40

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS
PROFESIONALES "ACATLÁN"

TUXEDO COMO MONITOR DE TRANSACCIONES (MIDDLEWARE) EN ARQUITECTURA 3 CAPAS

MEMORIAS DE DESEMPEÑO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A

SANDRA SÁNCHEZ RANGEL

ASESOR:
LIC. GUADALUPE DEL CARMEN RODRÍGUEZ MORENO

UNAM
CAMPUS ACATLÁN

ACATLÁN, MEXICO

OCTUBRE DEL 2001.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIAD NACIONAL AUTÓNOMA
DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ACATLÁN

TUXEDO COMO MONITOR DE TRANSACCIONES (MIDDLEWARE) EN ARQUITECTURA 3 CAPAS

MEMORIAS DE DESEMPEÑO PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN MATEMÁTICAS APLICADAS
Y COMPUTACIÓN
P R E S E N T A

SANDRA SÁNCHEZ RANGEL

ASESOR: LIC GUADALUPE DEL CARMEN RODRÍGUEZ MORENO

ACATLÁN, MÉXICO

OCTUBRE DEL 2001

Al matriarcado al cual pertenecí, las personas más fuertes, alegres e histéricas que he conocido, características que resaltan en un hogar de mujeres: mi madre que no tengo como agradecerle su fuerza, valentía y constancia para educar a cuatro hijas sola; la finalización de mi carrera académica es el resultado también de tu esfuerzo. Gracias por habernos querido tanto. Mis hermanas, que son lo que más quiero, a todas mi admiración y gratitud por su apoyo y compañía.

A Gaby, Gisela, Mónica y Maritrini.

Y aunque no pudo estar con nosotras, gracias

A mi padre.

En este momento quiero mencionar a los maestros que marcaron mi vida e hicieron posible mi desarrollo académico y aunque no recuerdo el nombre completo de todos ni el nombre de otros, mencionaré a los que recuerdo con más cariño: profesor Bilaboa. sec 66 Matemáticas; profesora Gloria Torices, sec. 66 Historia y Directora; Nachito, sec. 66 Español; profesor Pérez y Juárez, sec. 66 Física y Química; profesora Edith Zepeda, Preparatoria 4 Temas Selectos de Matemáticas; Ing. Jorge Zamudio, Cálculo ENEPAcatlán; Ing. Víctor Palencia, Ecuaciones Diferenciales ENEP Acatlán; Lic. Guadalupe Rodríguez, mi asesora y amiga ENEP Acatlán; finalmente a los profesores de DGSCA. A todos ustedes y a quienes desgraciadamente no puede recordar les agradezco su tiempo.

A mis queridos amigos que siempre le dieron alegría a mi vida por sus ocurrencias tan gratas: Maribel Dávila, Lupita Rodríguez, Alejandro Luna, Rosendo Villaseñor, Pilar Meléndez, Olivia Sánchez, Blanca Martínez, Mariana Pineda, Elio Vega, Soraya Jiménez y a los que ya no veo más. Gracias por su amistad y por compartir conmigo un poco de su vida y alegría.

Esta vida puede ser tan placentera como nos esforcemos por ello, siendo comprensivos cariños, pacientes y sinceros. Porque me das todo eso, que podría llamarse amor, a mi compañero de hace ya mucho tiempo, te doy las gracias por tu esfuerzo diario y por estar conmigo. Te Amo.

A mi esposo,

Arturo García Aidana

INDICE

1. ANTECEDENTES

1.1 Generalidades

Desde 1988 la recaudación de impuestos en nuestro país ha sido manejada a través del Sistema Integral de Recaudación (SIR) que maneja la Administracion General de Recaudación (AGR) de la Subsecretaría de Hacienda y Crédito Público (SSHCP).

El SIR es el encargado de recibir, controlar y administrar toda la información referente al pago de algunos de los impuestos más importantes en nuestro país

Para poder cumplir con este objetivo la SSHCP cuenta con las siguientes entidades.

1.1 Subsecretaría de Hacienda y Crédito Público (SSHCP). Dicta la normatividad y política económica del país.

1.2 Subsecretaría de Ingresos (SSI). Está encargada de diseñar e implantar las políticas de ingresos que permiten establecer instrumentos y medidas que facilitan la recaudación

1.3 Subsecretaría de Egresos (SEE) Prepara y controla el presupuesto de egresos de la Federación y elabora la ley de egresos

1.4 Procuraduría Fiscal de la Federación Proporciona asesoría fiscal tanto para la SHCP como para el Poder Ejecutivo, y resguarda los acuerdos y tratados internacionales firmados por México en materia fiscal.

1.5 Tesorería de la Federación (TESOFE) Se encarga de recaudar y concentrar los importes provenientes del pago de impuestos, derechos y demás ingresos federales, así como controlar el ejercicio del presupuesto de egresos de la Federacion, incluyendo servicios de deuda pública, tanto interna como externa

1.6 Unidad de Comunicación Social. Maneja las relaciones públicas de la Secretaría y difunde información proveniente de discursos, comunicados, conferencias, entrevistas a funcionarios, etc

El objetivo de la AGR es establecer las políticas y los programas para recaudar contribuciones, aprovechamientos, accesorios, productos e imposiciones de multas

Sus atribuciones principales son:

- Establecer la política y los programas para recaudar.
- Autorizar el cobro coactivo y garantías de los créditos fiscales
- Autorizar las devoluciones o compensaciones de contribuciones, así como otorgar estímulos fiscales y disminución de pagos provisionales
- Proporcionar información y orientación al contribuyente.
- Mantener actualizado el Registro Federal de contribuyentes
- Diseñar y probar formularios fiscales

1.2 Estructura Organizacional

La AGR divide sus actividades en 3 niveles para el logro de sus objetivos y el ejercicio de sus funciones

- El Nivel Central.
- El Nivel Regional
- El Nivel Local

1.2.1 Nivel Central

Constituido por 5 Administraciones Centrales, se encarga de emitir la normatividad, dictaminar los procedimientos con los cuales deben operar las demas áreas, evaluar la operación de los niveles regionales y locales, planear e implantar nuevos proyectos y atender a los grandes contribuyentes

Entre las numerosas funciones que tiene este nivel, destacan las siguientes

- Regula los procedimientos de operación para la recaudación, sustentarlos de manera legal y diseñar los avisos, declaraciones y demás documentos que influyen en el proceso

- Vigila y evalúa la operación de cobro de servicios que se realiza a través de las Oficinas Regionales y Oficinas Locales

- Concentrar la informacion contable de los niveles regionales e integrar la información contable de todas las Oficinas en el país

- Actualización del Módulo de Registro Federal de Contribuyentes - Se encarga de dar seguimiento a la operatividad de las Oficinas Locales Vigilar que las bases de datos de Módulo de RFC esten actualizadas, vigilar que se realice el proceso de depuracion de RFCs de manera permanentemente, se almacena toda la informacion de los contribuyentes a nivel Nacional

- Diseñar , coordinar e integrar los planes institucionales de la Empresa, así como buscar la optimización y sistematización de los procesos operativos

- En cuanto al área de Sistemas, ésta tiene como funciones principales coordinar los cambios y adecuaciones que el área de normatividad solicita a las diversas aplicaciones que tiene a su cargo.

- Atender y vigilar el cumplimiento del pago de servicios de los contribuyentes más importantes del país

1.2.2 Nivel Regional

Compuesto por varias Administraciones Regionales u Oficinas Regionales (ORs), coordina las actividades del nivel local y concentra la información de las declaraciones que presentan los contribuyentes

Tiene como funciones principales las siguientes

- Coordinar y concentrar la información de las Oficinas Regionales
- Operar y administrar el Módulo de Declaraciones y Pagos del sistema actual, a través del cual se introduce al sistema de forma manual o por medio de un reconocimiento de imagenes, la información relativa a los pagos que los contribuyentes realizan en los bancos y los que presentan en las Oficinas Locales
- Transmitir la información de los pagos a las Oficinas Locales
- Efectuar cierres contables diarios y procesos de validación que integran todos los movimientos que se han efectuado en la Oficina Regional en cada período

1.2.3 Nivel Local

Las principales funciones de las Oficinas Locales son las siguientes

- Administrar y mantener el concentrado de información local
- Atender y orientar a los contribuyentes en trámites relacionados con .
    - Movimientos a RFC, los cuales son Inscripciones cambios , suspensión, emision de códigos de barras y cédulas de identificación fiscal.
    - Atender solicitudes de devoluciones y compensaciones
    - Solicitud de pago en parcialidades (interaccion con el módulo de cobro)

- Vigilar el cumplimiento de las obligaciones de los contribuyentes con la información que las Oficinas Locales envían respecto a las declaraciones presentadas.
- Emitir avisos para los contribuyentes retrasados en pagos de impuestos

Actualmente existen varias Oficinas Locales en nuestro país que cuentan con alrededor de 16 a 64 terminales cada una

Las Oficinas Locales organizan sus actividades a través de las siguientes áreas.

- RFC y obligaciones fiscales en esta área se realizan los trámites relacionados con RFC y se da seguimiento al cumplimiento de los pagos de los contribuyentes
- Orientación y Servicios a Contribuyentes.
- Control de Créditos y Cobro Coactivo En esta área se controla el inventario de crédito de los contribuyentes y se realizan las actividades necesarias para hacerlos efectivos
- Muchos de los procesos que se realizan en las Oficinas Locales son manejados a través del sistema actual, aunque existen muchas otras actividades que continúan siendo manuales

1.2.4 Sistema Actual (SIR) y el Proceso General

El proceso de cobro de Servicios involucra la participación de alrededor de 12,500 empleados, y es alimentado por la información que proporcionan todos los contribuyentes del país

Las operaciones diarias se realizan a través de diversas entidades pertenecientes a la AGR tales como Oficinas Regionales, Oficinas Locales, Módulos de Atencion. Centro de Procesamiento Nacional (Oficina Nacional), y otras entidades como Bancos

La figura 1.1 esquematiza de manera general la organizacion de la AGR para el cobro de impuestos.

