Your ASP Application por J.D. Meier
- Windows DNA Performance Kit
- Monitoring Events in Distributed Applications Using Visual Studio Analyzer by Mai-Ian Tomsen

Libros

- Professional Active Server Pages 3.0, Wrox Press. (Especialmente el capitulo 26: Optimizing ASP Performance, por George Reilly and Matthew Gibbs)
- Microsoft Internet Information Services 5.0 Resource Guide (parte de Windows 2000 Server Resource Kit, Microsoft Press.
- Microsoft Internet Information Server Resource Kit (para IIS 4.0), Microsoft Press.
- Programming Distributed Applications with COM and Microsoft Visual Basic 6.0 por Ted Pattison, Microsoft Press.
- Effective COM por Don Box, Keith Brown, Tim Ewald, y Chris Sells; Addison-Wesley.
- Developing Web Usability: The Practice of Simplicity por Jakob Nielsen, New Riders.

Web sites sobre ASP

- Microsoft TechNet for IIS

- LearnASP.com
- 4GuysFromRolla.com
- 15Seconds.com
- AspToday.com
- Asp101.com
- AspLists.com. Algunos mailing lists especializados:
  - Fast Code!
  - ASP Advanced
  - Not Newbie
  - State Management
  - Scalability
  - Visual Basic Components
  - XML
  - C++/ATL Component Building
- UseIt.com: Web Usability

Estilo de programación de ASP

- ASP Best Practices por George Reilly
- ASP Quick Lessons por Charles Carroll
- Planning for ASP por John Meade
- ASP Guidelines por J.D. Meier

XML

- Inside XML Performance por Chris Lovett
- Inside MSXML3 Performance por Chris Lovett

ANEXOS

ANEXO 1: ESTÁNDARES DE PROGRAMACIÓN – VISUAL BASIC

INDICE

Estándares de Programación - Visual Basic

1. Objetivos.

- Reglamentar la documentación de programas para lograr un mejor entendimiento de los mismos.
- Mejorar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

2. *Declaración de Variables.*

Se propone la utilización de variables que mejor describan su función para el completo entendimiento del programador. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente

- La longitud debe ser lo más recomendable posible, no debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 13 caracteres.
- El ámbito en que desenvolverán las variables.
- El tipo de dato al que pertenece la variable.

Por lo tanto la estructura de la variable es como sigue:

Estructura	**Ámbito**	**Tipo de Dato**	**Descripción de la Variable**
LONGITUD. MAX.	← 1 →	← 3 →	← 9 →
FORMATO	*Minúscula*	*minúscula*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*g*	*Int*	*NumCta*

2.1 Ámbito.

Describe el ámbito en que la variable trabajará, además también sirve para definir constantes y variables que serán usadas como parámetros en procedimientos y funciones, este debe estar en minúscula. El caracter corresponde a la tabla siguiente.

g	Variable Global, es reconocida en cualquier parte de la aplicación.
p	Variable Privada, es reconocida solamente por el modulo en la que es declarada.
l	Variable Local, es reconocida tan solo en el Procedimiento o Función en la que fue declarada, al terminar este, la variable es destruida.
c	Constante

x	Variable de Parámetro

2.2 Tipo de Dato.

Define el tipo de dato al que pertenece la variable. Los mnemotécnicos corresponden a la tabla.

Mnemotécnico	Tipo de Dato
bln	Boolean
byt	Byte
col	Colección de Objetos
cur	Currency
dte	Date
tme	Time
dbl	Double
err	Error
int	Integer
lng	Long
obj	Object
Sng	Single
str	String
Udt	Tipo definido por el usuario.
ptr	Puntero
Vnt	Variant
Udc	Instancia de una clase definida por el usuario.

2.3 Descripción de la Variable.

Combinación de palabras que mejor describa la función de la variable, en esta combinación debe de comenzar con mayúscula y por cada cambio de palabras descriptivas volver a comenzar con mayúscula, cada palabra que se utilice no debe exceder los tres dígitos.

Ejemplos
gstrNomCli
pintConArt
cintIntAnu

2.4 Variables de Tipo Arreglo.

En el caso de las definiciones de arreglos de elementos, entonces la estructura de la variable cambiara en la adición de una "a" (minúscula) entre el primer dígito (tipo de variable) y la definición de tipo.

Ejemplos

GaintNumCli
PastrMnuPri

2.5 Variables con Referencia a Base de Datos.

Debemos también contemplar, que para el caso de variables que hagan referencia a campos en una base de datos, la estructura del prefijo del mnemotécnico será la descrita anteriormente, pero la descripción de la variable, obligatoriamente deberá contener el mismo nombre del campo al que hace referencia. Es para este caso en el que el limite máximo del tamaño establecido para una variable (13 caracteres), puede ser excedido.

Ejemplos
gstrNombreCliente
pintCostoUnitario
lbooSexo

3. *Definición de Controles.*

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo de control al cual pertenece y la función que cumple dentro de la aplicación.

Estructura	Prefijo	Nombre Descriptivo de la Variable
LONGITUD. MAX.	← 3 →	← ..... →
FORMATO	*minúscula*	*Primer carácter Mayúscula siguientes minúsculas(por palabra)*
EJEMPLO	*ado*	*NumCta*

3.1 Prefijo para el Control.

Formado por los tres primeros caracteres que representan los prefijos que identifican el tipo de control con el que se está trabajando. Los tipos de controles están definidos en la tabla de prefijos de los controles más utilizados presentada a continuación.

3.2 Nombre descriptivo del Control.

Formado por la descripción de la función que lleva a cabo el control, esta debe ser descrita en forma específica y clara.

Tipo de Control	Prefijo	Ejemplo
3DPanel	Pnl	pnlGrupo
ADO Data	Ado	adoBiblio
CheckBox	Chk	chkSoloLectura
ComboBox	Cbo	cboLenguaje
CommandButton	Cmd	cmdSalir
CommonDialog	Dlg	dlgFileOpen
Control (no especificado)	ctr	ctrActual
Data	dat	datBiblio
DataBound-ComboBox	dbc	dbcLenguaje
DataBound-ListBox	dbl	dblTipoTrabajo
DataBound-Grid	dbg	dbgResultadoQry
Data-ComboBox	dcb	dcbAutor
Data-ListBox	dls	dlsPublicista
Data-Grid	dgr	dgrTitulos
DirectoryListBox	dir	dirOrigen

Form	frm	frmPrincipal
Frame	fra	fraLenguaje
Gauge	gau	gauStatus
Graph	gra	graHistorico
grid	grd	grdDetallePedido
FlexGrig	flex	flexPedidos
HorizontalScrollBar	hsb	hbsVolumen
Image	img	imgIcono
ImageCombo	imc	imcProducto
ImageList	ils	ilsListaIconos
Label	lbl	lblNombreCampo
Line	lin	LinSeparador
ListBox	lst	LstNumerosTelefonicos
ListView	lvw	lvwEncabezados
Menu	mnu	mnuArchivo
OptionButton	opt	optGenero
PictureBox	pic	picGaleria
ProgressBar	pgb	pgbCargar
RichTextBox	rtf	rtfReport
Shape	shp	shpCirculo
Slider	sld	sldEscala
Spin	spn	spnNumeroPaginas
StatusBar	sta	staDiaHora
SysInfo	sys	sysMonitor
TabStrip	tab	tabOpcion
TextBox	txt	txtDireccion
Timer	tmr	tmrCronometro
ToolBar	tlb	tlbAccion
TreeView	tre	treOrganizacion
VerticalScrollBar	vsb	vsbAngulo

Nota

- No se deberá utilizar el símbolo de subrayado (_) para determinar el nombre de un control.

- Las tildes así como el uso de la letra “ñ” está prohibido en la determinación de nombres de controles.

- Se debe de tomar en cuenta las restricciones propias del lenguaje con respecto a la definición de nombres de controles.

- Todo control que haga referencia a campos en una base de datos deberá utilizar en su nombre el prefijo del control, y en la descripción obligatoriamente deberá contener el mismo nombre del campo al que hace referencia.

  Ejemplo:

  Si en un control TextBox se quiere capturar o mostrar los valores del campo **clieNombre** que se encuentra en la tabla Clientes el nombre del control sería **txtClieNombre** .

4. *Definición de Menús.*

Estructura	Prefijo	Nombre Descriptivo de la Variable
LONGITUD. MAX.	← 3 →	← ..... →
FORMATO	*minúscula*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	**mnu**	Archivo **Arc**Abrir

En el proceso de creación de un Sistema de Menús se debe tener en cuenta las siguientes consideraciones (para la creación de un *Sistema de Menús* consideraremos que un *Menú* es un elemento de nivel superior que agrupa dentro de sí un conjunto de *Elementos de Menú*):

- Todo objeto Menu debe utilizar el prefijo **mnu**.
- Cada *Menú* debe tener un nombre que describa las tareas que desarrollan sus elementos.
  p.e. **mnu**Archivo

- Los nombres de los *Elementos de Menú* se determinan basándose en la abreviatura (3 caracteres) del nombre del *menú* al cual pertenecen (no tomar en cuenta los prefijos del mismo) y en el nombre que mejor identifique la tarea que cumple.
  p.e. **mnu**Archivo
  p.e. mnu**Arc**Abrir
- Si se cuenta con *submenús*, la asignación de los nombres para los *elementos del submenú* siguen el mismo criterio explicado anteriormente para la creación de elementos de menús para un menú.
  p.e.
  mnu*Archivo*
  mnu**Arc***Imprimir*
  mnuArc**Imp***AreaSeleccionada*
  mnuArc**Imp***Todo*

- Si se utilizan *separadores* dentro de los elementos de un menú o dentro de submenús, el nombre del mismo debe ser **mnuSeparador*n*** (donde n es el número de separador que se está creando).
  p.e. **mnu**Separador1
- Se prohibe el uso de *arreglos de menús* debido a cuestiones de mantenimiento de los mismos.
- Si es necesario el bloqueo de ciertas opciones de menú por cuestiones de seguridad, se deberá utilizar una base de datos que almacene la información necesaria que permita configurar el acceso a los diferentes elementos del sistema de menús.

5. *Procedimientos y Funciones definidos por el Usuaio.*

El nombre de las funciones y procedimientos debe ser autodescriptivo de manera que no se requiera, en lo posible, entrar al código de la función para saber qué es lo que realiza.

Estructura	**Ámbito**	**Indicador Función o Procedimiento**	**Verbo (Acción)**	**Sustantivo**
LONGITUD. MAX.	← 1 →	← 1 →	← . . . →	← . . . →
FORMATO	Minúscula	Minúscula	*Primer carácter de cada palabra en mayúscula.*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*G*	*P*	*Asignar*	*Nombre*

Donde:

- **Ámbito**: puede ser:

g	Público o global
p	Privado

- **Indicador Función/Procedimiento**

f	Función
p	Procedimiento

Verbo-Sustantivo

El estándar para nombres de procedimiento es usar un Verbo que describa la acción realizada seguido por un sustantivo (objeto sobre el cual actúa). Se recomienda:

- Usar un nombre que represente una acción y un objeto. El nombre del procedimiento debe indicar qué hace el procedimiento a... o qué hace el procedimiento con....
- El verbo debe estar en infinitivo.
  - Ser consistente en el orden de las palabras. Si se va a usar ***VerboNombre***, siempre usar *VerboNombre*.
  - Ser consistente en los verbos y sustantivos usados. Por ejemplo si tiene un procedimiento **gp*AsignarNombre***, también usar **gp*AsignarApellido*** en vez de **gp*ColocarApellido***.
  - En el caso que el objeto al cual se refiere el verbo esta compuesto por más de una palabra, por ejemplo:

Para la acción **actualizar los saldos del cliente** se define:

gp*ActualizarSaldosCliente*

Verbo: Actualizar

Palabra Compuesta: SaldosCliente

Nota:

- No se hará uso de los caracteres:
  - Espacio en blanco " "
  - Carácter de subrayado "_"
- La nomenclatura de argumentos o parámetros pasados a los procedimientos/funciones así como para valores devueltos por funciones sigue las mismas convenciones que la nomenclatura para variables.

6. *Definición de Clases*

Para identificar las clases utilizaremos la siguiente nomenclatura:

Estructura	**Indentificador de Clase**	**Sustantivo**
LONGITUD. MAX.	← 3 →	← . . . . →
FORMATO	Minúscula	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	Cls	Cliente

6.1 Identificador de clase:

Siempre sera **cls**

6.2 Sustantivo:

Nombre que identifique claramente al objeto.

- Dentro de las clases se considera la nomenclatura de:

6.3 Propiedades:

Estructura	**Sustantivo**
FORMATO	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*Dirección*

Sustantivo: Nombre que identifique claramente la propiedad del objeto.

En el caso que la propiedad esté compuesta por más de una palabra, por ejemplo:
Para definir el lugar donde se almacenará el promedio semestral, se define:

PromedioSemestral

6.4 Métodos:

Estructura	**Verbo**	**Sustantivo**
FORMATO	*Primer carácter de cada palabra en mayúscula.*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*Insertar*	*Datos*

6.4 VerboSustantivo

- El estándar para nombres de procedimiento es usar un Verbo que describa la acción realizada seguido por un sustantivo (objeto sobre el cual actúa).
- En el caso que el objeto al cual se refiere el verbo esta compuesto por más de una palabra, por ejemplo:

Para la acción **actualizar los saldos del cliente** se define:

ActualizarSaldosCliente

Verbo: Actualizar

Palabra Compuesta: SaldosCliente

6.6 Eventos:

Estructura	Identificador de evento	Nombre del Evento
FORMATO	on	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	on	*ErrorDatos*

6.7 Descripción del Evento

Esta debe ser escrita en forma clara y específica. Puede estar formada por mas de 1 palabra (verbo o sustantivo) El primer carácter de cada una debe empezar con mayúsculas y no se deberá usar el símbolo de subrayado.

7. *Consideraciones para Visual Basic*

En adelante se describen muchas consideraciones generales, de performance recomendaciones para el buen desarrollo de aplicaciones en este entorno.

7.1 Reglas de codificación

- Los códigos deben ser autodocumentados y de preferencia que no ocupen más de dos pantallas para no hacer complicado el seguimiento.

  Se deben evitar los bucles innecesarios, el uso de variables auxiliares innecesarias.
  El código debe ser indentado para facilitar el seguimiento
  Por ejemplo:

```
Rutina para la calculo de planilla
        Leer_BasedeDatos();
        Obtener_Codigo_Cliente();
        Obtener_TasadeAumento();
        Realizar_Calculo();
        Grabar_Información();
```

- Utilizar la tecla TAB, en lugar de la barra espaciadora, para indentar el código . Esto lo hace más facil de interpretar y de hacerle seguimiento

  Ejemplo :

```
If Var1 = 0  then
        If Var2 <> " " then
                For icont = 1 to 25
                        MsgBox "  Mensaje "
                End For
        End  If
End If
```

- Para módulos,

  Para identificar a los Módulos utilizaremos la siguiente nomenclatura
  modXxxxxx

  mod : Identificador de Módulo
  Xxxxx : Descripción de lo que almacena el módulo

  Ejemplo :
  ModParametros

7.2 Otras consideraciones para Visual Basic

- Los nombres de los formularios tanto físicos como logicos deben ser los mismos

- Debe existir un archivo *.ini ubicado en la raíz del proyecto que contenga los parametros globales.

- De existir reportes serán ubicados en el directorio "Reportes".

- Cualquier tipo de gráfica será alojada en un directorio "imagenes".

- Cualquier otro tipo de objeto no nativo de visual basic será alojado en directorio individual.

- Todas las variables a utilizarse deberán ser declaradas (Option Explicit) , esto optimiza el uso de la memoria. Agrupar las declaraciones por Tipo de Dato

- Evitar declaraciones generales tales como : Dim MiObjeto as Object , esto hace que el Transaction Server no pueda interceptar las llamadas de los objetos

  En su lugar indicar la clase exacta del objeto del cual vamos a crear una instancia .

  Ejemplo : Dim MiObjeto as Lib.Clase

- Los Objetos que no se utilicen deberán ser liberados

  Ejemplo : Set MiObjeto = Nothing

- No utilizar sentencias SQL desde el Visual Basic , en su lugar utilizar llamadas a Stored Procedures

- Las rutinas de Validación deben ser disparadas cuando se presione algun botón que implique alguna acción , por ejemplo el registro de información. No utilizar rutinas de validación en los eventos LostFocus de los Controles, esto porque algunas veces , dependiendo de la PC , no se llegan a disparar.

- Si se va a trabajar con Imágenes , utilizar el Control " Image " en lugar del " Picture ", ya que este último consume mas memoria.

- Para aquellos gráficos que tengan que usarse en varios lugares de la aplicación se deberá utilizar un archivo de recursos . (.res)

- Comentarios,

- Cabecera

Cada Form, MDI Form, Module, Class Module, User Control , Property Page, DHTML Page, Data Report, Web Class, etc. Debe comenzar con una cabecera de declaración del siguiente tipo.

```
' ************************************************
' Procedimiento   : frmSeleccionarElementos
' Proposito       : Este Form es responsable de ...
' Inputs          : N/A
' Se asume        : N/A
' Efectos         : N/A
' Retorno         : N/A
' Notas           : N/A
' Modificaciones          : N/A
' Empresa         : XXXXX
' Autor           : Angel Aquino Quintana
' Fecha y hora    : 15/06/2001 - 04:13 pm.
' ************************************************
```

Adicionalmente cada procedimiento o funcion debe estar estructurado del siguiente modo:

```
Function blnValidarDatos()
' ************************************************************
' Proposito       : Este procedimiento es responsable de ...
' ************************************************************

' Cuerpo de la Funcion o Procedimiento

End Function
```

➢ Comentarios Genéricos

Aquellos declarados con un apóstrofe (').Queda a discreción del desarrollador la inclusión de comentarios en las porciones de código que considere conveniente. Los comentarios en código deben ser breves y precisos, ya que se considera que el código debe auto-documentarse por si solo.

- Establecer valores adecuados de "Tab Index" para componentes visuales.
- En cada formulario asignar las propiedades de cancel y default a los botones que mejor se ajusten.
- Emplear MsgBox homogénenos. Del tipo:

```
Mensaje = MsgBox ("mensaje",vbExclamation,"titulo")
```

- Todo los formularios poseen un ícono distintivo.
- Asignar las siguiente propiedades a los Formularios:
  - Border Style
  - StartUp Position

- Moveable
- Shown In TaskBar

ANEXO 2: ESTÁNDARES DE PROGRAMACIÓN - POWER BUILDER

INDICE

8. Objetivos.

- Reglamentar la documentación de programas desarrollados en Power Builder, para lograr un mejor entendimiento de los mismos.
- Mejorar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

9. *Declaración de Variables en Power Script.*

Se propone la utilización de variables que mejor describan su función para el completo entendimiento del programador. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente

- La longitud debe ser lo más recomendable posible, no debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 13 caracteres.
- El ámbito en que desenvolverán las variables.
- El tipo de dato al que pertenece la variable.

Por lo tanto la estructura de la variable es como sigue:

Estructura	Ámbito	Tipo de Dato	Descripción de la Variable
LONGITUD. MAX.	← 1 →	← 3 →	← 9 →
FORMATO	Minúscula	Minúscula	Primer carácter de cada palabra en mayúscula.
EJEMPLO	g	Int	NumCta

9.1. Ámbito.

Describe el ámbito en que la variable trabajará, además también sirve para definir constantes y variables que serán usadas como parámetros en procedimientos y funciones, este debe estar en minúscula. El caracter corresponde a la tabla siguiente.

g	Variable Global, es reconocida en cualquier parte de la aplicación.
p	Variable Privada, es reconocida solamente por el modulo en la que es declarada.
l	Variable Local, es reconocida tan solo en el Procedimiento o Función en la que fue declarada, al terminar este, la variable es destruida.
i	Variable de Instancia
s	Constante
c	Variable Shared
x	Variable de Parámetro

9.2. Tipo de Dato.

Define el tipo de dato al que pertenece la variable. Los mnemotécnicos corresponden a la tabla.

Mnemotécnico	Tipo de Dato
bln	Boolean
blb	Binary long object (blob)
byt	Byte
chr	Char
dte	Date
dcm	Decimal
dst	Datastore
dtm	Datatime
dtw	Datawindow
dwc	Datawindow Child
tme	Time
dbl	Double
emn	Enumerated
gro	Graphic Object
int	Integer
lng	Long
lvw	Listview
lvi	Listview Item
mnu	Menu
pwo	Power Object
rea	Real
str	String
stc	Structure
tab	Tab
tim	Time
tro	Transaction Object
tvw	Treeview
uin	Unsigned Integer
uln	Unsigned Long
udt	Tipo definido por el usuario.
wnd	Window

9.3. Descripción de la Variable.

Combinación de palabras que mejor describa la función de la variable, en esta combinación debe de comenzar con mayúscula y por cada cambio de

palabras descriptivas volver a comenzar con mayúscula, cada palabra que se utilice no debe exceder los tres dígitos.

Ejemplos
gstrNomCli
pintConArt
cintIntAnu

9.4. Variables de Tipo Arreglo.

En el caso de las definiciones de arreglos de elementos, entonces la estructura de la variable cambiara en la adición de una "a" (minúscula) entre el primer dígito (tipo de variable) y la definición de tipo.

Ejemplos
gaintnumcli
pastrmnupri

9.5. Variables con Referencia a Base de Datos.

Debemos también contemplar, que para el caso de variables que hagan referencia a campos en una base de datos, la estructura del prefijo del mnemotécnico será la descrita anteriormente, pero la descripción de la variable, obligatoriamente deberá contener el mismo nombre del campo al que hace referencia. Es para este caso en el que el limite máximo del tamaño establecido para una variable (13 caracteres), puede ser excedido.

Ejemplos
gstrNombreCliente
pintCostoUnitario
lbooSexo

10. Definición de Controles.

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo de control al cual pertenece y la función que cumple dentro de la aplicación.

Estructura	Prefijo	Nombre Descriptivo de la Variable
LONGITUD. MAX.	← 3 →	← ..... →
FORMATO	minúscula	Primer carácter Mayúscula siguientes minúsculas(por palabra)
EJEMPLO	ado	NumCta

10.1. Prefijo para el Control.

Formado por los tres primeros caracteres que representan los prefijos que identifican el tipo de control con el que se está trabajando. Los tipos de controles están definidos en la tabla de prefijos de los controles más utilizados presentada a continuación.

10.2. Nombre Descriptivo del Control.

Formado por la descripción de la función que lleva a cabo el control, esta debe ser descrita en forma específica y clara.

Tipo de Control	Prefijo	Ejemplo
Aplication	app	appNombreEjecutable
CheckBox	chk	chkSoloLectura
CommandButton	cmd	cmdSalir
DataPipeline	dpl	dplComunicacion
DataWindow	dwn	dwnListaArticulo
DropDownListBox	dlb	dlbPai
DropDownPictureListBox	dpl	dplProvincia
DirectoryListBox	dir	dirOrigen
EditMask	edt	edtMontoSoles
GroupBox	grb	grbDatosGenerales
Graph	gra	graHistorico
HScrollBar	hsb	hsbVolumen

Line	lin	LinSeparador
ListBox	lst	LstNumerosTelefonicos
ListView	lvw	lvwEncabezado
MultiLineEdit	mle	mleComentario
OLEControl	ole	oleChart
OLEControlCustom	olc	oleWord
Oval	ova	ovaContorno
Picture	pic	picPortada
PictureButton	pcb	pcbGuardar
PictureListBox	pcl	pclEspecies
ProgressBar	pgb	pgbCargar
RadioButton	rdb	rdbSexo
Rectangle	rec	recMarcoMapa
RichTextBox	rtb	rtbReport
RoundRectangle	rre	rreBordeMapa
SingleLineEdit	sle	sleApellidoPaterno
StaticText	stx	stxTasaBancaria
Tab	tab	tabOpcion
TreeView	tre	treOrganizacion
UserObject	uob	uobMiControl
VScrollBar	vsb	vsbAngulo

10.3. Consideraciones para la Definición de Controles.

- No se deberá utilizar el símbolo de subrayado (_) para determinar el nombre de un control.

- Las tildes así como el uso de la letra "ñ" está prohibido en la determinación de nombres de controles.

- Se debe de tomar en cuenta las restricciones propias del lenguaje con respecto a la definición de nombres de controles.

- Todo control que haga referencia a campos en una base de datos deberá utilizar en su nombre el prefijo del control, y en la descripción obligatoriamente deberá contener el mismo nombre del campo al que hace referencia.

  Ejemplo:

Si en un control SingleLineEdit se quiere capturar o mostrar los valores del campo **clieNombre** que se encuentra en la tabla Clientes el nombre del control sería **txtClieNombre**

11. Definición de Menús.

Estructura	Prefijo	Nombre Descriptivo de la Variable
LONGITUD. MAX.	← 3 →	← ..... →
FORMATO	minúscula	Primer carácter de cada palabra en mayúscula.
EJEMPLO	mnu	Archivo ArcAbrir

En el proceso de creación de un Sistema de Menús se debe tener en cuenta las siguientes consideraciones (para la creación de un *Sistema de Menús* consideraremos que un *Menú* es un elemento de nivel superior que agrupa dentro de sí un conjunto de *Elementos de Menú*):

- Todo objeto Menu debe utilizar el prefijo **mnu**.

- Cada *Menú* debe tener un nombre que describa las tareas que desarrollan sus elementos.
  p.e. **mnu**Archivo

- Los nombres de los *Elementos de Menú* se determinan basándose en la abreviatura (3 caracteres) del nombre del *menú* al cual pertenecen (no tomar en cuenta los prefijos del mismo) y en el nombre que mejor identifique la tarea que cumple.
  p.e. **mnu**Archivo
  p.e. mnu**Arc**Abrir

- Si se cuenta con *submenús*, la asignación de los nombres para los *elementos del submenú* siguen el mismo criterio explicado anteriormente para la creación de elementos de menús para un menú.
  p.e.
  mnu*Archivo*
  mnu**Arc***Imprimir*
  mnuArc**Imp***AreaSeleccionada*
  mnuArc**Imp***Todo*

- Si se utilizan *separadores* dentro de los elementos de un menú o dentro de submenús, el nombre del mismo debe ser **mnuSeparador*n*** (donde n es el número de separador que se está creando).
  p.e. **mnu**Separador1

- Se prohibe el uso de *arreglos de menús* debido a cuestiones de mantenimiento de los mismos.

- Si es necesario el bloqueo de ciertas opciones de menú por cuestiones de seguridad, se deberá utilizar una base de datos que almacene la información necesaria que permita configurar el acceso a los diferentes elementos del sistema de menús.

12. Definición de Ventanas.

Estructura	Prefijo	Clase	[Modulo]	Descripción de Ventana
LONGITUD. MAX.	← 1 →	← 3 →	← 3 →	← 9 →
FORMATO	minúscula	Minúscula	mayúscula	Primer carácter de cada palabra en mayúscula.
EJEMPLO	w	Con	EFC	Transferencia

Para la nomenclatura de las ventanas se debe considerar lo siguiente:

- Prefijo; toda ventana debe contar con el prefijo **w**.
- Clase; determina la clase de ventana que se está nombrando así se tiene;

Clase	Descripción de Clase de Ventana
con	Consulta
lis	Lista
rep	Reporte
man	Mantenimiento
bus	Búsqueda
fil	Filtro
otr	De acuerdo al objetivo de la ventana

- Módulo; prefijo **opcional** que identifica al módulo o sub sistema de una aplicación; se usará generalmente para aplicaciones grandes en donde para cada ventana interese saber al módulo al cual pertenecen.

13. Definición de DataWindow.

Estructura	Prefijo	Tipo	[Modulo]	Descripción de Ventana
LONGITUD. MAX.	← 1 →	← 3 →	← 3 →	← 9 →
FORMATO	minúscula	Minúscula	mayúscula	Primer carácter de cada palabra en mayúscula.
EJEMPLO	d	grd	REP	DiariosElectronicos

Para la nomenclatura de las ventanas se debe considerar lo siguiente:

- Prefijo; todo objeto DataWindow debe contar con el prefijo **d**.
- Tipo; determina el tipo de DataWindow que se está nombrando así se tiene;

Tipo	Descripción del Tipo de DataWindow
frm	Formulario
grd	Grid
rep	Reporte
gra	Gráfico
ddw	DropDownDataWindow
crs	Crosstab
ext	External
com	Composite
grp	Group
ole	Ole
rch	RichText
tab	Tabular

- Módulo; prefijo **opcional** que identifica al módulo o sub sistema de una aplicación; se usará generalmente para aplicaciones grandes en donde para cada ventana interese saber al módulo al cual pertenecen.

14. Procedimientos y Funciones Definidos por el Usuario.

El nombre de las funciones y procedimientos debe ser autodescriptivo de manera que no se requiera, en lo posible, entrar al código de la función para saber qué es lo que realiza.

Estructura	Ámbito	Indicador Función o Procedimiento	Verbo (Acción)	Sustantivo
LONGITUD. MAX.	← 1 →	← 1 →	← . . . →	← . . . →
FORMATO	Minúscula	Minúscula	Primer carácter de cada palabra en mayúscula.	Primer carácter de cada palabra en mayúscula.
EJEMPLO	g	p	Asignar	Nombre

14.1. Ámbito

Ambito	Descripción
G	Público o global
P	Privado

14.2. Indicador Función/Procedimiento

Indicador	Descripción
F	Función
P	Procedimiento

14.3. Verbo-Sustantivo

El estándar para nombres de procedimiento es usar un Verbo que describa la acción realizada seguido por un sustantivo (objeto sobre el cual actúa). Se recomienda:

- Usar un nombre que represente una acción y un objeto. El nombre del procedimiento debe indicar qué hace el procedimiento a... o qué hace el procedimiento con....
- El verbo debe estar en infinitivo.
- Ser consistente en el orden de las palabras. Si se va a usar ***VerboNombre***, siempre usar *VerboNombre*.
- Ser consistente en los verbos y sustantivos usados. Por ejemplo si tiene un procedimiento **gp*AsignarNombre***, también usar **gp*AsignarApellido*** en vez de **gp*ColocarApellido*.**

- En el caso que el objeto al cual se refiere el verbo esta compuesto por más de una palabra, por ejemplo:
  Para la acción **actualizar los saldos del cliente** se define:
  gp*ActualizarSaldosCliente*
  Verbo: Actualizar
  Palabra Compuesta: SaldosCliente

Nota

- No se hará uso de los caracteres:

  - Espacio en blanco " "
  - Carácter de subrayado "_"

- La nomenclatura de argumentos o parámetros pasados a los procedimientos/funciones así como para valores devueltos por funciones sigue las mismas convenciones que la nomenclatura para variables.

15. Consideraciones para Power Script.

El lenguaje de PowerBuilder llamado PowerScript debe codificarse bajo estándares para que el código de los aplicativos puedan ser leídos y mantenidos por cualquier desarrollador.

15.1. Manejo de eventos

Las herramientas actuales proveen del uso de eventos, lo cual resulta mucho más simple de desarrollar, pero se deberá de tener en cuenta en que evento de los controles se va a codificar.

Por ejemplo al hacer click sobre un botón, el script debría llamar a un evento de usuario de la ventana donde reside el código del botón y manejarla allí. La primera funcionalidad a la ventana misma, es decir, todo los scripts estarían en eventos de ventana. Por ejemplo: En el evento Clicked del Botón Nuevo tiene el siguiente script:

```
Parent.PostEvent("gpAgregarElemento")
```

Para ello se deben de crear eventos definidos por el usuario en la ventana. De esta forma el mantenimiento del código es más fácil porque todo el código está ubicado dentro la ventana en lugar de estar repartido en varios objetos como botones.

15.2. Paso de Parámetros

Cuando es posible, los valores usados por una ventana debería ser pasado como parámetros, en lugar de usar variables globales. Evitar el uso de variables globales es muy importante para aplicaciones Cliente/Servidor manejadas por eventos, en los cuales los eventos pueden ejecutarse en un orden no anticipado. En conclusión, se recomienda usar la menor cantidad de variables globales.

El paso de parámetros usa el objeto message proporcionado por PowerBuilder. También se usan estructuras para pasar parámetros entre ventanas. Estas estructuras incluyen paso de parámetros genéricos (para objetos heterogéneos), parámetros específicos a ventanas y valores genéricos de retorno. Esto significa que se puede pasar cualquier número de parámetros de cualquier tipo sin herencia.

15.3. Variables globales

Para mantener el encapsulamiento de objetos y eliminar un efecto no esperado, se debería de trabajar con el menor número de variables globales.

Sin embargo, hay casos donde variables globales tiene sentido como código de usuario, nombre de ventana MDI, nombre de aplicación. Se deben tener como mínimos las siguientes variables globales:

- Código de usuario
- Nombre de usuario
- Código de rol

15.4. Uso de Mayúsculas y minúsculas

El uso de caracteres mayúscula y minúscula facilitan la comprensión del código de PowerScript porque cada elemento del código tiene una distinta apariencia visual.

- Las palabras reservadas de PowerScript aparecen todo en mayúscula.
  Ejemplo: IF, THEN, ELSE ,END IF, CASE)

- Funciones de PowerBuilder aparecen en la primera letra de cada palabra en mayúscula y el resto en minúscula (ejemplo: IsValid, Trim, IsNull, Mid, OpenSheet, Modify, etc)

- Palabras reservadas de SQL aparecen todo en mayúscula.
  Ejemplo : SELECT, INSERT, COMMIT, ROLLBACK, UPDATE)

- Las variables, funciones y objetos definidos por el usuario según estándares

- Los tipos de datos de PowerBuilder aparecen con la primera letra en mayúsculas y el resto en minúsculas. Ejemplo : Integer, String, Boolean, etc.

15.5. Indentación

Todo el código de PowerScript debería ser indentado para reflejar la estructura del código. Se puede usar para indentar la tecla tab por cada nivel de anidamiento. La continuación de líneas (es decir, líneas seguidas de un ampersand (&)) son indentadas con un sólo tab antes que la siguiente línea continue. Las líneas deberían ser partidos de tal manera que todo el código entre dentro de la pantalla debiendo de ser innecesario realizar scroll horizontal para ver todo el código.