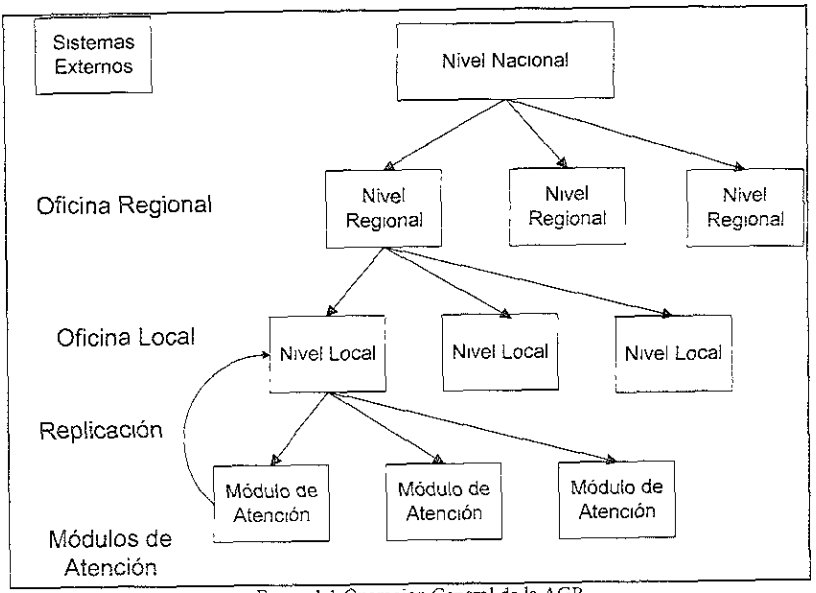


Figura 1.1 Operacion General de la AGR

En donde:

- La información a nivel nacional es integrada por la Oficina Central que dirige el departamento de Sistemas.
- Tanto la información regional general como el concentrado de los pagos de servicios es controlado por las Oficinas Regionales
- La atención al contribuyente en trámites no relacionados con pagos es responsabilidad de las Oficinas Locales
- Los avisos y/o cobros de servicios a los contribuyentes se efectuan a través de los Módulos de Atencion.
- Los Bancos también reciben pagos de impuestos

Por otro lado, durante el cobro de servicios se efectúa un proceso de actualizacion a 3 niveles

1. Los contribuyentes realizan movimientos a nivel local, a través de las Oficinas Locales, que afectan RFCs de la base de datos local
2. Al efectuarse un movimiento en el nivel local, se realiza una consulta a la base de datos nacional para verificar que el contribuyente no se encuentre dado de alta en otro lugar
3. A su vez la base de datos nacional actualiza a la base de datos regional que concentra los RFCs de los contribuyentes de todas las locales que estan en su jurisdiccion

1.2.5 Operación General a Nivel Regional y Local

De manera general un contribuyente puede realizar sus trámites a través del Banco o las Oficinas Locales de la siguiente manera

- Los trámites que se efectúan en el banco son aquellos relacionados con el pago de impuestos
- Después de recibir los pagos, el Banco envía los documentos a las Oficinas Locales más cercanas a su domicilio. Las Oficinas Locales las agrupan en lotes y las envían a la Oficina Regional, en donde la Oficina de Sistemas Regional las captura de forma manual o por medio de un OCR (Optical Character Recognition)
- Los datos capturados son procesados por contabilidad, y se transforman en asientos contables que viajan a la Oficina Local y en cintas mensuales que actualizan la base de datos nacional.
- Los trámites que se efectúan en la Oficina Local son aquellos relacionados con altas, cambios y servicios de RFC, solicitud de devoluciones, etc
- La documentación que acompaña a los trámites de RFC y Pagos de servicios en parcialidades, se pone en sobres e internamente se entrega al área de control de documentos, quien la distribuirá entre las áreas de RFC, Sistemas y Control de Créditos respectivamente.
- Al procesar la información, estas áreas generan archivos de impresión que se envían al departamento de Control de Módulos de Atención.
- Después de realizar las impresiones (de requerimientos, multas, avisos de devoluciones, cobros, etc) Control de Módulos de Atención las envía al área de Control de Documentos, quien las distribuirá a los contribuyentes.

Estos dos flujos, pago en el banco y servicios en la Oficina Local se ilustran en la figura 1 2

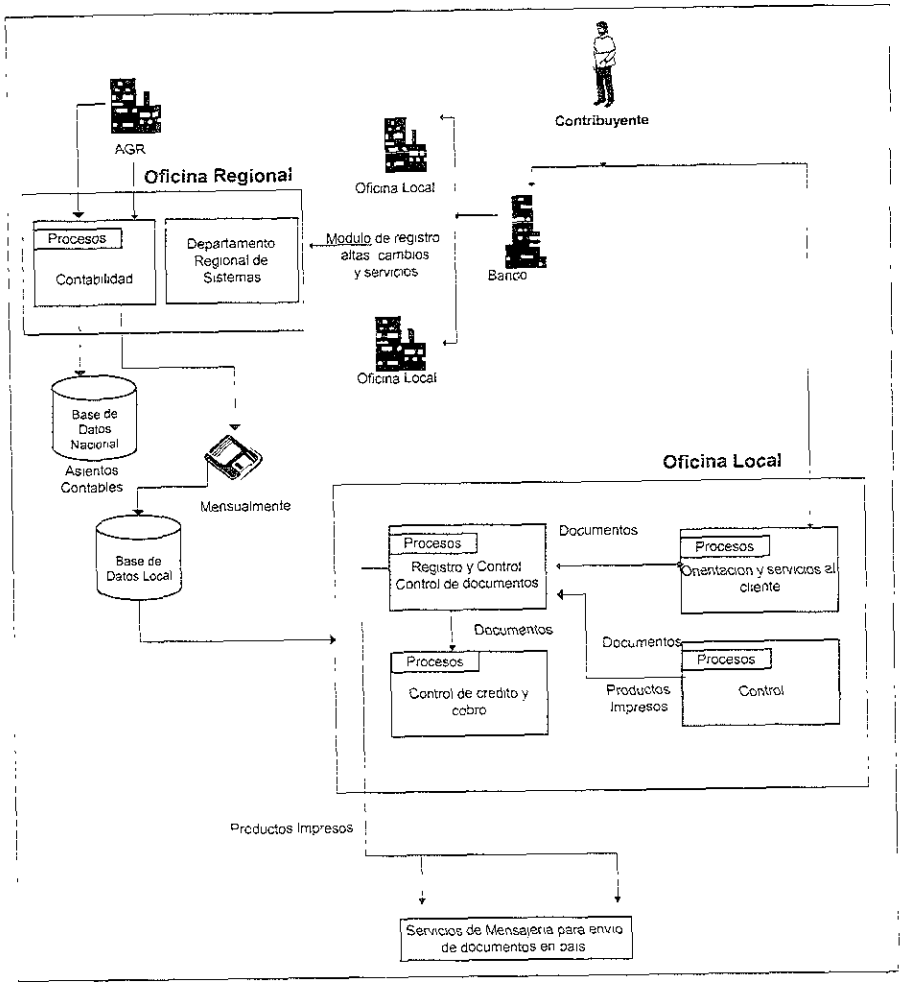


Figura 1 2 Operación General a Nivel Regional y Local

1.3 Módulos del SIR

Los módulos que forman al sistema actual, SIR, son

1 RFC - En este módulo se lleva a cabo la inscripción de los Contribuyentes (altas), y se prestan múltiples servicios como· cambio de domicilio, suspensión, cambio de actividades, emisión de cédulas de identificación fiscal, entre las principales. El número de movimientos que tiene este módulo diariamente es al rededor de 30.000

2 COBRO - En este módulo se manejan los créditos de los contribuyentes con la Secretaría de Hacienda

3. CONTROL DE OBLIGACIONES - En este módulo se vigilan las obligaciones de los contribuyentes y se emiten archivos de requerimientos.

4. CONTABILIDAD - En este módulo se lleva a cabo el Control de Ingresos por recaudación

5. NOTIFICACIÓN Y VERIFICACION - Esta parte del sistema es conocida también como línea de producción, y en ella se efectúan las impresiones de requerimientos, multas, notificaciones de la Secretaría.

6. DEVOLUCIONES y COMPENSACIONES - En este modulo se controla la devolución de dinero, por declaraciones con saldos a favor, esta parte del sistema está íntimamente ligada a la TESOFE. Actualmente este es uno de los módulos con mayores rezagos

7 DECLARACIONES y PAGOS - En este módulo se controlan todos los movimientos que hacen los contribuyentes en relacion al cumplimiento de sus obligaciones fiscales Este módulo es apoyado a través de los módulos de atención y captura manual El número de documentos que se maneja diariamente puede oscilar entre 25,000 y 110 000

8. ESTADÍSTICAS Este modulo maneja la información estadística solicitada por los usuarios.

1.3.1 Módulo Principal del Sistema Actual

Con la finalidad de dar un marco de referencia al proyecto "Sistema Nuevo", que automatizará las operaciones cobro de impuestos de la AGR a nivel nacional, así como al control de contribuyentes. se proporcionarán datos que faciliten su ubicación y que por tanto ayudan a comprender su importancia Estos datos son delimitados al módulo RFC. que se desarrolló por parte de SOFTTEK

1.3.2 Objetivo

El objetivo principal de este módulo es mantener actualizada la información de los contribuyentes (datos personales, obligaciones, servicios solicitados, bajas) en las bases de datos Nacional, Regional y Local

El submódulo principal de RFC es el de Control de documentos que tiene por objeto el recibir todos los trámites del contribuyente, clasificarlos y distribuirlos entre las áreas respectivas, y una vez realizados los procedimientos administrativos internos de cada área entregar órdenes de trabajo a los medios de mensajería para su envío

Los trámites de RFC se efectúan a través de los Módulos de Atención de las Oficinas Locales

1.3.3 Narrativa Operativa

El proceso de RFC inicia cuando el contribuyente solicita una inscripción, un cambio o un servicio (trámites) y termina con la actualización de la información en las bases de datos y con la solicitud para generar la Cédula Fiscal, las constancias de inscripción y los códigos de barras

Para iniciar cualquier trámite de RFC, el contribuyente llena los documentos de solicitud, después el contribuyente debe ir a cualquiera de Módulos de Atención, aquí sus documentos se meten en un sobre y se les pone un folio.