15.6. Declaraciones de variables

PowerScript permite a las variables ser declaradas al inicio antes de su primer uso.
Cuando el script es muy corto la declaración de las variables puede ir en las primeras líneas, pero si el script es muy grande (es decir, más que una pantalla completa) se deben separar en bloques, cada uno de los cuales debe ejecutar una tarea especifica. Cada sección debe tener un bloque de comentario seguido de las declaraciones de variables usadas sola en esa sección seguidas por el código. Si se usan variables aplicables a todo el script se deben declarar en la parte superior.

15.7. Valores de retorno de Funciones

Las funciones de PowerBuilder que retornan valores enteros generalmente retornan 1 para indicar el éxito y un valor negativo para indicar fracaso. Funciones pueden retornar uno de varios valores negativos para indicar varios modos de fracaso. Se debe escribir las funciones de usuario de esta forma para asegurar que el código sea consistente con PowerBuilder.

15.8. Mensajes de Error

Los mensajes de error deben ser lo mas informátivo posible, ambos, el usuario y el desarrollador deben hacer uso de mensajes para eliminar defectos del aplicativo y siempre indicar el objeto o función que genera el error si es apropiado el script.

En el título de los mensajes de error se deberá utilizar la variable global gs_aplic_nombre.

Ejemplos :

- Código de documento no es válido
- Tipo de Saldo de cuenta no es negativo
- Debe ingresar la fecha del documento

15.9. Llamadas a Procedimientos Almacenados

Sintaxis para SQL Server

- **Declarar**
  DECLARE**<nombre proclogic>** FOR**<owner>.sp_<nombre>** **@<parametro1> = <valor1>, .. ;**

- **Ejecutar**
  EXECUTE **<nombre proc logic>**;

- **Recuperar parámetros de salida**
  FETCH **<nombre proc logic>** INTO **(<parametros salida>);**

- **Cerrar procedimiento**
  CLOSE **<nombre proc logic>**;

Donde :
Owner : usuario creador del procedimiento
sp : prefijo de procedimiento almacenado
nombre : nombre de procedimiento almacenada

15.10. Uso de Iconos

Los iconos que se colocan dentro de los distintos objetos de PowerBuilder al escribir el nombre del icono no se deben incluir la ruta completa porque los nombres de iconos al generarse el ejecutable se incluyen en una lista de iconos y también porque pueden modificarse la ruta donde están grabados los bmp.

El icono nuevo corresponde al archivo bmp "c:\bmp\nuevo.bmp" el cual al ubicarse dentro de un objeto como un menu o boton se debe escribir solo "nuevo.bmp". El desarrollador debe colocar en el archivo autoexec.bat la ruta completa donde están ubicados los bmp.

El icono seleccionado para cualquier función debe de ser lo suficientemente representativo, explicativo e intuitivo posible para la función elegida.

15.11. Distribución de Librerías

Los objetos de PowerBuilder residen en librerías los cuales están contenidos en archivos con extensión PBL. Por el gran número de objetos que se requieren para construir los aplicativos se requiere de una distribución de las librerías clasificadas por tipos de objetos para una rápida ubicación de todos los integrantes de desarrollo.

Recomendación: Las librerías PBL no deben pasar de un 1 MB de tamaño.

15.12. Documentación de PowerScript

Se deberán de documentar todas las funciones y eventos en todos los objetos y controles de PowerBuilder.

Funciones: (Globales, de Menús, de Ventanas, de UserObject)

```
////////////////////////////////////////////////////////////////////////////////////////////
// Función          : <nombre_función>
//
// Propósito        : <descripción de la función>
//
// Argumentos       : <argumento 1> { <descripción de argumento 1> }
//                    <argumento 2> { <descripción de argumento 2> }
//
// Retorna          : <tipo de dato>  { <descripcion de retorno> }
//
// Autor    : <Nombre del desarrollador> <dd/mm/yyyy>
//
// Modificaciones: <autor> <fecha> <descripción>
//
////////////////////////////////////////////////////////////////////////////////////////////
```

donde: tipo de alcance es: public, private,

Documentación de Ventana

El evento open de ventana se debe documentar de la siguiente forma:

```
////////////////////////////////////////////////////////////////////////////////////////////
// Ventana          : <nombre_ventana>
// Propósito        : <descripcion del propósito de la ventana>
//
// Argumentos:
//      Entrada     : <argumento 1>   <descripcion de argumento 1>
//      Salida      : <argumento 2>   <descripcion de argumento  2>
//
// Autor   : <desarrollador>   <dd/mm/yyyy>
//
// Modificaciones:
//  <autor>    <fecha>  <descripción de la modificación>
//
////////////////
```

Eventos de Usuario: (Para ventanas, controles, userobject, menús)

En caso de eventos de usuario se debe especificar el propósito, los argumentos de recuperación y el tipo de retorno que devuelve el evento.

```
//////////////////////////////////////////////////////////////////////////////////////////// Propósito
<descripcion del propósito de la ventana>
//
// Argumentos       : <argumento 1> { <descripcion de argumento 1> }
//                    <argumento 2> { <descripcion de argumento 2> }
//
// Retorno          : <tipo de dato>  { <descripcion de retorno> }
//
// Modificaciones:
```

```
//  <autor>.<fecha> <descripción de la modificación>
//
////////////////////////////////////////////////////////////////////////////////////////////////////
```

ANEXO 3: ESTÁNDARES DE PROGRAMACIÓN - ACTIVE SERVER PAGE

INDICE

16. INTRODUCCIÓN.

16.1. Objetivos.

- Reglamentar la documentación de programas para lograr un mejor entendimiento de los mismos.
- Mejorar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.
- Proporcionar los lineamientos para el desarrollo de aplicaciones web orientadas al uso de plataforma Microsoft.

17. NOMENCLATURA.

17.1. Del Entorno de la Aplicación - Estructura del Site.

Dado a que las aplicaciones web alojan directorios y páginas su administración y mantenimiento será más efectivo si cuentan con estándares en cuanto a nomenclatura.

17.1.1. Directorios Web.

Estructura	Mnemónico de Aplicación / Directorio	Descripción de Directorio
LONGITUD. MAX.	← 3 →	← 12 →
FORMATO	*minúscula*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*cia*	*includes*
EJEMPLO	*cia*	*images*

- **Mnemónico**; es el identificador de la aplicación si el directorio web es de primer nivel ó identificador de directorio en caso que el directorio web sea de mayor nivel de anidamiento.

- **Descripción**; proporciona una idea concreta del tipo de páginas que aloja el directorio.

17.1.2. Páginas Web.

Estructura	Mnemónico de Directorio	Descripción de Página
LONGITUD. MAX.	← 3 →	← 12 →
FORMATO	*minúscula*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*cia*	*Index.htm*
EJEMPLO	*man*	*RegistroUsuario.asp*

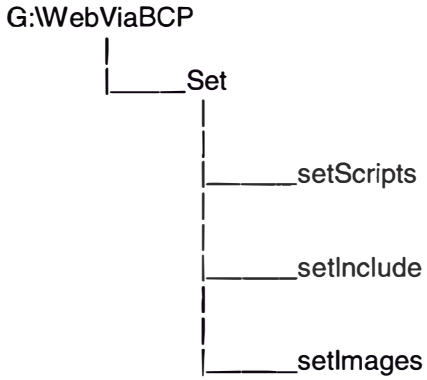
- **Mnemónico;** es el identificador del directorio web que aloja a la página en cuestión; su uso se hace necesario para identificar el módulo u subsistema al cual pertenece la página toda vez que pueden existir páginas con funcionalidad similar en directorios distintos.

- **Descripción;** proporciona una idea clara de la funcionalidad que realiza la página web cuando es usada.

17.1.3. Ejemplo de implementación de un web site.

- **Objetivo**; Se quiere crear una aplicación Web de un curso de explicación del protocolo SET, esta aplicación se incluirá dentro del WebSite de ViaBCP.

- **Solución;** Dado que físicamente el WebSite de ViaBCP se encuentra en el disco g:\WebViaBCP, las paginas de ejemplo, ira dentro de g:\WebViaBCP, la estructura de nuestra aplicación Web debería respetar la siguiente estructura.



Donde "Set" es la carpeta principal en donde se ubica la pagina inicial "default.asp o index.htm" y también el archivo de inicialización del aplicativo, en este caso el archivo globa.asa.
Dentro de la carpeta "scripts" se incluye las librerías de código Javascript con extensión *.js.
Dentro de la carpeta include irían los archivos Include, usados en las paginas html o asp.
Finalmente en la carpeta images, los gráficos conformantes del site.

18. Declaración de Variables.

Se propone la utilización de variables que mejor describan su función para el completo entendimiento del programador. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente

- La longitud debe ser lo más recomendable posible, no debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 13 caracteres.
- El ámbito en que desenvolverán las variables.
- El tipo de dato al que pertenece la variable.

Por lo tanto la estructura de la variable es como sigue:

Estructura	Ámbito	Tipo de Dato	Descripción de la Variable
LONGITUD. MAX.	← 1 →	← 3 →	← 9 →
FORMATO	*Minúscula*	*minúscula*	*Primer carácter de cada palabra en mayúscula.*
EJEMPLO	*g*	*Int*	*NumCta*

18.1. Ámbito.

Describe el ámbito en que la variable trabajará, además también sirve para definir constantes y variables que serán usadas como parámetros en procedimientos y funciones, este debe estar en minúscula. El caracter corresponde a la tabla siguiente.

g	Variable Global, es reconocida en cualquier parte de la aplicación.
p	Variable Privada, es reconocida solamente por el modulo en la que es declarada.
l	Variable Local, es reconocida tan solo en el Procedimiento o Función en la que fue declarada, al terminar este, la variable es destruida.
c	Constante
x	Variable de Parámetro

18.2. Tipo de Dato.

Define el tipo de dato al que pertenece la variable. Los mnemotécnicos corresponden a:

Mnemotécnico	Tipo de Dato
bln	Boolean
byt	Byte
col	Colección de Objetos
cur	Currency
dte	Date
tme	Time
dbl	Double
err	Error
int	Integer
lng	Long
obj	Object
Sng	Single
str	String
Udt	Tipo definido por el usuario.
ptr	Puntero
Vnt	Variant
Udc	Instancia de una clase definida por el usuario.

18.3. Descripción de la Variable.

Combinación de palabras que mejor describa la función de la variable, en esta combinación debe de comenzar con mayúscula y por cada cambio de palabras descriptivas volver a comenzar con mayúscula, cada palabra que se utilice no debe exceder los tres dígitos.

Ejemplos
gstrNomCli
pintConArt
cintIntAnu

18.4. Variables de Tipo Arreglo.

En el caso de las definiciones de arreglos de elementos, entonces la estructura de la variable cambiara en la adición de una "a" (minúscula) entre el primer dígito (tipo de variable) y la definición de tipo.

Ejemplos
gaintNumCli
pastrMnuPri

18.5. Variables con Referencia a Base de Datos.

Debemos también contemplar, que para el caso de variables que hagan referencia a campos en una base de datos, la estructura del prefijo del

mnemotécnico será la descrita anteriormente, pero la descripción de la variable, obligatoriamente deberá contener el mismo nombre del campo al que hace referencia. Es para este caso en el que el limite máximo del tamaño establecido para una variable (13 caracteres), puede ser excedido.

Ejemplos
gstrNombreCliente
pintCostoUnitario
lbooSexo

ANEXO 4: ESTÁNDARES DE PROGRAMACIÓN - SQL SERVER

INDICE

19. Objetivos.

- Estandarizar la nomenclatura de objetos de base de datos.
- Estandarizar la estructura de codificación del SQL Transact.

20. Normas Generales de nomenclatura para Base de Datos

- Mantener nombres cortos y descriptivos.
- Mantener nombres de objetos únicos, por ejemplo evitar crear la tabla VENTAS y un rol o vista con el mismo nombre.
- Por defecto, no se aceptan espacios en blanco en medio de los identificadores; sin embargo, su uso está permitido si se usan identificadores delimitados por comillas dobles. En el presente estándar, no se permiten los espacios en blanco como parte de un identificador.
- Para la definición de nombre de objetos de base de datos de acuerdo al caso se usará el caracter underscore "_" para separar las palabras_del_nombre.

21. Estándar de configuración de instalación de Base de Datos.

- **Character Set:** 850 Multilingual (Conjunto de caracteres válidos: 256)
- **Sort Order:** Binary Order, Case Sensitive.

22. Base de Datos

Nomenclatura:
El nombre de la base de datos debe estar en **mayúsculas.**

Ejemplo :

Base de Datos	:	SIGSC
Archivo MDF	:	SIGSC_Data
Archivo LDF	:	SIGSC_Log

23. Definición de Variables, parámetros y tipos de datos .

Aplica a:
- Stored procedures
- Triggers

Nomenclatura :
@xxxYYYY...

xxx : Identificador del tipo de dato
YYYY: Nombre de la Variable

Ejemplo: chrCODEMP, vchNOMBRE

Tipode Dato	Identificador
Bit	Bit
Int	Int
Smallint	Smi
Tinyint	Tni
Decimal	Dec
Numeric	Num
Money	Mny
Smallmoney	Smm
Float	Flt
Real	Rea
Datetime	Dtm
Smalldatetime	Sdt
Cursor	Cur
Timestamp	Tst
Uniqueidentifier	Uid
Char	Chr
Varchar	Vch
Text	Txt
Nchar	Nch
Nvarchar	Nvc
Ntext	Ntx
Binary	Bin
Varbinary	Vbn
Image	Img

24. Tablas.

Nomenclatura:
El nombre de la tabla debe ser descriptivo, en singular y en mayúsculas.
Las Tablas identifican una entidad del Sistema con un nombre completo.
Una tabla hija debe llevar el nombre de la tabla padre.
Las palabras deben ser separadas por un "_".

Ejemplo :

ASIENTOS
ASIENTOS_DETALLES

Los nombres de las **columnas** ser descriptivos y en mayúsculas.

Ejemplo :

```
CREATE TABLE EMPLEADO
(
        CODEMPLEADO     numeric(3, 0) NOT NULL ,
        NOMBRE          varchar (100) NULL ,
        DOCUMENTO       numeric(1, 0) NULL
)
```

Nota : Cuando se creen tablas temporales añadir el prefijo TEMP_XXXX para reconocerlas.

25. Constraints.

- **Nomenclatura Primary Key:** PK_*NombreTabla*
- **Nomenclatura Foreign Key:** FK_*NombreTablaOrigen_NombreTablaReferenciada*
- **Unique:** UQ_*NemónicoTabla_NombreUnique*
- **Default:** DF_*NemónicoTabla_NombreColumna*
- **Check:** CK_*NemonicoTabla_NombreCheck*

Ejemplo:

PK_CLIENTE
FK_FACTURA_CLIENTE
UQ_CLIENTE_CODIGO_CLIENTE
DF_CLIENTE_FECHA
CK_CLIENTE_CODIGO_CLIENTE

8. Indices.

Nomenclatura para los Indices :

I_XXX_ YYYY....

XXX : Tipo de índice(Unique, Clustered, NonClustered)
YYY : Nombre del Indice

Ejemplo:

IAK_CODEMPLEADO	(índice Unique)
ICL_CCUSTODIA	(índice Clustered)
INCL_CODDEPOSITANTE	(NonClustered)

9. Triggers.

Nomenclatura para los Triggers :

TR_NOMBRETABLE_D (Cuando se realiza una eliminación en la tabla).

TR_NOMBRETABLA_U (Cuando se realiza una actualización en la tabla).

TR_NOMBRETABLA_I (Cuando se realiza una inserción en la tabla).

TR_NOMBRETABLA_IU (Cuando se realiza una inserción o actualización en la tabla).

10. Stored Procedures.

Nomenclatura para los Stored Procedures :

Insert:
NombreTabla_Add (Insertaun registro a la tabla)

Update:
NombreTabla_Update (Actualiza un registro a la tabla)

Delete:
NombreTabla_Delete (Elimina un registro de la tabla)

Select:
NombreTabla_GetByID (Retorna un sólo registro por PK o AK).
NombreTabla_ListByXXXX (Retornan más de un registro)
NombreTabla_ListXXXX

Nota: No todos los stored procedures siguen esta convención, dado que algunos abstraen un mayor nivel de funcionalidad como ejecutar un proceso o algún query complejo.

Ejemplo :

EMPLEADO_Add
EMPLEADO _Insert
EMPLEADO _Update
EMPLEADO _GetByID
EMPLEADO _ListByEmpresa
EMPLEADO _ProcesaSueldo

Nota:

- Los nombres de los Stored Procedures **NO** deben comenzar con sp, esto porque generalmente el SQL piensa que son system procedures y los busca primero en la Base de Datos master

SET NOCOUNT ON(elimina la notificación del nro. de registros afectados por cada sentencia SQL lo cual incrementa el performance.