Control de documentos recibe los tramites enviados por los Módulos de Atención y los da de alta a través del subsistema de control de folios A partir de este ingreso son generadas las relaciones de sobres para entrega, a las cuales se anexan los documentos correspondientes, para cada area.

En el caso de RFC, el área de diagnóstico recibe los paquetes y separa los documentos que pertenecen a la Oficina Local Aquellos que no le pertenecen son regresados a Control de Documentos

Los documentos que pertenecen a la administración son revisados para comprobar que estén llenados correctamente y tengan la documentación requerida Al terminar esta revisión se anota en la relacion de sobres

para entrega el número de documentos factibles (serán aceptados posiblemente) y el de no factibles (rechazados). Por último, son enviados los documentos y su relación al área de Procesos en donde se captura el trámite en el subsistema de RFC

A partir de la captura se generan movimientos que son guardados en la base de datos local y transferidos al final del día a la Oficina Central vía satélite Esta hace una validación a nivel nacional para definir si el trámite se efectúa o no (reingresos). En caso de proceder se actualiza la base de datos nacional en caso contrario se marcan los RFCs para generar los rechazos. Cada mes la base de datos regional es igualada con la copia de la nacional vía cartucho

Cuando se termina de capturar el paquete se envía al área de Diagnostico los documentos aceptados y al área de Análisis de Resultados los rechazados La primera se encarga de lotificar los documentos para enviarlos a Resguardo Documental, y la segunda se encarga de hacer el seguimiento a los rechazos

A partir del procesamiento de la información son generados diversos productos Cédula Fiscal, constancias y códigos de barras, los cuales son impresos por el subsistema de Avisos Una vez que se tienen los productos se envían al área de Control de documentos, la cual los hace llegar al contribuyente a través de los medios de mensajería

El proceso descrito en esta narrativa se ilustra en las figuras 1.3 que se muestran a continuación .

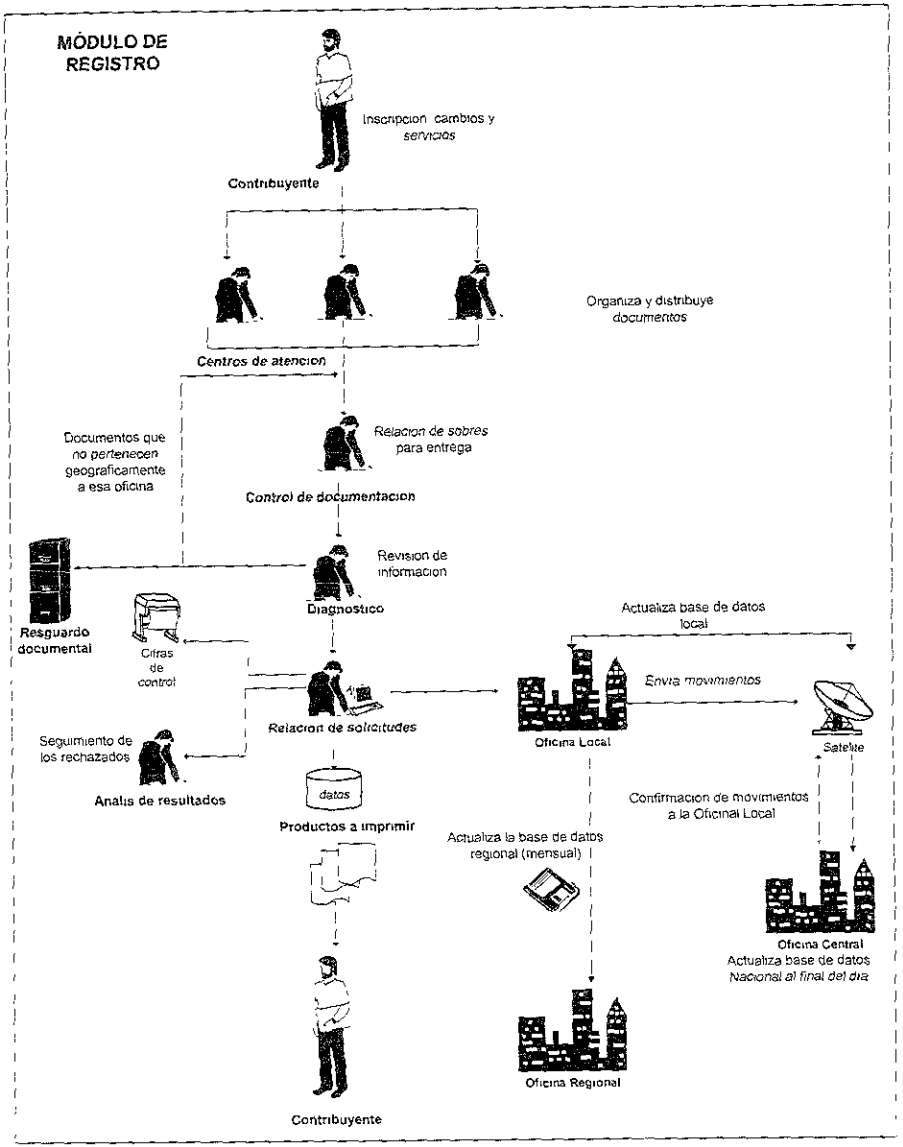


Figura 1 3 Proceso del Registro Federal de Contribuyentes

1.3.4 Narrativa Técnica

La funcionalidad del subsistema de RFC cubre tres principales tareas. Inscripciones, Cambios y Servicios

Las inscripciones registran al contribuyente en la base de datos local almacenando sus datos principales (nombre, domicilio, características, representante) en las diversas tablas de base de datos. Para autorizar el RFC se revisa que el contribuyente no esté dado de alta en la base de datos. de ser así el trámite es rechazado y se genera una multa al contribuyente. Si es aceptada la inscripción se registra el trámite en las tablas que correspondan Posteriormente existe otro sistema que se encarga de emitir la cédula

Los cambios, por ejemplo cambio de domicilio, son capturados y se valida que el contribuyente existe en la base de datos. de no estar, se rechaza el movimiento Los trámites que no hayan presentado irregularidades se registran en las tablas que correspondan

Los servicios, como son la reexpedición de documentos. son capturados y registrados en las tablas correspondientes Esta operación genera productos que son impresos por otro sistema para entregarlos al contribuyente.

Las tareas mencionadas registran movimientos que son transmitidos diariamente al Oficina Central vía satélite Ahí se realiza una validacion. si el movimiento es autorizado se actualiza la base de datos nacional, en caso contrario se marca el movimiento como rechazado

Posteriormente, la Oficina Central trasmite los reingresos a cada Oficina Local para que ésta a su vez actualice su base de datos A los movimientos rechazados se les generan avisos dependiendo del caso

El proceso descrito en esta narrativa técnica se ilustra en las figuras 1 4 que se muestran a continuación

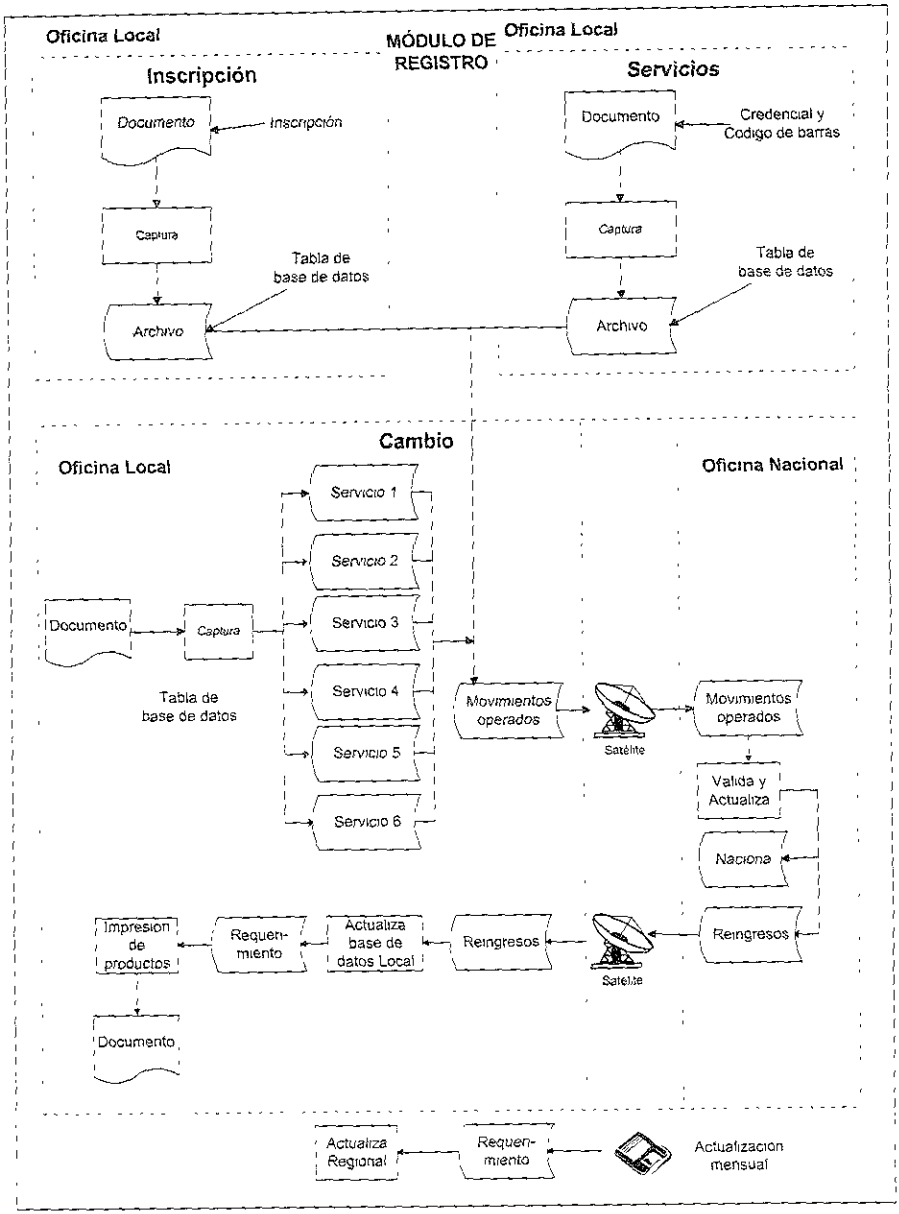


Figura 1 4 Proceso Tecnico del Registro Federal de Contribuyentes

1.4 Plataforma Tecnológica

A continuación se describe, de manera general, la infraestructura tecnológica con la que cuenta la AGR cuenta para llevar a cabo el cobro de Servicios (Nos limitaremos al enfoque de Sistemas).

1.4.1 Sistemas

En el Módulo de RFC, actualmente se utilizan principalmente equipos HP-9000 y UNISYS 6000 e Informix como ambiente de base de datos, distribuidos de la siguiente manera:

- 54 Oficinas Locales con equipo HP 9000
- 11 Oficinas Locales con equipo UNISYS 6000
- 9 Oficinas Regionales con equipo HP 9000

El total de aplicaciones, excepto las de OCR, se encuentran desarrolladas en un 80% en Informix-4GL versión 4.1X, el 15% en lenguaje "C" y el resto en scripts de shell

- Dentro del módulo de OCR (Optical Character Recognition), se cuenta con workstation HP modelos 735, 715/33 y 715/50, y como servidor para reconocimiento óptico de caracteres se utiliza un SUN SPARC station 20, el almacenamiento de las imágenes se realiza en un librería óptica modelo 100C.

La siguiente tabla, tabla 1.1 contiene una relación de los subsistemas que actualmente comprenden al sistema actual, incluyendo el alcance (Local, Regional o Central), así como el tipo de procesamiento y el volumen de operación de cada subsistema

RFC					
RFC de contribuyentes	X	X	X	On Line	Alto
Control de cédulas de identificación fiscal	X			On Line	Bajo
Código de barras	X			On Line	Alto/Medio
Estadísticas de RFC			X	Batch/Line	Alto/Medio
Subsistema de información de RFC	X			Batch	Alto
Consulta de la base de datos	X	X	X	On Line	Alto
Depuración de la base de datos	X		X	Batch	Bajo
Verificación de RFC	X		X	Batch	Bajo
Intercambio de información			X	Batch	Alto
Catálogos especiales		X		Batch	Medio

Tabla 1 1 Relación de Sistemas y Subsistemas

De la tabla anterior se observa que la mayor parte de los subsistemas se encuentran operando en el proceso por lotes o "batch"

Aunque el sistema fue migrado en dos ocasiones la concepción original se conservó. no se aprovecharon muchas de las bondades de la tecnologia de bases de datos Los Procesos son Batch

Los cambios formales que sufre el sistema al año son más de 700, mientras que los informales pueden ir de 700 a 1400 Cabe mencionar además que la documentación técnica actual de sistema es bastante deficiente

Actualmente el SIR opera a través de varios módulos, abarca más de 1000 programas y accede a más de 400 tablas de bases de datos El sistema esta hecho en un 90% con Informix 4GL, lo demás se encuentra en lenguaje C

Los cambios formales, que son los derivados por cambios en las disposiciones de ley, que sufre el sistema actual al año son más de 700. mientras que los informales pueden ir de 700 a 1.400 Cabe mencionar además que la documentación tecnica sistema actual es bastante deficiente

Este sistema ha soportado la operación recaudatoria de administraciones locales y regionales por varios años. Sin embargo, hoy en dia el sistema no responde adecuadamente a las necesidades de información y operación de usuarios y funcionarios, por lo que debe ser rediseñado

falta de integración entre los módulos y la información, la dificultad que representa la extracción de datos ra la toma de decisiones, el inadecuado manejo de créditos y el atraso en las devoluciones a los ntribuyentes, son algunos otros de los problemas que generan la necesidad de rediseñar al sistema actual esde sus bases, tomando las ventajas que ya presenta, pero subsanando las fallas que tiene y sobre todo gregándole nuevas funcionalidades que ahora se vuelven prioritarias.

Con el objeto de responder a las necesidades de cobro de impuestos, el sistema ha sufrido numerosos cambios, entre los que se encuentran dos migraciones y muchos programas "parche", que han tenido que ser desarrollados ante los constantes y repentinos cambios de nuestra legislación fiscal.

Esto, aunado al volumen de información, la falta de integridad entre los módulos que componen al sistema actual, la casi ausencia de esquemas de seguridad, la inflexibilidad ante el cambio la dificultad en la extracción de información para la toma de decisiones, dieron origen al proyecto "Sistema Nuevo"

Este proyecto de rediseño busca la conceptualización y construcción de un nuevo sistema, que aprovechará las técnicas de reingeniería y los avances tecnológicos para generar una aplicación más eficiente

roblemática y Oportunidades

eso recaudatorio involucra la participación de alrededor de 12.500 empleados, y es alimentado por la ación que proporcionan todos los contribuyentes de este país, sin duda una aplicación de estas ciones representa numerosas ventajas y desventajas.

ués del análisis realizado a los módulos que comprende el sistema actual y a los procesos asociados a él, tectaron múltiples ineficiencias las cuales serán detalladas a continuación

1 Orientación inadecuada al contribuyente para el llenado correcto de las diferentes solicitudes de trámites, lo que ocasiona
   1.1 Una gran cantidad de inconsistencias, debido a que desde un principio se introduce la información errónea en incluso en formatos no correspondientes al trámite. Los contribuyentes pueden darse de alta con características que no les corresponden
2. La información con la que opera el sistema actual no es confiable al igual que la transferencia de la misma. Por lo tanto la base de datos tiene mucha información desactualizada, incompleta e inconsistente
3 La base de datos no está normalizada dado que existen redundancias de tablas y campos.
4 No están establecidos estándares de programación y no se sigue la metodología existente.
5 Demasiada burocracia en el flujo de información en varios procesos, lo cual genera procesos manuales, repetitivos y gastos en papelería
6 El sistema no es flexible y por lo tanto no puede responder rápidamente a los cambios legales.
7 La funcionalidad en cuanto a la búsqueda de contribuyentes no es la adecuada ya que para localizarlos hay que buscarlos primero en la base de datos local y si no están ahí, es preciso salirse de la pantalla para buscarlo en la base de datos nacional, ya que son diferentes base de datos. Detallando este punto la base de datos local depende de qué Oficina esté operando, por ejemplo, en Tijuana se buscará al contribuyente en un servidor que se encuentra en Tijuana, si no está ahí , tendrán que buscarse al contribuyente en otra pantalla que trae información de la base de datos nacional que se encuentra en la Ciudad de México Existen 65 Oficinas Locales en el país
8 Actualmente los modulos no funcionan totalmente integrados, pues entre otras no cuentan con una comunicación trasparente, por lo que los descargos automáticos de información entre la mayoría de los módulos no se hace adecuadamente
9 La información a explotar no está actualizada para la mayoría de los módulos
10 No hay un banco de información para la toma de decisiones
11 La seguridad y controles con los que el sistema cuenta actualmente son débiles, ya que muchas tareas no dejan las debidas pistas de auditoría

En cuanto a la solicitud de cambios al sistema existen dos variantes que complican las adecuaciones.

12.1 Aquellas que se refieren a cambios de la legislación en las que los sistemas son modificados sin realizar un análisis para determinar el impacto de los cambios solicitados, y no son documentados

12.2 Las relacionadas con problemas de interpretación entre el área de normatividad y el área de sistemas.

13. Existe complejidad de administración del sistema

resumen, en la tabla 1.2 se tiene la relación de problemas y oportunidades a desarrollar. Se señala también s puntos donde el *middleware*[1] interviene como solución y posteriormente, en el capítulo 3, se expondrán .s características del middleware implementado, detallándose las mostradas en la tercera columna de la iguiente tabla

Falta de integridad entre los módulos	Desarrollo de un sistema integral.	Tuxedo brinda el *API* para la construcción de programas
Informacion desactualizada, inconsistente e incompleta	Limpieza de datos/Captura completa de datos	
Inflexibilidad del sistema al cambio.	Separación de Reglas y Objetos de Negocios.	Tuxedo se implementa en componentes.
Excesivo manejo de papel	Automatizacion de actividades y almacenamiento de imágenes	
Seguridad y controles débiles	Esquema de alta seguridad y pistas de auditoría	
Inadecuada orientación y servicio al contribuyente.	Esquemas de atención al contribuyente	
Falta de información y congruencia entre los reportes y las necesidades para la toma de decisiones	Manejo de información analítica y ejecutiva directa del Sistema	

[1] Middleware "crea, integra y administra aplicaciones distribuidas de gran escala en ambientes heterogéneos" Es la segunda capa en la arquitectura de 3 capas
httphttp://www aurorainfo com/middleware htm

Navegación poco amigable para los usuarios.	Nueva tecnología Cliente/Servidor tres capas.	
Demasiados procesos duplicados y/o manuales, falta de estándares en el manejo de claves para trámites.	Análisis de procesos y reingeniería	
Gran concurrencia de operaciones	Optimizar conexiones a base de datos así como monitoreo de transacciones. Optimizar los recursos de sistema operativo.	Tuxedo es un middelware del tipo monitor de transacciones y optimiza los recursos del sistema operativo Puede manejar un sistema de tipo de geografía global.
Incongruencia en la información	Implementar transacciones para cada operación y así asegurar la integridad de información.	Tuxedo puede llevar de principio a fin transacciones entre diversas bases de datos y en diferentes plataformas.
Administración del sistema complejo	Contar con herramientas y automatización de administración	Tuxedo brinda herramientas de administración y autoadministración de la aplicación