Estructura del Stored Procedure:

- Identificador, nombre de stored procedure., y parámetros
- Comentarios:
  - Descripción: "funcionalidad del stored procedure"
  - Descripción de Parámetros de entrada y salida
  - Autor:
  - Fecha Modificación:
  - Versión:
  - Cambios Importantes

- Declaración Variables locales
  - Mayúsculas y minúsculas
- Sentencias SQL
  - Palabras del lenguaje SQL, y funciones de sistema en MAYUSCULAS, columnas y otras variables en Mayúsculas.
  - Sentencias legibles e indentadas (cada clausula SQL en una línea nueva)

Ejemplo:

```
CREATE PROCEDURE EMPRESA_Add(
        @intCODEMPRESA int,
        @vchRAZONSOCIAL varchar(50),
        ...)
AS
/**************************************
*Descripcion: Añade un registro a la tabla HUB
*Fecha Crea: 19/02/2001
*Fecha Mod: 19/02/2001
*Parametros:    @CODEMPRESA :    Código de la empresa
*               @RAZONSOCIAL:        Razon Social de la Empresa
*               ...
*Autor: Juan Perez (14201)
*Versión: Final (Beta|Final)
*Cambios Importantes: Inclusión de la condición se consulta(15/02/2001)
*/**************************************
<Declaración de variables>
<Sentencias SQL>
```

11. Vistas.

Nomenclatura para las Vistas:

VW_YYY....

VW : Identificador para las Vistas
YYY : Nombre de la Vista en Mayúsculas

Ejemplo :
VW_CONSULTAR_PERSONERIA

Para el nombre de las Vistas utilizar verbos en Infinitvo,

Ejemplo :
VW_CONSULTAR_CUENTAS

Nota: La estructura de la vista debe ser similar a la del stored procedure.

12. Programación.

Utilizar mayúsculas para las sentencias propias del SQL

Ejemplo .-

```
SELECT      NumeroDocumento,
            TipoDocumento ,
            ApellidoPaterno,
            ApellidoMaterno,
            Nombre
FROM        DDEPOSITANTE
ORDER BY    NumeroDocumento
```

Utilizar el Tabulador para separar los campos de una condición (en la medida de lo posible)

Ejemplo .-

```
SELECT  'CodigoSegmentoSAB'          =    CodigoDepositante,
        'CodigoSegmentoCONASEV'      =    '00' +SUBSTRING(NumeroRegistro,2,1),
        'CodigoRetorno'              =    '0'
FROM    DDEPOSITANTE
WHERE   NumeroDocumento              =    @chrNumeroDocumento
AND     TipoDocumento                =    @chrTipoDocumento
AND     RelacionadorCorrelativo      =    @chrRelacionadorCorrelativo
```

Indentar el Código para conservar un orden

Ejemplo.-

```
CREATE PROCEDURE BUSCARCADENA
    (       @vchVariable    VARCHAR(255),
            @vchTipo        VARCHAR(1) =""
    )
AS
BEGIN
    IF LTRIM(RTRIM(@vchVariable)) <> "*"
            IF @vchTipo = ""
                    SELECT      NOMBRE      = name
                                TIPO        = type    ,
                                CREACION    = crdate
                    FROM        sysobjects
                    WHERE       name LIKE '%'+ @vchVariable + '%'
                    ORDER BY    type,
                                name
            ELSE

                    SELECT      NOMBRE      = name  ,
                                TIPO        = type
                                CREACION    = crdate
```

```
                FROM            sysobjects
                WHERE           name LIKE '%'+ @vchVariable + '%'
                AND             type = RTRIM(LTRIM(@vchTipo))
                ORDER BY name
END
```

ANEXO 5

CONEXIÓN A BASE DE DATOS Y HOST

POLITICAS ESPECIFICAS

1.1. Condiciones

1.1.1. Conexión a Bases de Datos

a) La forma estándar de acceder a bases de datos SQL Server desde aplicaciones desarrolladas en herramientas Microsoft será ADO con OLEDB.

b) La forma estándar de acceder a bases de datos SQL Server desde aplicaciones desarrolladas en Power Builder será empleando los drivers nativos de la herramienta.

c) La versión estándar de ADO a emplear es la que viene incluida en el MDAC 2.5 SP 2. Esta es la versión de MDAC que trae Windows 2000 SP2, pero se puede instalar también en Windows NT y Windows 95.

d) Sólo se deberá emplear los instaladores de MDAC proporcionados a través de los canales regulares.

1.1.2. Conexión al Host

a) COMTI será el medio estándar para la conexión al Host desde aplicaciones cliente/servidor desarrolladas con herramientas Microsoft.

b) Al hacer uso de COMTI se recomendará invocar transacciones en lugar de hacer link a programas.

c) Para las aplicaciones distribuidas en agencias se aceptará MCS Access como excepción.

d) Se mantendrá como opción válida el uso del APIs de SNA Server Client para aquellas aplicaciones que las vienen utilizando.

e) Se encuentra en evaluación MQ Series para utilizarlo en el futuro como herramienta de integración de plataformas.

1.2. Restricciones

a) No está permitido copiar nuevas versiones de MDAC de Internet o de cualquier otra fuente extra oficial.

1.3. Descripciones

a) MDAC (Microsoft Data Access Components) es un producto que incluye diferentes librerías para el acceso a datos, entre ellas ADO y OLEDB.

b) OLEDB es una interfaz estándar de bajo nivel a todo tipo de data. Está diseñada para proveer un acceso a datos universal en distintas plataformas.

c) ADO (ActiveX Data Objects) es una interfaz de alto nivel que permite utilizar los servicios de OLEDB de un modo mucho mas fácil a través de un modelo de objetos.

ANEXO 6: DOCUMENTACIÓN DE PROGRAMAS

INDICE

Documentación de programas

26. Objetivos.

- Reglamentar la documentación de programas para lograr un mejor entendimiento de los mismos.
- Mejorar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

27. Recomendaciones.

La documentación de programas debe realizarse durante la codificación del mismo, para que no se convierta en una tarea adicional que requiera tiempo extra del programador, evitando que deje de hacerse.

El control de la documentación se realizará durante la etapa de certificación de la calidad de la programación, por parte del supervisor, considerando los siguientes lineamientos y recomendaciones.

27.1. En cuanto a Legibilidad.

- Un comentario siempre debe preceder el código que describe. El lector debe ser capaz de analizar sólo los comentarios y obtener una buena idea de lo que hace el código y dónde buscar una actividad específica.
- Documentar código inesperado. Hay casos en que se emplean trucos para mejorar la performance en ciertas operaciones, de ser así usar comentarios para señalar que se está usando un truco para mejorar la performance a costa de la legibilidad.
- Evitar abreviaturas.
- El comentario debe ser lo menos ambiguo y lo más legible posible.
- Los comentarios no pueden rescatar código difícil. Una buena regla a seguir aquí es: "no comente mal código, re escríbalo". Esto es útil en particular cuando recién se están cambiando programas. Si sólo se están documentando y no se puede cambiar el código, pues simplemente coméntelo.
- Utilice espacios en blanco para incrementar la legibilidad.

27.2. En cuanto a Abundancia.

- Usar comentarios para preparar al lector acerca de lo que sigue.
- En lugar de gastar tiempo escribiendo demasiados comentarios, mejor es poner el esfuerzo extra en hacer el código más legible.
- La cantidad de comentarios debe de ser moderada, sin importar aspectos como la "cantidad", se debe de tener en cuenta que la "calidad" de representación del comentario es parte de los objetivos para una buena documentación de código.
- Una línea de comentario por cada diez líneas de código ejecutable es un buen promedio, pudiendo verse afectada por casos particulares.
- Coloque más comentarios de los que piense que son necesarios sin abusar de ellos.

27.3. Expresar lo que se quiere.

- Los comentarios deberían aportar información adicional, no parafrasear lo que se escribe en el código.

- Los comentarios que explican el cómo, operan al nivel del lenguaje de programación y no al nivel del problema. A este tipo de comentarios le resulta casi imposible describir la intención del código.

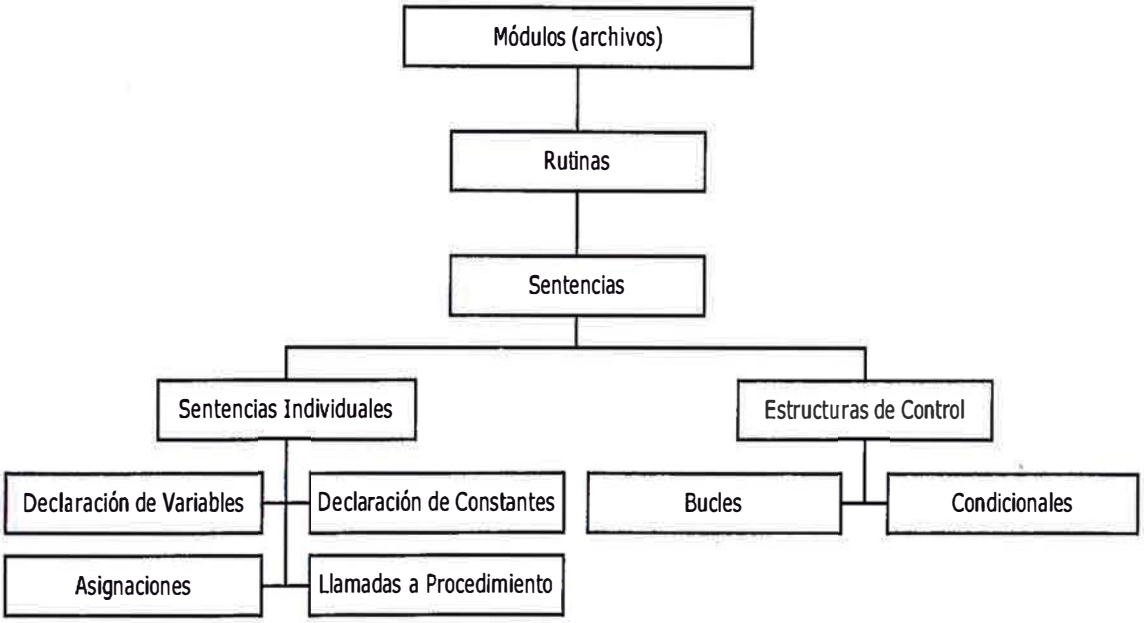
- Un comentario incorrecto es peor que la ausencia de comentario.

- Los comentarios no deberían nunca oscurecer el código.

28. Comentarios a partir de Niveles de Código.

La documentación de programas debe realizarse dentro del código fuente.

El enfoque seguido para documentar los programas, ha sido establecer correspondencia entre los comentarios y los niveles de código de los mismos:



A continuación se describen las técnicas para el uso de comentarios por cada nivel de código, empezando por el nivel Superior.

Para ello, en la presente propuesta planteamos el uso de una sintaxis para los comentarios correspondientes para los diferentes niveles:

28.1. Sintaxis para el uso de comentarios.

Por motivos de estandarización proponemos la siguiente sintaxis de comentado de instrucciones:

- **Símbolo de inicio de comentarios**: El indicador de inicio de comentario estará representado en la propuesta con un "asterisco entre paréntesis" (*).

(*) (Primer comentario) ; (Segundo comentario)

Para el caso de comentarios por bloque se utilizará la siguiente nomenclatura:

(**) (Inicio de bloque de comentario)
(Fin de bloque de comentario) (**)

➢ **Obligados:** Todo comentario de carácter obligatorio deberá de ir entre paréntesis.

(Ejemplo de comentario obligatorio)

➢ **Opcionales:** Todo comentario de carácter opcional deberá de ir entre corchetes.

[Ejemplo de comentario opcional]

Nota: Cabe resaltar que la obligatoriedad y la opcionalidad de los comentarios no está definida en razón al criterio del programador, sino en razón a su presencia en las sentencias correspondientes.

➢ **Separador de comentarios:** Para establecer la separación de dos comentarios ya sea opcional u obligatorio se deberá utilizar un "punto y coma" (;).

(Primer comentario) ; (Segundo comentario)

28.2. Documentación en Módulos.

➢ El desarrollo de rutinas debe estar concentrado en un solo módulo donde se especificará el propósito del mismo.

➢ Si es política de la Empresa, se puede especificar un comentario de Derechos de Propiedad.

➢ Cada archivo debe representar una funcionalidad, es decir un módulo.

➢ La funcionalidad del módulo debe hacerse en un máximo de dos líneas.

➢ La documentación del modulo en la medida de las posibilidades del lenguaje utilizar comentarios en bloque.

➢ Debe insertarse el siguiente marcador al inicio y al final del prólogo del módulo:

#M#:

Este marcador se ha contemplado para su posible uso a través de un compilador de comentarios, el cual facilite la labor de clasificación, catalogación, navegación en el código fuente, de módulos y rutinas. Además, podríamos beneficiarnos con la creación de un repositorio de módulos y rutinas para llevar el control de todos sus cambios.

Descripción de las partes de la documentación del módulo:

Nombre: Nombre del módulo expresado en forma literal.

Autor(es): Autor(es) del módulo.

Modificado por: Autor(es) de la última modificación del módulo.

Última Modificación: Fecha de última actualización del módulo o de alguna rutina que lo componga. El formato a utilizar será: (SSAA-MM-DD).

Número de Actualizaciones: Numeral que se incrementara por cada modificación que afecte a alguna rutina y por consiguiente al módulo.

Referencia al Diseño: Referencia al diseño(dependiendo si existe dicha documentación).

Objetivo: Objetivo que indicará de manera literal y concreta la finalidad del módulo.

Funcionalidad: Descripción de la funcionalidad del módulo (dicha descripción no excederá a dos líneas de comentarios).

Entradas: Descripción literal de los ingresos que recibe el módulo.

Salidas: Descripción literal de las salidas del módulo.

Variables Globales: Descripción literal de las variables globales usadas en el módulo respectivo.

Sintaxis:

Lenguajes que permiten comentarios de bloque	**Lenguajes que no permiten comentarios de bloque**
(**)(#M#)	**(*)(#M#)**
(Nombre)	(*) *(Nombre)*
(Autor(es))	(*) *(Autor(es))*
[Modificado por]	(*) *[Modificado por]*
(Última Actualización)	(*) *(Última Actualización)*
[Número de Actualizaciones]	(*) *[Número de Actualizaciones]*
[Referencia al Diseño]	(*) *[Referencia al Diseño]*
(Objetivo)	(*) *(Objetivo)*
(Funcionalidad)	(*) *(Funcionalidad)*
(Entradas)	(*) *(Entradas)*
(Salidas)	(*) *(Salidas)*
[Variables Globales]	(*) *[Variables Globales]*
(#M#)(**)	(*)(#M#)(*)

Nota: Se recomienda en la medida de lo posible utilizar comentarios de bloque.

28.3. Documentación de Rutinas.

- Los comentarios deben mantenerse en la parte superior de cada rutina.
- Las variables que maneja la rutina no deben documentarse cuando estas sean demasiado obvias.
- Se debe resaltar (según el entorno de programación) los comentarios a realizarse en la cabecera de rutina.
- La "Functionality" de la rutina debe hacerse en un máximo de dos líneas.
- Debe insertarse el siguiente marcador al inicio y al final del prólogo de la rutina: #R#.

Descripción de las partes de la documentación de la rutina:

Nombre: Nombre del módulo expresado en forma literal.

Autor(es): Autor(es) de la rutina.

Modificado por: Autor(es) de la última actualización del módulo.

Ultima Actualización: Fecha de última actualización en el formato (SSAA-MM-DD).

Número de Actualizaciones: Numeral que se incrementara por cada modificación que afecte a alguna rutina y por consiguiente al módulo.

Objetivo: Objetivo que indicará de manera literal y concreta la finalidad del módulo.

ParamVal: Enumeración y descripción de los parámetros por valor utilizados en la rutina.

ParamRef: Enumeración y descripción de los Parámetros por Referencia utilizados en la rutina.

ParamOut: Enumeración y descripción de los parámetros "OUTPUT" que recibe la rutina.

Data Externa: Enumeración y descripción de los datos externos utilizados en la rutina.

GlobalVar: Enumeración y descripción de las variables globales usadas en el desarrollo de la rutina.

Valor de Retorno: Descripción del valor de retorno de la rutina.

Funcionalidad: Descripción literal del proceso de la rutina.

Sintaxis:

Lenguajes que permiten comentarios de bloque

(**)(#R#)
(Nombre)
(Autor(es))
[Modificado por]
(Última Actualización)
[Conteo de Actualizaciones]
[Referencia al Diseño]
(Objetivo)
(Funcionalidad)

[ParamVal]
[ParamRef]
[ParamOut]
[Data Externa]
[GlobalVar]
[Valor de Retorno]
(#R#)(**)

Lenguajes que no permiten comentarios de bloque

(*)(#R#)
(*) (Nombre)
(*) (Autor(es))
(*) [Modificado por]
(*) (Última Actualización)
(*) [Conteo de Actualizaciones]
(*) [Referencia al Diseño]
(*) (Objetivo)
(*) (Funcionalidad)

(*) [ParamVal]
(*) [ParamRef]
(*) [ParamOut]
(*) [Data Externa]
(*) [GlobalVar]
(*) [Valor de Retorno
(*)(#R#)(*)

Nota: Se recomienda en la medida de lo posible utilizar comentarios de bloque.

28.4. Documentación en Sentencias Individuales.

En la Documentación de Sentencias individuales planteamos una serie de puntos iniciales para entender mejor la propuesta de este documento.

Se debe documentar:

- La sentencia individual que suficientemente complicada como para necesitar una explicación.
- La sentencia individual que tuvo alguna vez un error y se quiere llevar el rastro del error.

Situaciones especiales:

- Los comentarios de fin de línea deben ser alineados a la derecha del código, para no interferir con la estructura visual del mismo.
- En nuestra propuesta sólo se aplicará para las declaraciones de variables, constantes, marcar bucles anidados y sentencias nulas.

Se considera como sentencia individual a las siguientes sentencias:

28.4.1. Declaración de Variables.

Se debe declarar una variable por línea. El comentario debe ir al final de línea si el lenguaje de programación lo permite y guardar concordancia con la variable. Además se debe considerar:

- UNIDADES DE DATOS: las unidades de medida deben ser comentadas, por ejemplo: pulgadas, metros, kilómetros etc., no deben de ser asumidas como obvias.
- RANGOS DE VALORES NUMÉRICOS: se deben comentar los rangos (mínimo y máximo) de los valores posibles que van a tomar las variables.
- TIPOS ENUMERADOS DE DATOS: se pueden definir valores específicos para una variable, es decir los valores y su significado, siempre y cuando el lenguaje de programación lo permita. Estos tipos enumerados deben ser comentados.
- FLAGS A NIVEL BIT: se debe documentar las variables declaradas para ser usadas como campos booleanos, con valores de 0 y 1.
- VARIABLES GLOBALES: se debe especificar un comentario donde haya sido declarada la variable, indicando el propósito de la variable, y porque ésta necesita ser global.
- DECLARACION DE PARAMETROS : se debe documentar los parámetros utilizados en las funciones.

Sintaxis

VARIABLE (*) (Propósito de la Variable) ; [Unidad de la Variable] ;
(*) [Rango de Valores] ; [Valores Correspondientes a la
(*) Enumeración] ; [Flags a Nivel de Bit] ; [Justificación
(*) de Variable Global]

(Tipo Parametro) PARAMETRO (*) (Propósito del Parámetro) ;
(*) [Unidad del parámetro] ;
(*) [Rango de Valores] ;
(*) [Valores Correspondientes a la
(*) Enumeración] ;
(*)[Flags a Nivel de Bit]

Los valores de retorno de la función tendrán las mismas características que las variables excepto por el propósito y la justificación de variable global.

Ejemplo1: Declaración de variables

```
Dim intCounter as Integer ' Contador para recorrer los arreglos;
```

Ejemplo2: En el caso de declaración de parámetros

```
Function CalcularArea ( _
   ByVal dblLargo _        ' largo del terreno en metros
   ByVal dblAncho _        ' ancho del terreno en metros
) As Double                ' valor en metros cuadrados
```

28.4.2. Declaración de Constantes.

- Se debe declarar una constante por línea.
- El comentario deberá ir al final de la línea, si el lenguaje de programación lo permite.

Sintaxis

CONSTANTE (*) (Propósito de la Constante) ; [Unidad de la
(*) Constante] ; [Flags a Nivel de Bit]

Ejemplo:

Const dblIGV as double' Representa el Impuesto General a las Ventas;

28.4.3. Asignaciones.

- Se debe describir el propósito(sólo asignaciones relevantes)

Sintaxis

(*) (Propósito de la Asignación)

ASIGNACION

Ejemplo

' El valor del sobregiro es el monto excedido por la tasa de sobregiro
intSobreGiro = (intSaldo – intImporte) * dblTasaSobreGiro

28.4.4. Llamadas a Rutinas.

- Se debe describir el propósito.
- Descripción de cada parámetro de entrada.
- Descripción del valor de retorno esperado.

Sintaxis

(*)(Propósito de la Llamada);[Parámetros que se Envía a la Rutina]; (*)[Descripción del Valor de Retorno Esperado]

Ejemplo

'Obtener el código del cliente a partir de su nombre; strCliente
'contiene el nombre del cliente;
intObtenerCodigo = ObtenerCodigo(strCliente)

28.5. Documentación de Sentencias de Control.

28.5.1. Bucles.

En la Documentación de Sentencias individuales planteamos una serie de puntos iniciales para entender mejor la propuesta de este documento.

Para un bucle o iteración, se debe indicar el propósito del mismo, el porque del valor inicial y final del bucle, los motivos de los posibles quiebres o salidas del bucle (break,exit loop,etc) y las razones de la condicional del bucle si en caso existiese.

- Se debe documentar las sentencias nulas.
- Se recomienda documentar el fin de un bucle muy largo o anidado.

Sintaxis

(*) (Propósito del Bucle) ; (Razón del Valor Inicia) ; (Razón de valor

(*) Final) ; (Razón de la Condicional de Fin) ; [Quiebres] ; [Sentencias

(*) Nulas]

INICIO DEL BUCLE

SENTENCIAS INTERNAS

_____

_____

FIN DE BUCLE

Ejemplo
'Buscar los clientes que tienen saldo negativo ;
'Inicio en el primer registro ; Termina en fin de archivo
rsClientes .MoveFirst
While not rstClientes.Eof
....
Wend

28.5.2. Condicionales

Para una sentencia condicional se debe documentar la razón para la condición y un resumen del desenlace.

Sintaxis

(*) (Propósito de la condición) ; (Resumen del Desenlace)

INICIO DE LA CONDICIONAL

SENTENCIAS INTERNAS

_____

_____

_____

FIN DE LA CONDICIONAL

Ejemplo

'Detectar si el saldo es negativo para determinar si está sobregirado
If intSaldo < 0 Then
...
End If

29. Beneficios de una Buena Documentación.

- La documentación hace más legible un programa.

- Al documentar bien un programa desde el principio, se evita que para cada modificación deba estudiarse profundamente el funcionamiento del programa re descubriendo todo lo no documentado, con la desventaja adicional de que generalmente quien modifica el programa no es siempre quien lo escribió.

- Facilita la reutilización de módulos y rutinas desde cualquier otro programa o el mismo.

- Facilita el posterior entendimiento del programa.

- Ayuda a determinar cuando debe ser re escrito un código. Si existen problemas para explicar el código con un comentario, probablemente el código esté mal escrito. En otro caso, al registrar los cambios en el prólogo de la rutina, puede determinarse que, si hay demasiados cambios en una rutina, lo más probable es que ésta deba ser re escrita.

- Facilita la detección de módulos con cohesión débil. Al poner un comentario acerca de lo que se supone que debe hacer un módulo e indicar las cabeceras de las rutinas que usan, puede determinarse si dichas rutinas están en concordancia con el propósito del módulo.

- Facilita la detección de la frecuencia de uso de variables globales en las rutinas.

- Con una documentación bien hecha el lector experimentará una sensación agradable, una invitación a estudiar en detalle el código escrito.

30. Conclusiones.

- Una buena documentación sólo se logra si se escribe con el programa, lo que requiere de una modesta cantidad de disciplina. Esta se logra con la práctica y con el deseo de lograr buenos resultados en nuestro trabajo. Es muy importante que antes de escribir un procedimiento, el programador escriba su especificación. El ver la documentación como algo "extra" al programa es un error que generalmente tiene un alto valor.

- El escribir la documentación al hacer el programa se logrará un programa sintácticamente correcto y claramente escrito.

- Al documentar correctamente lograremos un mejor programa, simplemente porque un programa sin documentación no es un programa completo.

ANEXO 7: MEJORES PRÁCTICAS PARA VISUAL BASIC, ASP Y SQL SERVER

Mejores prácticas para Visual Basic, ASP y SQL Server

Contenido

Introducción

Este documento es un documento de mejores prácticas para el área de sistemas de la Empresa. Abarca cuatro áreas:

- El modelo de componentes
- Las mejores practicas para codificar en Visual Basic
- Las mejores practicas para páginas ASP (algunas de estas practicas también pueden ser aplicadas para páginas HTML).
- Mejores prácticas para el acceso y la manipulación de la data.

Además se incluye un anexo con normas de programación útiles en el contexto de la Empresa.

El propósito de este documento es brindarle a los programadores una guía rápida de consulta, no para explicar en detalle el funcionamiento de las diversas las herramientas y servicios que se mencionan. Para mayor información se incluye al final de este documento algunas referencias útiles.

Diseño del Modelo de objetos (Windows DNA).

Modelo de objetos

Las aplicaciones se desarrollaran empleando el modelo Windows DNA, este modelo divide la lógica de las aplicaciones en tres capas estas capas son:

- Capa de Servicios de Presentación
- Capa de Servicios de Negocios.
- Capa de Data

Cada una de estas capas se encarga de las siguientes tareas.
La **Capa de Servicios de Presentación** es responsable de:

- Recoger la información del usuario
- Enviar los información del usuario a los servicios de negocio para su procesamiento
- Recibir los resultados del procesamiento hecho en los servicios de negocios.
- Presentar estos resultados al usuario.
- Generalmente esta capa es una página ASP, o un programa Visual Basic con interfase de usuario.

La **Capa de Servicios de Negocios** es responsable de:

- Recibir lo enviado por la capa de presentación.
- Interactuar con los servicios de Data para ejecutar las *operaciones del negocio* para la cual la aplicación fue diseñada (por ejemplo; el cálculo de un préstamo, el importe de un impuesto, procesar una planilla, etc.)
- Enviar los resultados procesados a la capa de presentación.
- Esta capa es, en general, un componente COM corriendo bajo MTS o COM+

La **Capa de Data** es responsable de:

- Recibir los pedidos de data desde la capa de Servicios de Negocio
- El almacenamiento de la data.
- La recuperación de la data.
- El mantenimiento de la data
- La integridad de la Data.
- Enviar la data pedida a los servicios de negocios.
- Esta capa se puede dividir en dos:
  - Un componente COM que se encarga de acceder a la data, es decir el que llama al Stored Procedure por ejemplo.
  - Un componente que esta en el servidor en sí, en el ejemplo anterior es un Stored Procedure.

Como reglas se puede entonces tener que:

- **Nunca la capa de servicios de presentación accede a la data**. Es decir NUNCA una página ASP llama a un Stored Procedure.

- **Nunca la capa de servicios de negocio accede a la data**. Es decir NUNCA una componente de la lógica de negocios crea un RecordSet desde una tabla

- **Se debe tratar siempre de usar Stored Procedures,** estos son los que acceden, agregan, modifican y borran la data.

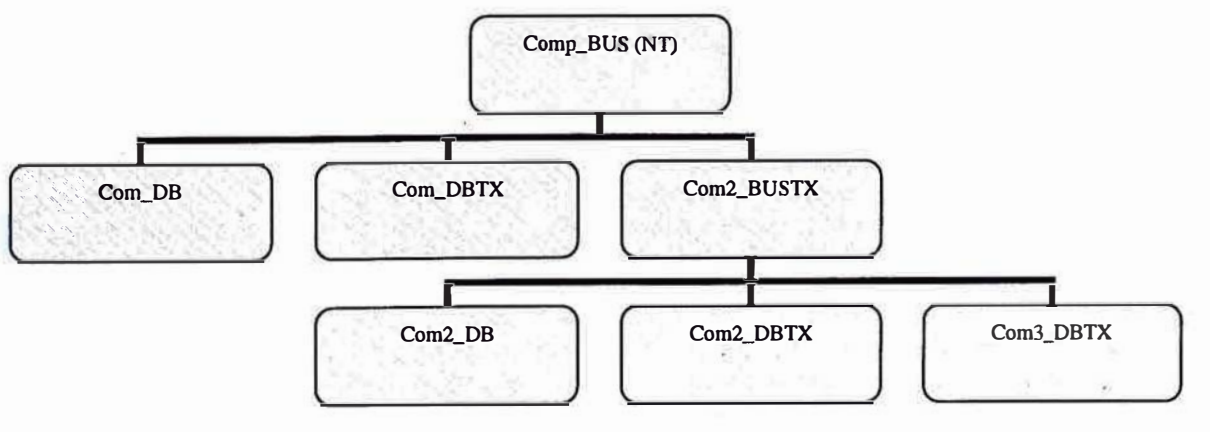
Además como nomenclatura para los componentes utilizaremos como sufijo BUS y DB para señalar si el componente es de la capa de negocios o de la campa de data. Así: TLC_Cliente_BUS, es un componente de la capa de negocios y TLC_Cliente_DB es un componente de la capa de data.

Modelo Transaccional

Se deben seguir los siguientes lineamientos:

1) Se considera componentes transaccionales a todos aquellos componentes que agregan, modifican o eliminan información.
2) Los componentes transaccionales serán registrados en MTS con:
   a. *Requires Transaction* si el componente participa en una transacción, es decir es llamado desde otro componente transaccional.
   b. o *Requieres New Transaction*, si el componente inicia una transacción.
3) Todos los componentes que no sean transaccionales se registrarán con *Not Supported.*
4) Generalmente sólo son transaccionales componentes de la capa de data. Sin embargo componentes de la capa de negocios que enlistan a varios componentes transaccionales de la capa de data, u otros componentes transaccionales de la capa de negocios también deben ser transaccionales.

Como nomenclatura para los componentes transaccionales utilizaremos como sufijo BUSTX y DBTX para señalar que el componente es transaccional, así TLC_Cliente_BUSTX es un componente transaccional de la capa de negocios.



En el esquema anterior:

- Los componentes Comp_BUS, Com_DB, y Com2_DB no son transaccionales.

- Com_DBTX, Com2_BUSTX, Com2_DBTX y Com3_DBTX son transaccionales.

- Siendo Comp2_BusTX un componente de la capa de negocios que enlista dentro de su transacción a los componentes de la capa de datos Com2_DB, Com2_DBTX y Com3_DBTX si cualquier de estos componentes aborta la transacción se abortará cualquier cambio realizado en los demás componentes.

- Los componentes Comp_BUS, Com_DB, y Com2_DB se declaran con Not Supported.

- Los componentes Com_DBTX, Com2_BUSTX se declaran con Requires New transacction

- Los componentes Com2_DBTX y Com3_DBTX se declaran con Requieres Transacction

NOTA **En MTS se debe crear el componente que es enlistado dentro de otra transacción utilizando la función CreateInstance y NO CreateObject.**

Implantación

Cuando los distintos componentes se implantan, es conveniente declarar a los componentes BUS como componentes de tipo SERVER. Para hacer esto en el browser de MTS seleccionar el paquete que contiene los componentes BUS, seleccionar properties, seleccionar el tab activation y allí seleccionar la opción Server application.

Para los componentes DB se debe, por lo general, seleccionar Library application.

Esto es porque los componentes de manejo de data se pueden ejecutar en el mismo contexto de memoria que los componentes de lógica de negocio, y no en contextos separados. Con esto se logra un mejor rendimiento.

Mejoras en programación de Visual Basic

Los principales puntos a verificar en la codificación de componentes y aplicaciones Visual Basic se exponen a continuación. El uso consistente de estas técnicas de programación traerá como resultado aplicaciones y componentes más rápidos, más escalables y más estables.

No usar variables de tipo Variant

El tipo por defecto de las variables en Visual Basic es Variant. Esto es útil para los programadores principiantes y para las aplicaciones donde la velocidad no es un factor importante. *Cuando se trata de aplicaciones y componentes reales donde la velocidad si juega un rol determínate se debe evitar a toda costa el uso de Variants.* La perdida en velocidad es causada porque Visual Basic convierte las variables Variant al tipo apropiado en tiempo de ejecución. El uso del tipo apropiado de variables elimina este paso extra y son más rápidas.

Además Visual Basic asume las variables no declaradas como Variants, para evitar esto se debe usar la sentencia **Option Explicit,** que fuerza al programador a declarar todas sus variables. Existen dos maneras de usar Option Explicit, la primera es escribir la sentencia como primera línea de todos los módulos, la segunda es seleccionar la opción Require Variable Declaration en el Tab Editor del dialogo Options disponible en el menú Tools.

Se debe tener cuidado cuando se declara múltiples variable. Si no se utiliza la cláusula *As* las variables serán declaradas como variants.

Por ejemplo, en la siguiente declaración las variables X e Y son Variants.

```
Dim X, Y, Z As Long
```

Para que las tres variables sean de tipo Long se debe escribir la sentencia como sigue:

```
Dim X As Long, Y As Long, Z As Long
```

Usar variables Integer y Long integer para cálculos matemáticos.
Para operaciones aritméticas se debe evitar usar los tipos de variable *Currency, Single y Double.* Use el tipo *Long* cada vez que pueda especialmente en los ciclos (loops). El tipo Long es el tipo nativo de datos para CPUs de 32 bits, por esto las operaciones con ellos son muy rápidas. Si no se puede usar el tipo Long se pueden usar los tipos Integer y Byte. En muchos casos se puede usar enteros largos (long) en vez de variables de punto flotante. Por ejemplo, si se selecciona el valor twips o pixels de la propiedad ScaleMode en todas las formas y controles, se pude usar enteros largos para los valores de posición y tamaño del control y de los métodos gráficos.
Cuando se realizan divisiones se debe tratar de usar el operador de división entera (\) si no se necesita un resultado con decimales. El cálculo con enteros es siempre más rápido que el cálculo con valores de punto flotante ya que las operaciones son realizadas directamente por la CPU y no por el coprocesador matemático.
Si se necesita realizar cálculos con valores decimales se debe usar el tipo *double* en vez del tipo *Currency.*
La siguiente tabla muestra los tipos numéricos con relación a su velocidad de ejecución.