Tabla 1.2 Resumen de Problemas y Oportunidades

5.1 Problemas Específicos del Módulo RFC

:s problemas especificos del módulo son·

Si un trámite es capturado de forma incorrecta, el sistema genera automáticamente los rechazos y no es posible cancelarlos En caso de que el operador se de cuenta de que cometió un error avisa a control de documentos mediante una relación emitida manualmente

- Cuando el departamento de sistemas corre procesos masivos provoca que la respuesta del sistema sea más lenta o incluso se detenga la operación
- De vez en cuando la operación de diagnóstico y procesos es interrumpida porque la relación de los sobres para entrega, tarda en ser impresa.
- Es muy lento el tiempo de respuesta para atender una solicitud que no llega a la administración correspondiente
- Las cifras de control arrojadas por el sistema no concuerdan con los controles llevados manualmente.
- Cuando un contribuyente es dado de baja como no localizado y efectúa algún aumento o disminucion de actividades el RFC es reactivado automáticamente y el trámite es realizado, sin validar información y solicitar en su caso un nuevo domicilio
- La infraestructura de comunicación y enlaces satelitales es lenta.
- Existen contribuyentes que no están registrados en las bases de datos nacional, regional y local (alrededor del 1%)
- Hay un 10% de rechazados iniciales (trámites no aceptados desde la validación local) al mes
- No hay forma de controlar que todos los documentos recibidos por diagnóstico sean capturados.
- Las Altas económicas (RFC de contribuyentes no encontrados en la base de datos a pesar de que ya se habían registrado) no generan necesariamente la misma homo clave.
- La información no sirve para la toma de decisiones, existen campos que no se utilizan o no pueden utilizarse
- Desfases en los tipos de actualización de las bases de datos local, regional y nacional, así como entre la información de las locales

A continuación se muestra en la tabla 1 3 el Resumen de Problemas y Oportunidades especificos del Módulo RFC

Datos incompletos, erróneos y duplicados		Limpieza de datos/cruce de datos con base de datos confiables
Desfases en los tiempos de actualización de las bases de datos local, regional y nacional, así como entre la información de las locales		Estrategias de replicación de información y comunicaciones
Falta de respuesta a los trámites de los contribuyentes		Esquema de atención al contribuyente en las Oficinas Locales con información de trámites Reingeniería al proceso de alta de los contribuyentes
Incongruencia entre el status real de los contribuyentes y la información que maneja el sistema		Integridad del sistema, cruce de información con otros módulos
La información no sirve para la toma de decisiones		Reingeniería/ Definición de las necesidades de información de los usuarios y áreas centrales
Existen campos que no se utilizan o no pueden utilizarse.		Rediseño de tablas y en general "objetos del negocio".

Tabla 1.3 Resumen de Problemas y Oportunidades del Módulo RFC

..6 Propuesta

1.6.1 Objetivos de Propuesta de Solución

El objetivo de la propuesta de solución es rediseñar los procesos relacionados con la administración tributaria en la parte recaudatoria, y construir un sistema informático que soporte las tareas de sus distintas áreas, cubriendo las necesidades de manejo de información tanto de usuarios operativos como analistas y autoridades.

El proyecto estará basado en una arquitectura global evolucionable, que le permita responder a los cambios permanentes en el entorno, que incluya un enfoque de apoyo al contribuyente, que permita cubrir las necesidades de integración de información y con otros sistemas, todo esto con el menor esfuerzo y en el menor tiempo posible, considerando los retos y necesidades del corto, mediano y largo plazo

1.6.1.1 Objetivos Globales

Entre los objetivos globales se tienen los siguientes

1. Rediseñar los Procesos

- Con base en una ponderación de diversos criterios (costo, oportunidades-retorno a la inversión, nivel de esfuerzo, cambios organizacionales o legales, etc.)
- Cuidar la confidencialidad e integridad de la información
- Garantizando su auditabilidad
- Con un enfoque hacia el contribuyente
- Lograr un mayor y más justo cobro de impuestos
- Optimizando los recursos que éstos consumen
- Para que sean mejor administrados (niveles de servicio, métricas)
- Apoyarse en la tecnología

2. Crear un Sistema Informático

- Verdaderamente integral.
- Evolucionable (pensando en el cambio permanente)
- Extensible (cobertura geográfica, procesos, liga con otros sistemas).
- Robusto, seguro y administrable.
- Ágil y sencillo de manejar (para todo tipo de usuario).
- Que permita obtener información operativa, analítica y ejecutiva
- Que se desarrolle modularmente y en poco tiempo (comparado con estándares del mercado)
- Que no este atado a plataformas tecnológicas cerradas o propietarias.

3. Transmitir y/o Reforzar Conocimientos y Habilidades al Personal para que:

- El enfoque al contribuyente sea un factor común
- Sean elemento activo en el cambio y se minimice su rechazo
- Puedan trabajar con la productividad esperada
- Sea mejorado su ambiente de trabajo.
- Estén conscientes de la importancia del proceso recaudatorio y de cada una de sus partes.
- Estén conscientes sobre todos los aspectos asociados a seguridad de información

1.6.2 Solución Conceptual Global

En esta sección se expone la Arquitectura Global propuesta para "El sistema Nuevo", esta arquitectura se detallara en el capítulo 2 de Arquitectura

Algunas de las principales ventajas de una Arquitectura Global son:

- Permite ver el sistema (incluyendo los procesos) y sus elementos a un nivel general.
- Permite tener un marco para construir cada elemento, de manera que interactúe con todos los demás
- Evita que los cambios y modificaciones a cada elemento se realicen sin ningún orden ni lineamiento, que pondría en peligro la interoperabilidad con los demás elementos, o dificultaría el mantenimiento y administración del elemento en cuestion

1.6.2.1 Bosquejo de la Arquitectura Ideal

Con base en los objetivos de "Sistema Nuevo", en los procesos y sistemas actuales, en las opiniones y sugerencias recabadas en todas las entrevistas realizadas, así como en sistemas y procesos del mismo rubro de otras partes del mundo, se ha concebido una arquitectura con los siguientes grandes bloques

- *Comunicación con el Contribuyente*: Conformado por todos los procesos que establecen una comunicación directa con el contribuyente, ya sea para darle algún tipo de información, para que él ejecute un trámite, o para comunicarle sobre algún adeudo pendiente
- Procesos/Sistemas Operaciones: se incluyen los módulos operativos del proceso principal de la AGR: pagos, control, créditos y cobranzas, etc.
- Administración de los Procesos y Reglas del Negocio: cuyo objetivo es incluir las tareas asociadas con modificar un proceso operativo (y los programas informáticos que lo soportan), o actualizar el sistema por cambios normativos-legales
- Contabilidad: independientemente a que los procesos operacionales incluyan un enfoque contable para los contribuyentes, la contabilidad es un proceso que recibe información de los demás bloques y modulos, y la procesa para obtener una consolidación
- Información de análisis y Ejecutiva, típicamente las consultas para usuarios de análisis o ejecutivos, involucran tratamientos temáticos de la información que no tienen que ver con la manera en que se maneja la información para los sistemas operacionales.
- Sistema Actual, que representa a todos los módulos y subsistemas que hoy conforman al SIR. El cual deberá convivir por un periodo de alrededor de dos años con el Sistema Nuevo, donde se buscará "encapsular" las funcionalidades del SIR para tener más libertad en la migración de los distintos módulos y bloques
- Otros Sistemas: Es indispensable que la nueva arquitectura, se consideren las interacciones que se tendrán con los sistemas de las áreas con las que se relaciona.

En la figura 1.5 se muestra el diagrama del bosquejo de la arquitectura ideal y su relación con su entorno.

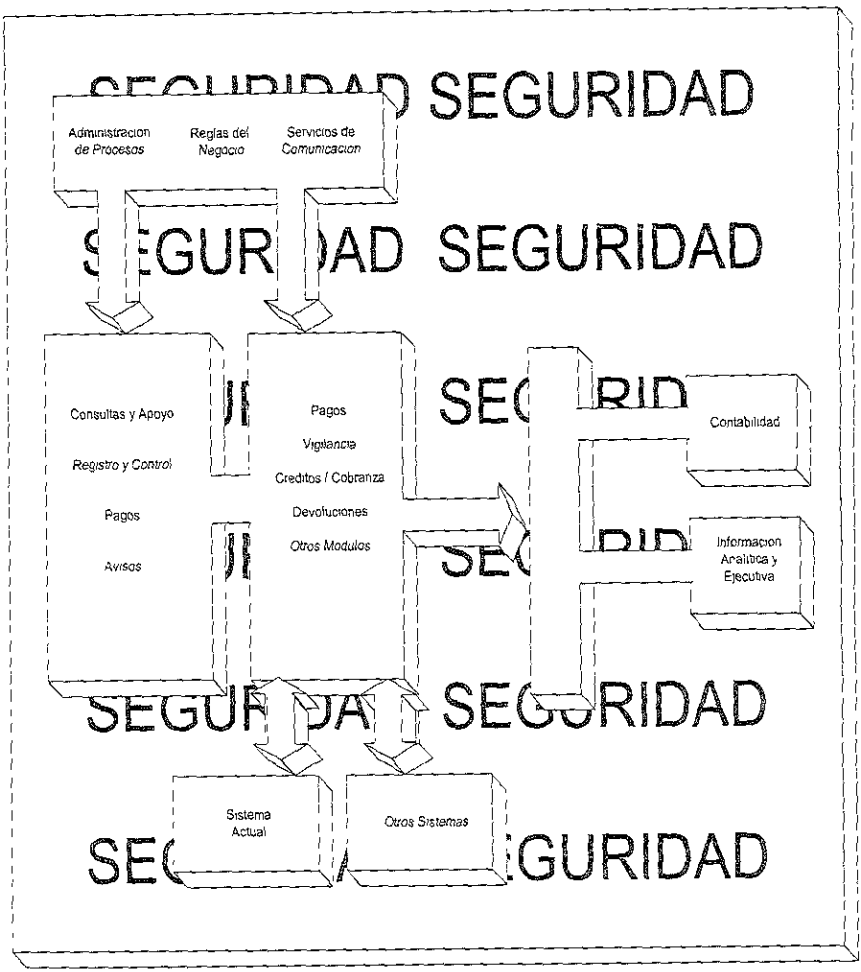


Figura 1 5 Sistema Nuevo y su entorno

1.6.2.2 Líneas Tecnológicas

Para soportar adecuadamente una arquitectura como la que se ha planteado para el "Sistema Nuevo", es necesario seleccionar, utilizar e implantar adecuadamente diversas tecnologías. En los siguientes párrafos revisaremos brevemente cada uno de los tres grupos de líneas tecnológicas que se han definido: Cliente/Servidor de varias capas e Infraestructura en general.