Tipo Numérico	Velocidad
Long	El más rápido
Integer	
Byte	
Single	
Double	
Currency	El más lento

Guardar las propiedades usadas frecuentemente en un cache.
Es más rápido cambiar el valor de una variable que el de una propiedad Si está obteniendo el valor de un propiedad muy frecuentemente (como por ejemplo en un ciclo) el código correrá mucho más rápido si se asigna el valor de esta propiedad fuera del ciclo y luego se utiliza el valor de la variable en vez de la propiedad. Por lo general las variables son entre 10 a 20 veces más rápidas que las propiedades.
No se debe acceder al valor de una propiedad más de una vez en un procedimiento al menos que el valor haya cambiado. En vez de esto asigne el valor de la propiedad a una variable y usa esta en vez de la propiedad. Por ejemplo:
Método **muy lento:**

```
For i = 0 To 10
    picIcon(i).Left = picPallete.Left
Next I
```

Método **mucho más rápido**:

```
picLeft = picPallete.Left
For i = 0 To 10
    picIcon(i).Left = picLeft
Next I
```

Método **muy lento:**

```
Do Until EOF(F)
    Line Input #F, nextLine
    Text1.Text = Text1.Text + nextLine
Loop
```

Método **mucho más rápido**:

```
Do Until EOF(F)
    Line Input #F, nextLine
    bufferVar = bufferVar & nextLine & vbCrLf
Loop
Text1.Text = bufferVar
```

Sin embargo el código siguiente realiza la misma operación que el código anterior y es aún más rápido:

```
Text1.Text = Input(F, LOF(F))
```

Como se puede apreciar siempre existen distintos métodos para realizar la misma tarea, *el mejor algoritmo es generalmente la mejor optimización.*

Guardar los resultados de funciones en un cache.

La técnica de guardar los valores de una propiedad en una variable también se puede utilizar con los valores retornados por las funciones, guardar estos en variables evita llamadas frecuentes a la librería de Visual Basic (Msvbvm60.dll), por ejemplo:

Método **muy lento:**

```
dblPi = 3.1415
x = 0
For i = 0 To 1000
    x = i * abs(dblPi)
Next I
```

Método **mucho más rápido**:

```
dblPi = 3.1415
intAbsPi = abs(dblPi)
x = 0
For i = 0 To 1000
    x = i * intAbsPi
Next I
```

Usar variable a nivel de módulos en vez de variable estáticas (Static)

Declarar variables como Static es muy útil cuando se necesita almacenar un valor entre las distintas ejecuciones de un procedimiento. Almacenar el mismo valor en una variable de nivel de modulo, es decir una variable almacenada fuera de todo procedimiento, se ejecutará de forma mucho más rápida. Se debe tener en cuenta, sin embargo, que solo un procedimiento debe alterar el valor de la variable. La concesión por la cual se opta es que el código será menos legible y más difícil de mantener.

NOTA **Los componentes desarrollados para COM NO DEBEN usar la sentencia *Static,* ni usar variables a nivel de modulo.**

Remplazar llamados a procedimientos por código en línea.
Si bien usar procedimientos hace al código más modular, las llamadas al procedimiento siempre agrega tiempo y trabajo en el momento de la ejecución. Si un ciclo llama muchas veces a un procedimiento es mejor copiar el código del procedimiento dentro del ciclo evitando así la llamada.

NOTA **Sin embargo, si el código debe ser copiado en varios ciclos el tamaño de la aplicación aumenta, además se hace más difícil de mantener. Se debe mantener un equilibrio entre tener el código de un procedimiento directamente en línea, con la posibilidad de tenerlo en un procedimiento separado.**

De la misma forma llamar a un procedimiento que reside en el mismo modulo es más rápido que llamar al mismo modulo que reside en un módulo .BAS separado. Si el mismo procedimiento debe ser llamado desde diversos modulo esta ventaja no existe.
Por ejemplo, método **lento:**

```
For i = 0 To 1000
    b = b + FuncLlamada(i)
Next I

Public Function FuncLlamada(intP1ra1 as Integer) as Integer
    FuncLlamada = intP1 * intP1
End Sub
```

Método **más rápido**:

```
For i = 0 To 1000
    b =  b + (i * i)
Next I
```

Usar constantes cada vez que sea posible.
Usara constante hace más rápida la aplicación. Las constantes hacen, además, más legible el código y más fácil de mantener. Si la aplicación, o el componente, tiene números o cadenas que no cambian se deben declarar como constantes. Las constantes son resueltas a la hora de la compilación con los valores apropiados insertados en el código. El valor de las variables es resuelto en tiempo de ejecución lo cual es bastante más lento.
Cada vez que sea posible se deben usar las constantes listadas en el Object Browser en vez de crear nuevas. Se puede incluir módulos que contengan constantes que no van a ser usadas pues cuando la aplicación se compile estas no formaran parte del ejecutable.
Pasar los argumentos usando ByVal en vez de ByRef.
Cundo los procedimientos reciben parámetros que no van a ser modificados es más rápido pasar los argumentos por valor (ByVal) que pasarlos por referencia (ByRef). En Visual Basic los argumentos son ByRef por omisión pero pocos procedimientos modifican los valores de sus argumentos. Si no se necesita modificar los argumentos dentro del procedimiento se debe usar ByVal, por ejemplo:

```
Private Sub HacerHalgo(ByVal strNombre As String, ByVal intEdad As Integer)
```

NOTA **En caso de los componentes de una aplicación COM no se debe utilizar NUNCA la cláusula ByRef, sobretodo si la aplicación va a ser implementada en un entorno donde existan Firewalls entre los distintos servidores.**

Asignar un tipo a los argumentos opcionales.
En versiones anteriores de Visual Basic los argumentos opcionales eran siempre de tipo variant, en la versión actual (6.0) se puede aumentar la velocidad de ejecución especificando el tipo de cada uno de los argumentos opcionales.
Si un procedimiento tiene argumentos pasados ByVal, como en el ejemplo siguiente, se usarán 16 bytes del stack para pasar el argumento de tipo variant

```
Private Sub HacerHalgo(ByVal strNombre As String, _
Optional ByVal vntEdad As Variant, _
Optional ByVal vntPeso As Variant)
```

Se empleará menos espacio del stack por llamada, y menos data será movida en la memoria, si se codifica la misma función asignando un tipo a los argumentos opcionales:

```
Private Sub HacerHalgo(ByVal strNombre As String, _
Optional ByVal intEdad As Integer, _
Optional ByVal IntPeso  As Integer)
```

El acceso a un argumento opcional con un tipo definido es más rápido y además brinda errores de compilación si se envía información con un tipo de dato incorrecto.
Se debe usar OLEDB.
OLEDB es 33% más rápido que ODBC, por esto se debe tratar de usar cada vez que sea posible.
Funciones de manejos de String
Las funciones: Chr, Format, LCase, Left, LTrim, Mid, Right, RTrim, Space, String, Trim, UCase, encargadas de manejar cadenas de caracteres pueden utilizar el prefijo $ con lo cual declaran que el parámetro que están recibiendo es de tipo String y no Variant. Esto permite que la función se ejecute con mayor rapidez. Ej.:

```
strPrueba1 = Mid(strNombreUsuario,1,5)
```

Es menos eficiente que:

```
strPrueba1 = Mid$(strNombreUsuario,1,5)
```

Manejo de errores
La estructuras de manejos de errores son variadas se propone la siguiente para mantener consistencia en todas las aplicaciones

```
Private function fucntion1() as string 'Esta es la función o procedimiento
Dim strPrueba1 as String
On Error Goto ErrHandler
    'Aquí se escribe el código de la función
```

```
Exit:
    'Aquí se escribe el código para terminar la función
    'blanqueo de variables, etc.
    strPrueba1 = "".
    Exit Function              'En el caso de procedimientos es Exit Sub
ErrHandler:
    'Aquí va el manejo de errores
    Goto Exit
End Function
```

Uso de Mid$

Mid$ es una función que puede ayudar a escribir algoritmos de concatenación de cadena de caracteres más rápidos. Ej.:

```
strBuffer = ""
strString1 = "Hola"
strString2 = "Chau"
strBuffer = strBuffer & strString1
strBuffer = strBuffer & strString2
```

Esto funciona bien pero el siguiente código es más eficiente:

```
strBuffer = Space$(9)
strString1 = "Hola"
strString2 = "Chau"
Mid$(strBuffer,1,4) = strString1
Mid$( strBuffer,5,5) = strString2
```

En este caso la ganancia de velocidad es mínima pero cuando la concatenación está dentro de un ciclo y este se repite muchas veces la ganancia pude ser de varios minutos. Este tipo de manejo también aplica para manejo de buffers y estructuras similares

Configuración del compilador de Visual Basic para crear componentes COM eficientes.

Se debe tener en cuenta las siguientes opciones en las propiedades de proyecto (Menú Project, Project Properties), para crear componentes que se ejecuten de manera eficiente.

1) En el tab *General* se debe seleccionar:
    a. **Unattended Execution,** ya que los componentes se ejecutan de manera desatendida, sin interacción del usuario. Además esto marca al componente como thread-safe, es decir utilizara el modelo de apartamento para el threading.
    b. **Retained in memory,** con esta opción se le indica a MTS o COM+ de mantener el componente el tiempo más largo que pueda sin descargarlo, este acelera el tiempo requerido para cargar el componente, ya que gran parte de las veces ya se encuentra en memoria.
    c. **Se debe seleccionar Apartment Threaded como Threading Model.**
2) En el tab *Compile* se debe seleccionar:

a. **Optimize for fast code,** pues siempre se van a ejecutar muchos componentes en memoria es mejor que se ejecuten rápido.
b. **Favor Pentium Pro**, todos los procesadores Pentiums desde la versión II son compatibles con Pentium Pro, y seleccionando esta opción los componentes se ejecutaran más rápido.
c. **Se debe cambiar el DLL Base Address.** Es necesario saber que la primera vez que se cargue la DLL se tratará de cargar en esta dirección de memoria. Por omisión todas las DLLs creadas con Visual Basic tratan de cargarse en la dirección &H11000000 lo que hace más lento el proceso de carga de la DLL pues muchas veces esta dirección de memoria se encuentra ocupada por otra DLL creada con Visual Basic.

Páginas ASP y HTML

- **Las páginas enviadas deben pesar menos de 100 Kb.**

- **El tamaño de cualquier archivo que pueda ser bajado debe ser claramente informado**, de ser posible.

- **Se debe tratar de no usar "frames",** de no ser posible se debe tratar de tener una versión "sin frames" actualizada. Aproximadamente un 13% de las visitas usan browsers que no soportan frames, además los incapacitados físicos no pueden usarlas con facilidad. La funcionalidad de Back, Forward e impresión son difíciles de manejar con frames y es difícil que los usuarios creen bookmarks o enlaces a información que se encuentra dentro de un frame (El ejemplo más clásico es un bookmark a un producto en ViaCompras, se puede hacer, pero cuando volvemos al producto deseado no se puede comprar porque falta un frame)

- **Los enlaces de navegación deben ser colocados siempre en le mismo lugar en todas las páginas**, además deben ser etiquetados consistentemente.

- **Los enlaces de navegación deben no sólo deben estar al final de la página sino también en otro fácilmente asequible (a la izquierda, arriba o a la derecha)**

- **No se debe obligar al visitante a navegar por páginas que solo contienen enlaces de navegación.**

- **Se deben proveer etiquetas de texto par todos los botones y enlaces gráficos** (algunos usuarios deshabilitan las imágenes para tener una navegación más rápida)

- **Se deben evitar los callejones sin salida**, debe al menos existir un enlace a la página principal en todas las páginas.

- **El alto y ancho de todas las imagines debe ser especificado**. Además se debe proveer un texto en el tag ALT de todas las imágenes. Cuando el gráfico es un bullet se debe usar "*" (asterisco) para el tag ALT. Esto se hace para un rápido despliegue de los elementos de la página, que no tienen que esperar a bajar la imagen o el bullet. De no

hacerlo el motor de rendering (el encargado de mostrar la página) debe esperar por la imagen para saber donde va el texto en la página.

- **Para escalar a múltiples servidores las variables de sesión deben ser eliminadas** para, de esta forma, tener aplicaciones stateless (sin estado). Se debe utilizar software como Windows Load Balancing Service o hardware como LocalDirector de Cisco.

- **Se debe usar SSL solo donde es necesario.** SSL es una operación costosa. Por ejemplo se puede poner SSL el las páginas POST del formulario de registro del usuario, pero no en la página cuando viaja sin información dentro de ella. Es decir se debe usar SSL cuando se debe proteger algo.

- **Los nombres de archivos deben ser continuos** (se debe usar "_" y no espacios para cortar palabras) y deben incluir una extensión de tres letras.

- **Se recomienda incluir tags de búsqueda en todas las páginas**.

- **El objeto session debe ser deshabilitado en el servidor**, para mejorar el rendimiento.

- **Se debe obtener los recursos tarde y liberarlos temprano:** Se debe crear los componentes cuando se necesitan y liberarlos (igualándolos a Nothing) los más rápido posible.

- **Se debe tratar de no mezclar código ASP <%=...%> con código HTML:** es mejor tenerlo en bloques separados.

- **Todas las páginas ASP deben incluir "Option Explicit":** De esta manera se comprueba que todas las variables han sido definidas, el código se ejecutará más rápido y será más fácil de leer.

- **No se deben usar colecciones (collections).**

- **Se debe tratar de usar una solo línea par el Dim de las variables:** Visual Basic Scripting Edition (VBScript) interpreta todas las líneas, por esto colocándolas todo en un solo lugar aumentará la velocidad de ejecución

- **Se debe tratar de no redimensionar (redim) los arreglos:** Esta operación consume mucho tiempo. Como estrategia se puede usar la del peor escenario y diseñar los arreglos al tamaño más grande esperado.

- **No se deben declarar las variables como "Public".**

- **Se debe tratar de realizar validaciones básicas del lado del browser:** Se debe tratar de nunca llegar a llamar a la base de datos con información incorrecta, pues esto es un muy costoso. Realizar validaciones del lado del cliente (browser) descarga al servido del uso de

CPU, pudiendo de esta manera atender otras operaciones, se evitan, además viajes innecesarios al servidor sólo para realizar una validación básica (tipo del dato, largo del campo, que el dato se encuentra dentro de un rango, etc.). Pero se debe tratar de mantener un balance entre las validaciones del lado cliente y el lado servidor, pues si la validación del lado cliente viaja como parte de la página y entra dentro de los 100 Kb de la primera regla.

- **No se debe tratar de no usar variables del servidor (Server Variables).** Cuando se accede a una variable del servidor, el servidor Web pide TODAS las variables al servidor y no solo la que se requirió, esto es largo, pero una vez realizada la operación los siguientes llamados son rápidos.

Mejores prácticas en la capa de data
Componentes de acceso a Data.

- Cuando el componente no necesite más una conexión esta debe ser cerrada.

- No se deben reutilizar conexiones. Una conexión es para acceder siempre a la misma información, si se necesita acceder a información distinta se debe utilizar una conexión distinta.

SQL Server

- Siempre se debe especificar las columnas de la sentencia Select, NUNCA se debe usar SELECT * FROM nombre_de_tabla, al menos que realmente se necesiten todas las columnas.

- Los Stored Procedures son la manera más eficiente de manipular (seleccionar, actualizar, insertar, borrar) la data ya que residen en el servidor y este puede optimizarlas para un mejor rendimiento en tiempo de ejecución. Además envían menos data a través de la red.

- Se debe utilizar RowCount (SET ROWCOUNT) y/o la clausula WHERE para retornar el número mínimo de filas necesarias por cada consulta. Cuando se necesitan demasiadas filas se debe considerar diseñar las paginas ASP con un mecanismo de barrido (por páginas, por ejemplo).

Por ejemplo si un Stored Procedure necesita retornar más de 50 líneas, el componente puede pasar un par de parámetros al Stored Procedure; @PrimeraRegistro, y @CantidadDeRegistros. El parámetro CantidadDeRegistros asegura que solo se van a procesar una cantidad de registros fijos y el parámetro PrimerRegistro indica desde cual registro se va a comenzar. Esto brinda un control completo sobre el rendimiento de los componentes y páginas, ya que si el proceso es lento se puede disminuir la cantidad de registros a leer.

- Las columnas de la clausula WHERE deben ser, si es posible, indexadas, y NUNCA se debe hace un join en columnas no indexadas.

- Se debe usar SQL Enterprise Manager para ver el plan de ejecución (query plan) y las estadísticas de E/S (statistics I/O). El plan de ejecución muestra si SQL usa los índices o realiza un barrido de tabla (table scan)

para responder a una consulta. Las estadísticas de E/S muestran el número de lecturas lógicas (desde el cache en memoria) y las lecturas físicas (desde el disco). Esta información puede ser muy útil para diagnosticar problemas de velocidad de las consultas, consultas que sin razón aparente se ejecutan muy despacio.

- Cuando SQL realiza un barrido de tabla la ejecución comienza en el primer registro de la tabla, cada registro es recuperado y comparado con la condición especificada en la cláusula WHERE y retornado si cumple con el criterio de selección. Independientemente de los registros retornados todos los registros son consultados, por esto un barrido de tablas puede ser muy costos en termino lectura y escrituras de páginas.

- Si una tabla posee uno o más índices, el optimizador de consultas (query optimizer) puede optar por no utilizar ninguno de ellos y realizar una barrida de tabla si es que el uso de los índices es muy costoso o inútil para la consulta.

- Al usar un índice no se requiere la recuperación de todos los registros de la tabla (excepto si la cláusula WHERE aplica a todos los registros). Para consultas que retornan un porcentaje pequeño de una tabla grande las ganancias en término de E/S por usar índices, en vez de barrer toda la tabla, pueden ser muy importantes.

- Se pueden crear muchos índices en tablas que sólo se leen, en tablas de lectura-escritura se debe tratar de tener el menor número posible.

- Se deben distribuir los logs y las bases de datos en distintos discos, para que no interfieran uno con otro.

- Se debe normalizar la base de datos basado en el rendimiento no en la pureza del modelo.

- Se debe tratar, en la medida de lo posible de manejar tablas con pocas columnas y que estas columnas sean cortas.

- Se debe tratar de evitar columnas de ancho variable. Una columna de ancho variable es aquella que permite NULLs o cuando la columna es de tipo varbinary o varchar.

- Se debe archivar la información vieja (información poco, o nada consultada) para de esta forma mantener pequeña la base de datos, y por lo tanto más eficiente.

Anexo 1: Estándares de programación en Visual Basic
Definición de objetos.
Usaremos los siguientes prefijos para los objetos significativos, el prefijo constará de tres letras minúsculas que representan al control, excepto para los componentes del proyecto donde el prefijo será una letra mayúscula; seguido irá la descripción de dicho objeto con la primera letra en mayúscula.

Componentes del proyecto.

Objeto	Prefijo	Ejemplo

Clase	C	CCuenta
User Control	U	UmoduloCobranzas
Data Report	D	Dtransferencias
Form	F	FSaldos
Módulo	M	Mcobranzas
Multiple Document Interface	MDI	MDIPagosMasivos
Property Page	P	Pusuarios
Web Class	W	WEditText

Controles.

Objeto	Prefijo	Ejemplo
3D Panel	pnl	PnlControl
Animated Button	ani	AniAceptar
Check Box	chk	ChkFiltrarTipo
Combo Box	cmb	CmbTipoCuenta
Command Button	cmd	CmdAceptar
Common Dialog	dlg	DlgAbrirCuenta
Communications	com	ComFax
Control	ctr	CtrActual
Data	dat	DatBiblioteca
Data Bound Combo Box	cbo	CboAnexo
Data Bound Grid	dbg	DbgResultadoBusqueda
Data Bound List Box	dbl	DblTipoTrabajo
Data List	dlt	DltCuentas
Data Picker	dtp	DtpPublicado
Data Repeater	drp	DrpLocalizacion
DataCombo	dtc	DtcCuentas
DataGrid	dgd	DgdSaldos
Directory list box	dir	DirOrigen
Drive List Box	dlb	DlbDestino
EditText	txt	TxtMonto
File List Box	flb	FlbOrigen
Flat Scroll Bar	fsb	FsbMover
Frame	fra	FraOpciones
Gauge	gau	GauEstado
Graph	gra	GraTotales
Grid	grd	GrdPrecios
Hierarchical Flexgrid	flx	FlxOrdenes
Horizontal scroll Bar	hsb	HsbPantalla1

Image	img	ImgLogo
Image Combo	imc	ImcProducto
Imagen List	ils	IlsIconos
Label	lbl	LblNumeroCuenta
Line	lin	LinSeparacion
List Box	lst	LstCuentas
List View	lvw	LvwCabecera
MAPI Session	mps	MpsSesion
MAPI message	mpm	MpmEviarMensaje
MCI	mci	MciVideo
Menú	mnu	MnuCobranzas
Month View	mvw	MvwPeriodo
MS Chart	ch	ChVentas
MS Tab	mst	MstInicio
Msflexgrid	flg	FlgCuentas
OLE Container	ole	OleWorksheet
Option button	opt	OptSoles
Picture Box	pic	PicVGA
Picture Clip	pcl	PclToolbar
ProgressBar	prg	PrgLoadFile
Remote Data	rdt	RdtTitulos
RichTextBox	rtb	RtbReporte
Shape	shp	ShpOvalo
Slider	sld	SldEscala
Spin	spn	SpnPaginas
Sstab	sta	StaConfiguracion
Status Bar	stb	StbCuentas
SysInfo	sys	SysMemoria
TabStrip	tab	TabOpciones
Text Box	tbx	TxtEstado
Timer	tmr	TmrHora
Toolbar	tlb	TlbCobranzas
TreeView	tre	TreOrganizacion
UpDown	upd	UpdDireccion
Vscrollbar	vsb	VsbPantalla2

Objetos de Base de Datos

Objeto	Prefijo	Ejemplo
Command	cmn	cmn
Connection	cnx	cnx
Database	dtb	dtb

DBEngine	dbe	dbe
Err	err	err
Field	fld	fld
Parameter	prm	prm
QueryDef	qry	qry
RecordSet	rst	rst
TableDef	tbd	tbd
Workspace	wrk	wrk

Variables

Las reglas que usaremos para identificar una variable son:

- Debe comenzar con una letra
- No puede incluir un punto o un carácter de declaración de tipo ($,&,!,%,#,@).
- Debe ser única en el mismo alcance.
- Se utilizará "Option Explicit" por defecto.
- Las variables y constantes globales preferentemente deben ser definidas en un solo módulo BAS. Por ejemplo: constantes.bas, valores.bas.

Alcance de vida de la variable

Alcance	Prefijo
Global	G
Nivel de modulo	M

Tipo de dato de la variable

Tipo de dato	Nemónico
Bolean	bln
Class	cls
Collection	col
Currency	cur
Date	dat
Double	dbl
Integer	int
Long	lng
Object	obj
Single	sng
String	str
User defined type	udt
Variant	Var

Nomenclatura y Declaración

El nomenclatura de una variable será:

Prefijo_alcance + prefijo_tipo de dato + descripción

La descripción brindará un nombre inteligible a la variable (NombreDelProveedor, FechaDeNacimiento, FechaDeAfiliación, Etc.) Se deberán tener en cuenta las siguientes reglas:

- Cada variable debe ser declarada en una línea diferente.
- Solo la primera letra de cada palabra se escribirá en mayúsculas.

- No se usaran espacios, ni guiones(_), ni guines bajos (_), solo caracteres y números.

Por ejemplo: gintSaldoCuentaProveedor
Es una variable con alcance global, de tipo entero (integer) donde sospechamos se lleva el ssaldo de la cuenta de un proveedor.

Constantes

El nombre de una constante sigue las mismas reglas que los nombres de las variables, con la diferencia que todo va en mayúscula, y no se necesita comenzar con una letra que defina el tipo de dato.
Para declarar una constante simbólica (aquella que se utilizará una y otra vez en el código), se utilizará la palabra const, además de las mismas reglas que se aplican a las variables.
Una constante será declarada de la siguiente forma:
[Public|Private] Const *nombre_constante* As [tipo] = *expresión*
Los prefijos de los alcances son:

Alcance	Prefijo
Global	GC
Modular	MC

Nomenclatura y Declaración

La nomenclatura par referenciar a una constante será:
ALCANCE + '_' + DESCRIPCIÓN
El tipo de datos de una constante es el mismo que el usado para la variable.
Por ejemplo: GC_MAXIMOARREGLOPROVEEDOR
Define una constante de alcance global que define un número máximo de un arreglo de proveedores.

Comentarios

Serán aquellas frases que vayan precedidas de una comilla simple (') , en consecuencia no se ejecutará acción alguna.

- Serán claros y específicos evitando redundancias
- Toda la frase estará escrita en minúscula.
- Se ubicarán al lado de la sentencia de código a la cual hace referencia, pudiendo ocupar más de una línea.
- En el caso en que se realice una modificación que este asociada a un ticket, colocar el número de ticket dentro del comentario, el cual hará referencia a una especificación:

Nro. Ticket:	
Nro. Requerimiento:	
Responsable:	
Fecha:	
Hora:	
Descripción:	

Procedimientos y funciones.
Los nombres de procedimientos se deben escribir en mayúsculas y minúsculas y debe tener la longitud necesaria para describir su funcionalidad. Además los nombres de funciones deben comenzar con un verbo.
Se usará el prefijo "g" en caso la función o procedimiento sea global.
La estructura de un procedimiento o función será la siguiente:
[Static] [Private|Public] [Function|Sub] *Nombre* ([Parámetros]) As [tipo]
Dentro de la función o procedimiento se especificará las siguientes partes.

```
'Autor:
'Fecha de última modificación:
'Funcionalidad:
'Argumentos de entrada:
'Argumentos de salida:

Inicio del cuerpo del programa

[sentencias]

End [Function|Sub]
```

Parámetros del procedimiento o función
Además de su tipo los Parámetros de los procedimientos o funciones tendrán los siguientes prefijos:

Tipo de parámetro	Prefijo	Ejemplo
De Entrada	I	istrCodigo
De Salida	O	ointSaldo
De Entrada y Salida	Io	iodoubleSueldo

Nomenclatura y Declaración
La nomenclatura par referenciar a una función o procedimiento será:
Verbo infinitivo + sujeto en el que se actúa (comienza con letra mayúscula).
Por ejemplo: AbrirArchivoDePlanillas
Es una función (o procedimiento) que muy probablemente abre un archivo de planillas.
Niveles de anidamiento
Para los niveles de anidamiento se usará indentación de una tabulación (tab).

Estructura condicionales

```
If control.enable  = true then
    Negrita =1
Else
    Negrita =0
Endif
```

Condiciones múltiples

```
Select Case B
    Case 0
            Print "Hola"
    Case 1
            A= 10
End select
```

Estructura de repetición

```
For i=1 to 20
    For j=2 to 21
            Print numero
    Next j
Next i
```

```
Do while saludo = ""
    Saludo= "Hola"
Loop
```

```
Do Until saludo= "Fin"
    Saludo= "Hola"
Loop
```

Nombre de archivos
Para los nombres de los archivos se usará el siguiente formato:
Modulo que pertenece + Acción + extensión
Ejemplo: Si el archivo pertenece a la funcionalidad de registrar transferencias dentro del módulo transferencias se tendrá:
transferencia_registrar.frm
Formato de plantilla de especificación de procedimientos o funciones.
Para cada procedimiento o función se debe entregar un documento con la siguiente información:

Módulo	{nombre del Modulo}
Programa	{nombre del programa}
Fecha Inicio	{fecha de inicio}
Fecha de termino	{fecha de término}
Fecha de entrega	{fecha de entrega}
Nombre del Diseñador	{nombre del diseñador}
Nombre del Programador	{nombre del programador}
Nivel de Dificultad	{nivel de dificultad}
Descripción	{explicación breve del programa}

Parámetros de las funciones o procedimientos
Además de lo anterior se documentará **todos** los parámetros de las funciones y procedimientos con la siguiente infoirmación.

Parametros	Nombre Corto	Descripción
{nombre del parámetro}	{identificador}	{descripción del parámetro}

Uso de tablas
Se documentará por cada tabla de la base de datos la siguiente información:

Nombre completo de la tabla en la base de datos

Nombre de Programa	SELECT	INSERT	UPDATE	DELETE

Por ejemplo:

Telecrédito.dbo.LogTelecredito				
Nombre de Programa	SELECT	INSERT	UPDATE	DELETE
ActualizaLogTelecredito		X		
LimpiaLogTelecredito				X
ConsultaLogTelecredito	X			

ANEXO 8: Encuestas a Jefes de Departameno y Análistas Técnicos

ADMINISTRACION DE SERVIDORES CENTRALIZADA
CONSOLIDACION DE SERVIDORES
ENCUESTA

I. GENERALES DE ADMINISTRACION

1. Por qué se esta optando por la Consolidación de Servidores? (seleccionar todas las que apliquen)

- ☐ Mejor Administración
- ☐ Menor Costo Total de Propiedad
- ☐ Mejor Disponibilidad
- ☐ Mejor Confiabilidad
- ☐ Otros
- ☐ Seguridad
- ☐ Mejor Performance
- ☐ Rápida Implementación de Cambios
- ☐ Mejor tiempo de respuesta al Cliente
- ☐ No conozco lo suficiente para irame undicar porque

2. Qué comentarios tiene sobre la Administración Centralizada de Servidores? (Consolidación de Servidores)

3. La documentación de sus Aplicativos con que cuenta es: (seleccionar todas las que apliquen)

- ☐ Modelo de Datos (MER) ( ) Aplicativos
- ☐ REF ( ) Aplicativos
- ☐ RET ( ) Aplicativos
- ☐ RMS ( ) Aplicativos
- ☐ Mapa de Aplicación ( ) Aplicativos
- ☐ Otros, Detalle: _____
- ☐ Presentación del Aplicativo ( ) Aplicativos
- ☐ Manual de Instalación ( ) Aplicativos
- ☐ Modelo de Procesos ( ) Aplicativos
- ☐ Modelo de Componentes ( ) Aplicativos
- ☐ Modelo de Objetos ( ) Aplicativos
- ☐ Diagrama de Casos y Usos ( ) Aplicativos

4. Cuanto tiempo le demandaría entregar una copia de esta información? (De la Pregunta 3)

- ☐ Menos de 2 días
- ☐ Mas de 5 días
- ☐ De 2 a 5 días
- ☐ Otro: Especifique_____

5. Cual es el Canal de Comunicación adecuado para este tipo de proyecto?

- ☐ Email
- ☐ Tickets
- ☐ Teléfono
- ☐ Otro: Especifique _____

6. Quienes desempeñan la labor de administración de servidores de desarrollo?

- ☐ Servicorp
- ☐ Soporte propio
- ☐ IDS
- ☐ Ninguna

Si es soporte propio, favor indicar quien los apoya y en que medida (tiempo):

PERSONA	% Tiempo Invertido	
1.		☐ BCP ☐ MP
2.		☐ BCP ☐ MP
3.		☐ BCP ☐ MP
4.		☐ BCP ☐ MP
5.		☐ BCP ☐ MP
6.		☐ BCP ☐ MP

7. Cómo le gustaría que se mejore la labor de administración de servidores de desarrollo?

☐ Migraciones Simples

☐ Optimizar Performance

☐ Aplicación de Parches en el momento adecuado

☐ Contar con base de datos similares a producción

☐ Mantener una política de Backups Confiable.

☐ Controlar el Manejo de Versiones

☐ Otros, Especifique_____

II. SEGURIDAD DE LA INFORMACION

8 Le puede ser de utilidad contar con información enmascarada y depurada de producción?

☐ SI ☐ NO

Detalle el porque SI,NO:

9 Maneja procesos para obtener copia de la información de producción?

☐ SI ☐ NO

Si la respuesta es SI, especifique:

☐ Uso Procesos de Enmascáramiento

☐ La Información es proporcionada por ( ) IDS y ( ) Seguridad de la Información

☐ Enmascarada ☐ Sin cambios

☐ Otros: Detallar_____

_____

10. Se ejecutan procesos de depuración de información ?

☐ SI ☐ NO

11. Si la respuesta es SI, especifique ?

☐ En Desarrollo

☐ En Certificación

☐ En Producción

Detalle, si es en Producción:

III. ESTANDARES

12. Que utiliza al momento de desarrollar o dar mantenimiento a un Aplicativo?

☐ UML ☐ MSF

☐ OOP ☐ Revisión de Pares ( ) REF ( ) RET ( ) CODIGO FTE.

Otros, Especifique: _____

_____

IV. AMBIENTES DE CERTIFICACION

13. Cuales son sus Ambientes de Certificación?

Servidores	Tipo Servidor	Aplicativos	Otros aplicativos en el servidor
1.			
2.			
3.			
4.			
5.			
6.			