1.6.2.2.1 Cliente/Servidor de Varias Capas

Como una evolución de la arquitectura Cliente/Servidor de primera generación, han surgido en los últimos 5 años, diversas herramientas denominadas Cliente/Servidor de segunda generación. Las características más sobresalientes de este tipo de arquitecturas son: Particionamiento de las aplicaciones, Utilización de *Middleware*, Soporte a clientes en Internet, Administración de aplicaciones, Balance de cargas, Tolerancia a fallas.

Como se puede deducir de las características anteriores, la arquitectura Cliente/Servidor de segunda generación (también llamada de n-capas) significa algo más que elegir un front-end (clientes) y un manejador de base de datos. En concreto para su elección debemos considerar al menos:

- Los front-ends (clientes) de desarrollo y consulta que podremos utilizar.
- Los distintos tipos de *middleware* que soporta, sus posibilidades y el rendimiento asociado a cada uno de ellos, así como la interoperabilidad entre los distintos elementos.
- La forma de particionar la aplicación y de soportar reglas del negocio.
- Las facilidades de todo el ambiente a las distintas fases de un proyecto para el diseño, programación, implantación y para la administración.

- Cuenta con el *API* para programar las *reglas del negocio* en diferentes leguajes y plataformas
- Modularidad
- Rendimiento
- Independencia de datos
- Transparencia de localización
- Independencia de funciones
- Escalabilidad
- Alto rendimiento
- Confiabilidad en misión crítica
- Soporta estándares
- Ruteo dependiente a datos
- Balanceo de cargas
- Manejo de prioridades
- Replicación de servidores
- Tolerancia a fallas

Con las líneas tecnológicas se finaliza la solución global a la problemática y con esto finaliza también el capítulo 1.

En el siguiente capítulo detallaremos la Arquitectura Tecnológica que se plantea aquí como la solución Global al problema Ese capítulo estara enfocado a conceptos teóricos de la Arquitectura Cliente/Servidor como primer punto, posteriormente se describira la arquitectura final del "Sistema Nuevo" En los siguientes capítulos nos delimitaremos únicamente al Módulo de RFC

2. ARQUITECTURA TECNOLOGICA

Como sustento a la propuesta de solución, se va a tratar la parte teórica de la Arquitectura Cliente/Servidor desde sus inicios, así como su evolución a la Arquitectura Cliente/Servidor N capas detallando la parte del middleware que es la capa a la cual pertenece Tuxedo. Finalmente se detallará la Arquitectura implementada para el Sistema Nuevo Se empezará describiendo algunos conceptos básicos para poder entender el paradigma Cliente/Servidor

2.1 Conceptos Teóricos de la Arquitectura Cliente/Servidor

La tecnología Cliente/Servidor surge en la década de los 90's Esta tecnología se caracteriza por seguir una de las interacciones más simples entre programas de computadora que es el paradigma de petición/respuesta Un programa que hace la petición a otro programa de hacer algo. Al hacer la petición se puede enviar información o ejecutar cierta acción y reportar si tuvo exito o no la petición El programa al que le fue hecha la petición ejecuta la acción y generalmente regresa un resultado En este tipo de comunicación al programa que hace la petición se le llama "Cliente" y el programa que realiza la petición se le llama "Servidor" Este paradigma se muestra en la figura 2 1

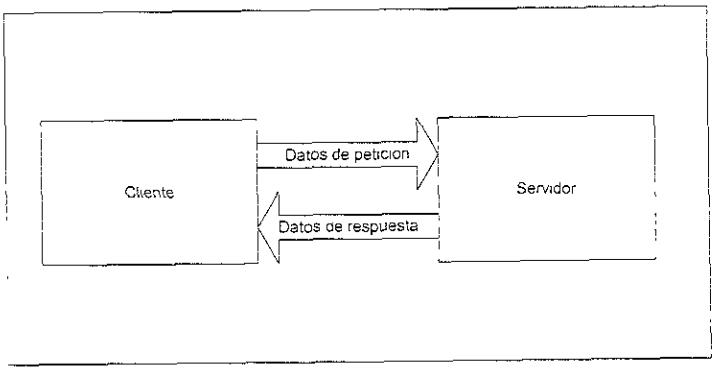


Fig 2 1 Interacción Petición/Respuesta

La tecnología Cliente/Servidor es un paradigma que se hace posible por una combinación de hardware de cómputo poderoso, de alta disponibilidad, rápido y relativamente bajo costo en la tecnología de comunicación Un número de conceptos, incluyendo sistemas distribuidos y proceso de transacciones distribuidas son asociados con la tecnología Cliente/Servidor Esta tecnología ha llevado a sistemas de cómputo de un simples, monolíticos y administrados centralmente a Sistemas distribuidos los cuales son rápidos, ya que es económico acceder a la información para cualquier casi cualquier persona.

La figura 2.2 ilustra como algunas empresas poderosas pueden usar la tecnología Cliente/Servidor. Por ejemplo, un Banco, en cada oficina bancaria se puede tener una computadora local con únicamente datos necesarios, mientras la oficina central mantiene archivos centrales en un gran conjunto de servidores Un sistema Cliente/Servidor apropiadamente diseñado puede dar servicio cientos de usuarios con alto rendimiento Un claro ejemplo es el uso de cajeros automáticos donde cada cajero funciona como cliente y cada opción del menú dispara la petición de un servicio Si el cliente pidiera retirar dinero y no nos encontráramos en la ciudad donde se abrió la cuenta bancaria, probablemente la peticion se ejecutaría en un servidor remoto, pero esto es transparente para el cliente el cual obtendrá como resultado de éxito el dinero y como resultado fallido un mensaje de saldo insuficiente

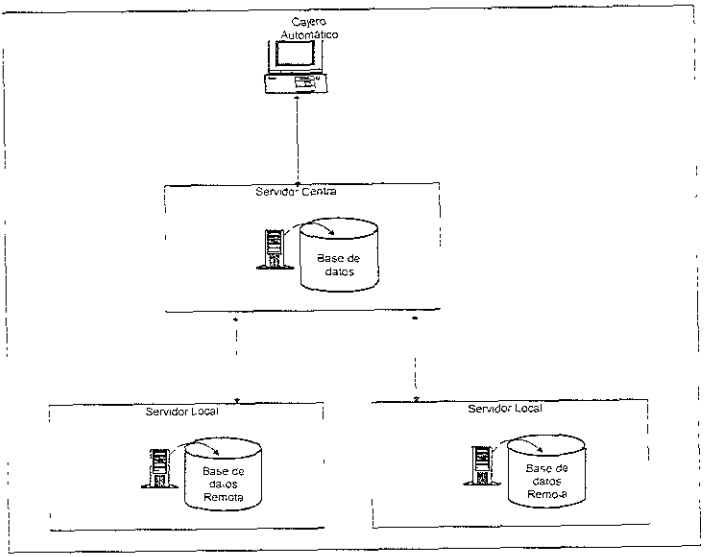


Figura 2 2 Sistema Cliente/Servidor

La tecnología Cliente/Servidor se ha hecho popular gracias a las siguientes características:

- Permite el uso de hardware de bajo costo.
- Soporta escalabilidad.
- Soporta sistemas tolerantes a fallas
- Brinda mayor facilidad para la administración de datos.
- Soporta interfase gráfica para el usuario *front ends*

2.1.1 Definiciones Básicas de Cliente/Servidor

A continuación se definen los términos más importantes que se utilizan en los sistemas Cliente/Servidor que son las que fundamentan las decisiones tomadas para la arquitectura del Sistema Nuevo

Programa

Un programa es una unidad de software, representa una entidad pasiva

Proceso

Un proceso es la entidad de ejecución reconocida por el sistema operativo, es un programa en ejecucion Esta ejecución es secuencial Un proceso requiere de recursos (memoria, CPU, dispositivos de E/S, stack) para su ejecución

Cuando un programa es reconocido por el sistema de operación y tiene asignado recursos, éste se convierte en proceso.

Sistema Distribuido

Un sistema distribuido es la integracion de un conjunto de servicios informáticos repartidos en un grupo de computadoras autónomas conectadas a través de una red que presentan una conexion transparente al usuario Estas computadoras autonomas están equipadas con un software distribuido que les permite coordinar sus actividades

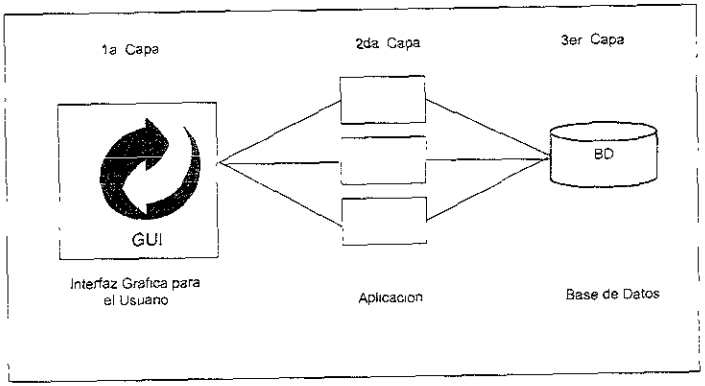


Figura 2 3 Sistema Distribuido

En la Figura 2.3 podemos ver como diversos equipos de cómputo están interactuando transparentemente a través de la red Por ejemplo, si alguien estuviera desde su computadora personal conectado a una estación de trabajo podría imprimir algún documento sin saber realmente quién está atendiendo esa petición de impresión

Proceso Distribuido

Proceso Distribuido significa que el proceso de una unidad de la aplicación de trabajo utiliza más de un proceso independiente de computadora, estos procesos lo son de la aplicación y no son parte del sistema operativo, la base de datos u otro sistema de soporte. De hecho, el proceso distribuido es definido como una habilidad de sistema definidor por el desarrollador de la aplicación, ya que los procesos distribuidos de soporte, incluyendo el procesamiento de la base de datos, no son de la aplicación del desarrollador, esto es que no forman parte del sistema aplicativo. Por ejemplo si se hiciera la petición de retiro de dinero desde un cajero. el proceso de retirar dinero podría dividirse en dos procesos independientes, uno que se encargará de validar que el número de cuenta esté autorizado a retirar el dinero y otro proceso que actualice la base de datos con un saldo menor al que se tenía antes de hacer el retiro Estos dos procesos son independientes y son procesos aplicativos que forman parte de una unidad específica que es retirar dinero.