14. En caso de una caída inesperada del servidor. Se cuenta con documentación necesaria para su pronta recuperación ?

☐ SI ☐ NO

Si la respuesta es SI, especifique:

☐ Build Document ☐ Diagrama de Arquitectura

☐ Procedimiento de Instalación ☐ Otro, Especifique_____

15. Cuanto tiempo le demandaría entregar una copia de esta información? (De la Pregunta 14)

☐ Menos de 2 días ☐ De 2 a 5 días

☐ Mas de 5 días ☐ Otro: Especifique_____

16. Dónde y cómo almacena la información diariamente (Código Fuente)?

☐ Computadora del Programador ☐ Servidor Centralizado

☐ PC Centralizada ☐ PC del AT Responsable

☐ Ninguna ☐ Otro, Especifique

17. Que herramienta utiliza para la administración de Código Fuente?

☐ VSS ☐ CCC/Harvest

☐ Ninguna ☐ Otro, Especifique

18. Maneja un control de las 3 últimas Versiones de su Código Fuente?

☐ SI ☐ NO

19. En caso use el CCC/Harvest. Qué comentarios le merece ?

V. ADMINISTRACION DE PROBLEMAS

20. Cómo administra los problemas que se presentan durante el Desarrollo?

- ☐ MSDN Internet / Tech Net
- ☐ Usamos repositorio de problemas y soluciones
- ☐ Consultoría a MCS
- ☐ Otro, Especifique
- ☐ Ninguno

21. Que tipos de problemas se presentan en producción?
Priorizar por ocurrencia. (1) Frecuentemente ... (5) Casi Nunca

- ☐ Componentes ( )
- ☐ Base de Datos (SQL Server) ( )
- ☐ SNA Server ( )
- ☐ Paginas WEB ( )
- ☐ Ninguno ( )
- ☐ Otro, Especifique ( )

VI. MIGRACION

22. Existen planes de Migración?

- ☐ Windows 2000 Server
- ☐ SQL Server 2000
- ☐ Host Integrator
- ☐ VB 4.0, 5.0 a VB 6.0
- ☐ Ninguna
- ☐ Otros, Especifique

APLICACIÓN	HERRAMIENTA A MIGRAR
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

23. Los planes de Migración son a?

- ☐ Mediano Plazo (De 2 a 6 meses)
- ☐ En Proceso
- ☐ Largo Plazo (De 6 meses a más)
- ☐ No se ha definido una fecha

VII. SEGURIDAD

24. Existe un Backup de la información? ☐ Si ☐ No
Realiza Backup de:

- ☐ Fuente
- ☐ BD (SQL)

- [ ] Documentación (Aplicativos, congelados)
- [ ] Temas de Interés
- [ ] Otros Especifique _____

25. Que medios de seguridad utiliza? (BACKUPS)

- [ ] Copia a Disco Duro
- [ ] Copia a Cinta
- [ ] ARCSERVE
- [ ] DUMP / LOAD (SQL Server)
- [ ] Otro
- [ ] Ninguno

26. Si usa un medio de seguridad, lo verifica ? (RESTORE de PRUEBAS)

- [ ] SI
- [ ] NO

Con qué frecuencia ? _____

27. Las copias de seguridad se efectúan:

- [ ] Automáticamente (Agenda)
- [ ] Manualmente (Por Demanda)

VIII. SOPORTE y OPTIMIZACION

28. Controla la Performance de los Servidores?

- [ ] SI
- [ ] NO

Si la respuesta es SI, Especifique con que producto:

- [ ] Performance Monitor
- [ ] System Monitor
- [ ] Event Viewer
- [ ] Trace SQL
- [ ] Task Manager
- [ ] Patrol
- [ ] Otro: _____
- [ ] SQL Probe

29. Con que Frecuencia controla la Performance de los Servidores ?

- [ ] Diariamente
- [ ] 3 Veces a la Semana
- [ ] Cada 15 Días
- [ ] Mensualmente
- [ ] Otro: Especifique_____

30. Controla el efecto de la Performance sobre los Aplicativos?

- [ ] SI
- [ ] NO

Detalle: Cómo y con que frecuencia ?_____

31. Con que frecuencia se tiene que instalar, reinstalar o formatear los Servidores ?

- [ ] Cada 3 días
- [ ] Cada 5 días
- [ ] Semanalmente
- [ ] Quicenalmente
- [ ] Mensualmente
- [ ] Desconozco
- [ ] Cada vez que tengo problemas? ( ) Veces al ( )Día / ( ) Semana / ( ) Mes / ( ) Año

32. La reinstalación de servidores, se debe principalmente a: ?

- ☐ Problemas con el S.O.
- ☐ Problemas con MTS, COM+
- ☐ Problemas con el Aplicativo
- ☐ Problemas de Hardware
- ☐ Problemas con Antivirus
- ☐ Problemas con SQL Server
- ☐ Otros:

33. Considera necesario usar alta disponibilidad en Servidores de Desarrollo?

☐ SI ☐ NO

32. Si la respuesta es SI, Especifique ?

- ☐ RAID 1
- ☐ RAID 10
- ☐ Clustering
- ☐ Otro
- ☐ RAID 5
- ☐ Otro
- ☐ NLB

Por qué ? _____

_____

33. Utiliza Servidores Virtuales?

☐ SI ☐ NO

Si la respuesta es SI, Especifique:

☐ VMWARE ☐ Otro

Comentarios de VMWARE:

34. Existen planes de crecimiento de sus servidores, componentes de los mismo?

☐ SI ☐ NO

Si la respuesta es SI, Especifique:

Servidor	Característica	Aplicativo - Proyecto
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		

35. Tiene planes para Adquirir más Servidores?

☐ SI ☐ NO

Si la respuesta es SI, Especifique:

Servidor	Característica	Aplicativo - Proyecto
1.		
2.		
3.		
4.		

5.		
6.		
7.		
8.		
9.		
10.		

36. Tiene presupuesto asignado para la adquisición de Servidores para desarrollo?

☐ SI ☐ NO

Si la respuesta es SI, transferiría su presupuesto:

☐ Si ☐ No

Cuál es monto aprobado que tiene asignado Porqué? _____ _____

US$ _____ (sin IGV) _____

_____

37. Cuantos proyectos tiene en cartera ?

☐ De 1 a 3 ☐ De 4 a 6

☐ Más de 6 ☐ Ninguno

OBSERVACIONES

RECOMENDACIONES:

GRACIAS POR SU TIEMPO!!

ANEXO 9 : RELACIÓN DE SERVIDORES

				CONFIGURACION ACTUAL				
	NOMBRE	MARCA	MODELO	PROCESADOR	MEMORIA	DISCO	OTRO	SW BASE
1	ADOSAPSRV1	IBM	IBM PC Server 330	Pentium Pro 200Mhz	256	4 DE 4 GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª
2	ADOSAPSRV2	Compaq	Compaq Proliant 5500	2 procesadores Pentium de 200Mhz	256	2 DE 9.1 GB 1 DE 4 GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª
3	ADOSAPSRV3	Compaq	Compaq Proliant 5500	2 procesadores Pentium de 200Mhz	256	1 DE 9.1 GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª, MTS 2.0 SP1 - Option Pack 4 de Windows NT
4	ADOSAPSRV4	IBM	IBM Netfinity 5000	Pentium III de 500Mhz	256	1 DE 4GB 1 DE 9.1 GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª, MTS 2.0 SP1 - Option Pack 4 de Windows NT, SNA Server 4.0 SP3
5	ADOSAPSRV5	IBM	IBM Netfinity 5000	2 Pentium III de 500Mhz	512	1 DE 4 GB 1 DE 25 GB	CD ROOM	Windows NT Server SP6, SQL Server 7.0 SP 3
6	ADOSAPSRV6	Compaq	Compaq Proliant 6000	2 Procesadores 400Mhz	512	1 DE 18 GB 1 DE 4 GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª, MTS 2.0 SP1 - Option Pack 4 de Windows NT, SNA Server 4.0 SP3
7	BCPRECOLAMO_BK	Compaq	Compaq Proliant 5500	2 procesadores Pentium de 550Mhz	512	4 DE 9.1GB	CD ROOM	Windows NT Server SP6, SQL Server 6.5 SP 5ª
8	BCPCDDSRV1	IBM	NetFinity 7100	2 Pentium III Xeon 750 Mhz	256Mb	25 GHz	Tape Backup, CD-ROM	WNT4.0 sp 6.0, SQLServer 7.0 sp 3, SNA 4.0 sp1, MTS2.0, IIS 4.0

9	CDD215	IBM	300 GL	Pentium MMX 200Mhz	32Mb	2.4Gb		WNT4.0 sp 6.0
10	BCP_SSDESA	IBM	NetFinity 5600	2 procesadores Pentium III 1Ghz	512MB	3 discos de 36.4 GB	Raid5	- NT4 SP4 - Exchange cliente - Sybase 12
11	BCPRENTA2	NCR		4 procesadores Pentium 166mhz	512MB	26GB (Distribuidos en 6 discos)		- NT4 SP4 - SQL 7.0 - Option pack 4.0 - Crystal reports 8.5
12	BCPDAD1	IBM	NetFinity 5600	1 Pentium III 667 Mhz	262 MB	25.3 GB	CD-ROM	SQL 7.0 SP 2 / NT Server 4.0 SP 6
13	HB2_D	Unisys	Aquanta DS/6	P/6 200 MHz	256	8.08	CD-ROM	NTServer SP6 IIS SQL Server 6.5
14	TRM_D	Compaq	Deskpro	Pentium II 333 MHz	64	4		NTServer SP3 Intersoft 8.31
15	BD1_D	Unisys	Aquanta DM/6	Pentium Pro 200 MHz	128	8.27	CD-ROM	NTServer SP6
16	Campaign_D2	Unisys	Aquanta DL	Pentium I 133 MHz	32	2.97		NT WS SP6
17	Sdialer_D2	Unisys	Aquanta DL	Pentium I 133 MHz	64	1.01		NT WS SP6
18	TSERVER_D	IBM	Netfinity 5100	Pentium III 863 MHz	128	16	CD-ROM	NTServer SP6 Genesys 5.16
19	BCPCYRSA_OLD	Compaq	Proliant 1600	2 Pentium II 300 MHz	256	30	CD-ROM TAPE BACKUP	NT Server 4 sp4 SQL Server 7
20	BCPTRANSACTOR	IBM	Netfinity 5100	P II 664Mhz (*)	256 MB	a) 8.5 Gb b) 8.5 Gb	CD ROM	NT 4.0 sp 6a op 4 / SQL 7.0 / SNA
21	SERVIMATSRV	Unisys	5907	P I 90 Mhz	64 MB	a) 1.0 Gb b) 2.0 Gb		NT 4.0 sp4 / SQL 6.5
22	CCTC (PC)	IBM	300GL	P III 547 Mhz	196 MB	10 Gb		NT 4.0 sp 6a op 4 / SQL 7.0 / SNA

23	S999100 (PC)	ATT	S10	P I 89 Mhz	64 MB	a) 2.0 Gb b) 2.0 Gb		NT 4.0 sp4 / SQL 6.5 / SNA 4.0
24	S999997 (PC)	IBM	300GL	P III 730 Mhz	128 MB	10 Gb		NT 4.0 sp4 / SQL 6.5
25	S999998 (PC)	IBM	300GL	P III 548 Mhz	128 MB	10 Gb		NT 4.0 sp4 / SQL 6.5
26	ADOSRV4	IBM	AT/AT Compatible	1 Pentium 167 MHZ	64 MB	2.0 GB		Nt Server 4.0 SQL Server 6.5 SP 3
27	ISUITE_D	COMPAQ	ML370G2	2 Pentium III	1 GB	63 GB	TAPE BACKUP	NT Server 4 sp6 SQL Server 7
28	S111141B	Unisys	S40	2 Pentium 166	128MB	8GB	Unidad de cinta 4mm.	
29	CRED_COMER	IBM	NetFinity 5000	2 PII - 400Mhz	1GB	30GB		
30	DESATLCAPP1	NCR	3426	Pentium Pro	260 Mb	8 Gb	CDROM, Unidad backup	NT 4 SP6a, Option Pack
31	DESATLCSQL	NCR	3426	Pentium Pro	260 Mb	8 Gb	CDROM, Unidad backup	NT 4 SP6a, SQL 7 SP2, SNA Server 4 SP4
32	TESTTLCSNA	NCR	3426	Pentium Pro	128 Mb	8 Gb	CDROM, Unidad backup	NT 4 SP6a, Option Pack, SNA Server 4 SP4, Connect 3.3
33	TESTTLCSQL	IBM	Netvista	Pentium III	512 Mb	20 Gb	CDROM	NT 4 SP6a, SQL 7 SP 2, Connect 3.3
34	TESTTLCAPP1	IBM	Netvista	Pentium III	512 Mb	20 Gb	CDROM	NT 4 SP6a, Option Pack
35	TESTTLCAPP2	IBM	Netvista	Pentium III	512 Mb	20 Gb	CDROM	NT 4 SP6a, Option Pack
36	TESTTLCAPP3	IBM	Netvista	Pentium III	512 Mb	20 Gb	CDROM	NT 4 SP6a, Option Pack
37	TESTFILESVR	IBM	Netvista	Pentium III	512 Mb	20 Gb	CDROM	Win 2k Advanced Server SP 2, Option Pack, Visual Studio (VSS)

38	TLCINT01	IBM	PC 300GL	Pentium III	128 Mb	10 Gb		NT 4 SP6a, SQL 7 SP2, SNA Server 4 SP4
39	TLCMN02	IBM	PC 300GL	Pentium III	128 Mb	10 Gb		NT 4 SP6a, SQL 7 SP2, SNA Server 4 SP4
40	SQLSERVER	NCR	3426	Pentium Pro	128 Mb	4 Gb	CDROM, Unidad backup	NT 4 SP6a, SQL 7 SP2
41	TLCPRE01	NCR	3426	Pentium Pro	200 Mb	8 Gb	CDROM, Unidad backup	NT 4 SP6a, Option Pack, SNA Server 4 SP4
42	TLCPRE02	NCR	3426	Pentium Pro	260 Mb	8 Gb	CDROM, Unidad backup	NT 4 SP6a, Option Pack, SNA Server 4 SP4, Connect 3.3
43	DESATLCDEBUG	IBM	Netvista	Pentium III	512 Mb	20 Gb	CD ROM	NT 4 SP6a, Option Pack, SQL 6.5 SP 5
44	TTFS01	COMPAQ	Deskpro	Pentium	112 Mb	4 Gb		NT 4 SP6a, SQL 7 SP2, Option Pack
45	PEDESA	COMPAQ	Proliant	Pentium II	128 Mb	3Gb	CD ROM	NT 4 SP6, Option Pack, SQL 7.0 Sp2, SNA
46	BCPPAGONET	IBM	Netfinity 5000	2 Proc. Pentium III 648 MHZ	512 MB	20 GB	CD ROM	Windows NT 4.0 OP 4 IIS 4.0 WAS 3.0 FP 3 (Redirector)
47	B2BDESA02	IBM	Netfinity 5100	1 Proc. Pentium III 730Mhz	512 MB	18 GB	CD ROM	Windows NT 4.0 OP 4 IIS 4.0 WAS 3.0 FP 3 DB2 6.1 FP 4 SNA Server 3 SP 3 Cics Transaction Gateway 3

48	B2BDESA03	IBM	Netfinity 5100	2 proc. Pentium III 730 MHZ	512 MB	20 GB	CD ROM	Windows NT 4.0 OP 4 Cliente DB2 6.1 SNA Server 3 SP 3 + COMTI ADO 2.5 MTS 4.0
49	BCPB2B	COMPAQ	Proliant 1600	2 proc Pentium II 300 MHZ	512 MB	4 GB	CD ROM	
50	BCP2000A	COMPAQ	PROLAINT 3000	PENTIUM II 300 MHZ	256 MB	30 GB		MSSQL SERVER 6.5
51	BCP_SSED	COMPAQ	DESKPRO	PENTIUM 166 MHZ	64 MB	3 GB		MSSQL SERVER 6.5
52	BCPFINANDESA	IBM	Netfinity 5100	Pentium III 800Mhz	256	28 GB		NT 4.0 / SQL 7.0
53	BCPMCDESA	IBM	PC Server 325	Pentium Pro 200 Mhz	128	14 GB		NT 4.0 / SQL 6.5
54	ARSERVER	IBM	PC 300 GL	Pentium II 200 Mhz	132	10 GB		NT 4.0 / SQL 7.0
55	CREDDESA	AT&T	Globalyst S10	Pentium 166 MHz	128	4 GB		NT 4.0 / SQL 6.5
56	BCPMCDESA2	AT&T	Mod. 3430	Pentium	128	5 GB		NT 4.0 / SQL 6.5
57	BCPSEG4	IBM	Netfinity 5100	Pentium 550 Mhz	750	20 GB		
58	**HB1_D**	IBM	NF5100	2 Pentium III 733 Mhz,	256 MB	4 discos de 8.5 GB en raid 5	CDROM	NT 4.0 Enterprise SP6a, Option Pack 4.0 (Standard),SNA Client 4.0
59	**HB4_D**	IBM	NF5100	2 Pentium III 733 Mhz,	512 MB	4 discos de 8.5 GB en raid 5	CDROM	NT 4.0 SP6a, Option Pack 4.0 (Standard), SQL 6.5 SP5 SNA Server 4.0 SP3 + COMTI

60	**VIATIENDAS002**	IBM	NF5100	1 Pentium III 866 Mhz,	512 MB	4 discos de 9 GB	CDROM	NT 4.0 , DB2 , WebSphere
61	**WEB_D** (Reemplazaría a CDE1)	Compaq	DeskPro	Pentium	64 MB	3 GB		NT 4.0 Enterprise SP6a, Option Pack 4.0 (Standard), SNA Client 4.0, SQL 6.5 SP5
62	**CENTER47**	IBM	Personal Computer 300GL	Pentium 348 Mhz	64 MB	4 GB		WorkStation, option pack 4.0, Office 97
63	**HB0_D (Prestado)**	Compaq	Proliant 5000	2 Pentium 333 Mhz	256 MB	1 disco 4GB 2 disco 8.46 GB	CDROM	Windows 2000 sp2