Procesos Cliente/Servidor

El proceso cliente manda peticiones al el proceso servidor, el cual responde con resultados a esas peticiones El proceso servidor provee servicios a sus clientes, usualmente de manera de procesos específicos que sólo ellos pueden hacer La interaccion entre el cliente y el servidor es cooperativa, y transaccional en el cual el cliente es proactivo y el servidor es reactivo

Únicamente cuando el proceso de una aplicacion es distribuida. el sistema puede ser escalable y permitir el uso de múltiples plataformas

El proceso distribuido no significa que "varios pasos de trabajo" se puedan hacer por diferentes procesos, como lo hacen los mainframes Es proceso distribuido sólo cuando una unidad sencilla de trabajo está siendo hecha por múltiples procesos. Entendamos a unidad de trabajo como una operacion específica, por ejemplo "Alta de un Proveedor" en un sistema de Compras

La figura 2 4 ilustra el procesamiento distribuido Una unidad simple de trabajo de aplicación usa más de un proceso (en este caso, dos) Tales procesos cooperan para completar la tarea por medio de una comunicacion de protocolo. Cada proceso de aplicación podría acceder a bases de datos independientes una de otra. Las bases de datos podrían o no ser del mismo tipo.

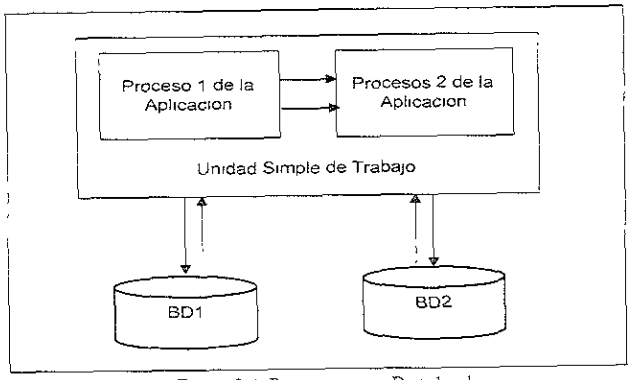


Figura 2 4 Procesamiento Distribuido

Base de Datos Distribuida

Una base de datos puede ser definida como una colección de información permanentemente almacenada en una computadora y accesible por medio de una aplicación

En una base de datos distribuida, los datos estan divididos entre más de una instancia. una instancia puede definirse como un grupo de objetos de base de datos, pero puede ser tratado como una simple base de datos lógica por la aplicacion Una base de datos distribuida puede existir en la misma plataforma o, usualmente en múltiples plataformas. El concepto de una base de datos distribuida es esencial para entender la importancia del manejo de transacciones

La característica principal que tienen las bases de datos distribuidas es la habilidad de acceder datos ubicados en múltiples sitios. localizados en diferentes nodos de una red, en forma transparente a los usuarios finales Figura 2.5

Un sistema de bases de datos distribuidas correctamente implementado, permite manejar un elevado volumen de datos, así como múltiples conexiones concurrentes de las aplicaciones de los usuarios

Las ventajas que ofrece el uso de bases de datos distribuidas son

1. Permite una mayor disponibilidad de información
2. Evita pérdida de información por fallas al tener copia de los datos en diferentes localidades.
3. Permite replicación y fragmentación de información
4. Aumenta autonomía local
5. Permite respaldos en línea

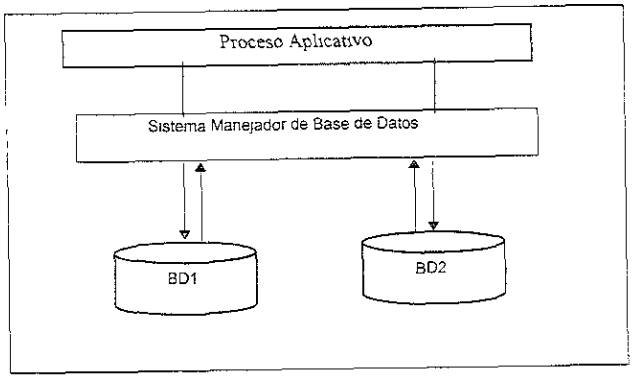


Figura 2 5 Base de Datos Distribuida

2.2 Definición General de Cliente/Servidor

"El término Cliente/Servidor se refiere a la relación entre dos entidades (sistemas o procesos) El cliente es el sistema que hace una petición de trabajo a el sistema servidor En la mayoría de la situaciones, quién es el cliente y quién es el servidor se hace a través de la relación del que solicita "[1].

Los servidores brindan servicios a los clientes que los solicitan Aquí es importante distinguir el término servidor del término servicio Se va a definir a un sistema o proceso que recibe peticiones como el servidor y a un sistema o proceso que envía peticiones como el cliente. Algunas veces un servidor puede enviar una petición a otro servidor, pero este punto se describe más adelante.

El cliente y el servidor pueden existir o no en máquinas físicas distintas Un cliente puede tener una relación con diferentes servidores y un servidor puede satisfacer peticiones de múltiples clientes. La relación entre el cliente y el servidor se lleva al cabo por medio de transacciones que consisten de peticiones bien definidas y de su respuesta

2.2.1 Servidores Hardware

Se ha hecho común hoy en día para casi cualquier usuario que esté conectado a una red, el contar con una variedad de servicios de uno o más sistemas en la red. En este caso, la máquina que provee el servicio es un servidor Normalmente no hay interacción directa entre el usuario final y la máquina servidor La interacción del usuario es ejecutada con algún otro hardware, la computadora personal del cliente, el cual puede ser una computadora de un tipo completamente diferente al hardware que le está brindando el servicio, el servidor

[1] Definicion de Cliente/Servidor Technical Foundations of Client/Server Systems Pag 7
Carl L Hall, John Wiley & Sons, Inc 1994

El sistema operativo y el software de red hacen las funciones de un servidor de hardware. En otras palabras, una vez que una máquina es definida como servidor, la máquina necesita software especializado para ser un servidor Algunos ejemplos de este tipo de servidores son

1 Servidor de impresión, con impresoras asociadas a los servidores y usados por todos en la red, figura 2 6
2. Servidor de archivos, con archivos comunes almacenados en un servidor
3 Mecanismo de almacenaje dando servicio a estaciones de trabajo que tienen poca o ninguna capacidad de almacenaje

Cuando un servidor de hardware está dando servicio, este es transparente para el usuario y usualmente tiene poco o no tiene efecto en el diseño de la aplicación Este tipo de servidores no se va a detallar ya que el objetivo de este trabajo está orientado a servidores aplicativos, esto es, que han sido diseñados y construidos con un objetivo específico.

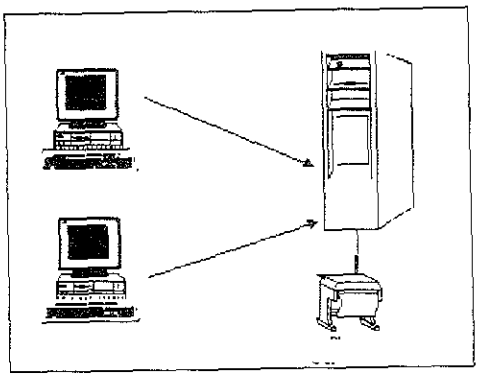


Figura 2 6 Servidor de Impresion

2.2.2 Servidores Software

Los servidores de software son brindados por software diseñado específicamente para manejo de peticiones. Este software especial es frecuentemente brindado en un servidor de hardware Es común encontrar que el hardware brinda el software de servidor y también el software de clientes

Los servicios brindados por los servidores de software frecuentemente son de alguno de los siguientes tipos.

- Servicios de archivos de datos
- Servicios de llamadas remotas a procedimientos
- Servicios de base de datos
- Sistemas Mejorados Cliente/Servidor, los cuales brinda una capacidad real de proceso distribuido.

A continuación se detalla cada uno de estos servidores de software.

1. Servidores de Archivos

Con un servidor de archivos, el cliente, generalmente una computadora personal, pasa peticiones para obtener registros de un archivo sobre una red al servidor de archivos Esta es una forma muy primitiva de servicio de datos que necesita muchos intercambios de mensajes sobre la red para encontrar los datos pedidos Los servidores de archivo son muy usados para compartir archivos a través de la red Son indispensables para crear repositorios de documentos compartidos, imágenes, dibujos de ingeniería, y otros objetos grandes de información

- Servidores de Bases de Datos

Con un servidor de base de datos, el cliente pasa su petición SQL (Structured Query Language) como un mensaje al servidor de base de datos. El resultado de cada comando SQL es regresado sobre la red El código que procesa la petición SQL y los datos residen en la misma máquina. El servidor usa su propio poder de procesamiento para encontrar los datos requeridos en lugar de pasar todos los registros de regreso a cliente y dejar que el cliente encuentre los datos que requiere, como es el caso de un servidor de archivos El resultado es un uso mucho más eficiente del poder de proceso distribuido Generalmente es necesario escribir código para la aplicación del cliente o se puede comprar paquetería Los servidores de base de datos proveen fundamentos para sistemas de apoyo en decisiones que requiere consultas hechas a la medida y reportes flexibles, por ejemplo el reporteador de "Cronos" (Cronos es un producto comercial) Otro ejemplo de obtención de datos mediante una sentencia SQL es dentro de un stored procedure (procedimiento almacenado en la base de datos)

- Servidores Transacionales

Con un servidor transaccional el cliente invoca procesos remotos que residen en el servidor con un motor de base de datos *SQL* , estos procesos remotos en el servidor ejecuta un conjunto de sentencias *SQL* El intercambio en la red consiste de una comunicación sencilla de peticion/respuesta al mensaje (al contrario del enfoque de los servidores de base de datos de una peticion/respuesta al mensaje para cada sentencia SQL en una transacción) Las sentencias de *SQL*, todas fallan, o todas tienen exito.

Estas sentencias de *SQL* agrupadas se llaman transacciones

Con un servidor transaccional se crea la aplicación Cliente/Servidor escribiendo el código tanto del cliente como del servidor. El componente cliente usualmente incluye una interfase gráfica . El componente servidor usualmente consiste de transacciones de SQL contra una base de datos. Estas aplicaciones tienen un nombre "Proceso de Transacción en Línea " OLTP (On Line Transaction Processing). Ellas tienden a ser aplicación de misión crítica que requieren respuesta de uno a tres segundos el 100% de las veces

Las Aplicaciones OLTP, además requieren fuertes controles sobre la seguridad e integridad de la base de datos.

- Servidores Groupware

Una nueva clase de sistema está surgiendo hoy en día para dirigir el manejo de información semiestructurada, como texto, imagen, correo y flujo de trabajo

Estos nuevos sistemas Cliente/Servidor, colocan a la gente en contacto directo con otra gente Lotus Notes es el ejemplo líder de tal sistema Software de groupware especializado puede ser construido sobre un conjunto de APIs (Aplication Programming Interface) Cliente/Servidor. En la mayoría de los casos las aplicaciones son creadas usando un lenguaje script e interfaces basadas en formas distribuidas por el proveedor El middleware de comunicación entre el cliente y el servidor son definidas por el vendedor.

- Servidores de Objeto

Con un servidor de objeto, la aplicación Cliente/Servidor es escrita como un conjunto de objetos que se comunican entre sí. Objetos cliente se comunican con los objetos servidor usando un agente de petición de Objeto ORB (Object Request Broker) El cliente invoca un método brindado por una clase de objeto servidor La ORB localiza la clase objeto servidor, involucra el método requerido y da los resultados al cliente objeto

2.2.3 Integridad Transaccional

Una transacción es definida como un conjunto de acciones, la transacción se completa exitosamente si todas las acciones inserciones, actualizaciones o borrados de registros fueron totalmente correctas, si alguna de ellas fallara las acciones que se hubieran hecho antes deben regresarse a su estado original

La integridad transaccional en la base de datos requiere que la base de datos debe permanecer consistente con las reglas del negocio todo el tiempo En el siguiente capítulo detallaremos como se asegura la Integridad Transaccional

2.2.4 Características del Modelo Cliente/Servidor

Atributos del Cliente

El proceso cliente es proactivo, ejecuta peticiones al servidor, generalmente esta dedicado a la sesión de su usuario y empieza y termina con la sesión. Un cliente puede interactuar con un servidor o con múltiples servidores para completar su trabajo. Siempre es necesario por lo menos un servidor.

A nivel aplicativo, el cliente es responsable de mantener y procesar la interfase de diálogo con el usuario. Esto generalmente incluye la ejecución de lo siguiente:

- Manejo de pantalla.
- Menú.
- Captura y validación de datos.
- Proceso de ayuda
- Recuperación de errores.

En una aplicación gráfica también incluye:

- Manejo de ventanas.
- Captura de mouse y teclado
- Control de cajas de diálogo.
- Administracion de video y sonido (en aplicaciones multimedia)

En la práctica, el cliente no siempre ejecuta la validación de datos. En algunas de las validaciones de la aplicación se requiere algún proceso para validar los datos. Un cliente robusto también implementa lógica del negocio asociada con una aplicacion, mientras un cliente sencillo soporta solamente la lógica de presentación y validación mínima de datos.

Para administrar toda la interacción con el usuario, el cliente esconde efectivamente el servidor y la red del usuario, esto es que para el usuario lo que pase cuando él hace una petición es transparente, únicamente recibe el resultado. Esto crea la ilusión de que la aplicación entera es ejecutada localmente sin el uso de otros procesos, máquinas o redes.

Atributos del Servidor

El proceso del servidor es reactivo, esto es que ejecuta su proceso a la llegada de una petición de sus clientes Un proceso servidor generalmente está en espera de una petición siempre, ofreciendo servicios a muchos clientes Esos servicios pueden ser brindados directamente por el servidor o indirectamente por un proceso esclavo generado por cada petición del cliente. La Figura 2.7 muestra el uso de procesos maestro y esclavos para brindar un servidor lógico para un cliente Por ejemplo, el servidor de base de datos Oracle siempre está a la espera de peticiones del cliente. Cuando una petición llega, el servidor de Oracle genera un proceso esclavo dedicado a manejar esa petición, permitiendo que el proceso maestro reciba otra petición inmediatamente

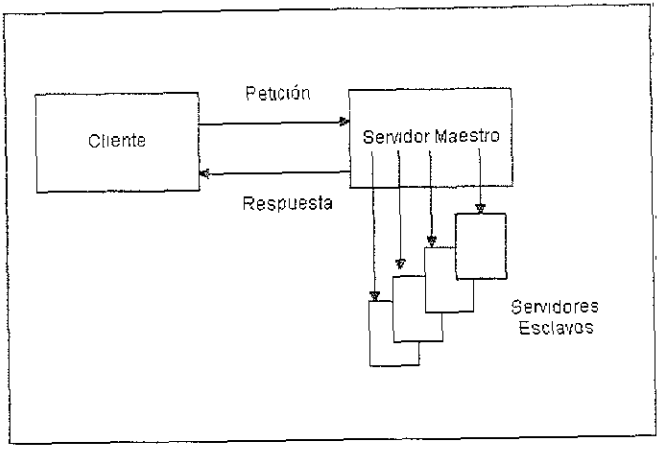


Figura 2 7 Servidores Esclavos

Un servidor tiene una función específica, ejecuta un conjunto de transacciones predefinidas, relacionadas funcionalmente Generalmente existen múltiples servidores para proveer una operación (alta de suscriptor, por ejemplo), para los clientes Potencialmente, múltiples servidores pueden acceder a un servidor al mismo tiempo El servidor debe resolver cualquier situación, por ejemplo el bloque de un registro, donde nadie puede actualizar éste hasta que el servidor termine de utilizarlo, esto para prevenir corrupción en el resultado

El servidor ejecuta toda la logica requerida para procesar una petición y no necesita de la interacción de otros servidores (esta es una afirmación es limitada, mas adelante en este capítulo se especifica esta afirmación) Generalmente, las peticiones son tratadas de modo transaccional (esto es, una unidad indivisible de trabajo)

Por ejemplo, si la petición es agregar un nuevo empleado a la base de datos, el servidor podría actualizar varias tablas para mantener la integridad de la información. Ya que se requieren varias inserciones y actualizaciones, el servidor debe asegurarse que todos las operaciones o ninguna de ellos sea aplicado en la base de datos para mantener la integridad de la información Si el servidor soporta procesos concurrentes, debe asegurarse de que las operaciones requeridas son aisladas y que no interfieren con las operaciones requeridas para satisfacer otra petición concurrente

Si un cliente usa múltiples servidores, es su responsabilidad invocarlos como sean requeridos. El proceso servidor generalmente incluye todos los procesos asociados con acceder, almacenar y organizar datos compartidos. actualizar previamente datos almacenados y cualquier administracion de otros recursos compartidos Los recursos compartidos pueden ser datos, CPU, almacenamiento en disco o cinta, comunicación, y a veces administración de memoria

Atributos de comunicación

El estilo de comunicación entre el cliente y el servidor es transaccional y cooperativa Una característica de este estilo de comunicación es que el servidor regresa sólo el resultado relevante a la petición del cliente Esto contrasta con el estilo conversacional donde ambos procesos intercambian cualquier número de mensajes en una misma sesión con conversación (se retomará el término de conversación en el siguiente capítulo) Idealmente la cantidad de datos enviados es al menos la cantidad necesaria para que el cliente termine su trabajo

Las aplicaciones Cliente/Servidor sí impactan la carga de la red, pero es el menor impacto de cualquier alternativa de procesamiento en red En la implementación Cliente/Servidor, el cliente es responsable por el control de la pantalla, y no de la información viaja por la red El cliente transmite sólo datos validados y recibe sólo los datos necesitados para desplegar en los campos de la pantalla. Mucho menos cantidad de datos son transitan por la red

La comunicación Cliente/Servidor generalmente es implementada usando llamadas remotas a procedimientos (Remote Procedure Calls, RPCs) En una invocacion RPC, el cliente para el proceso después de hacer una petición al servidor y espera por una respuesta El servidor empieza a procesar sólo a la peticion de un cliente y cesa después de cumplir esa peticion Cuando el cliente recibe la respuesta reanuda su proceso.

Al servidor podrían requerírsele peticiones simultáneas de múltiples clientes. Por lo que una petición podría llegar mientras el servidor aun está atendiendo otra peticion de otro cliente. El servidor debe proveer facilidades de encolar esas peticiones o de procesarlas concurrentemente Por ejemplo el servidor puede procesar peticiones concurrentemente creando un proceso hijo para atender cada petición En este caso el

servidor debe ser muy cuidadoso en regresar la respuesta al cliente correcto. Otra manera de atender peticiones concurrentes, es la replicación de servicios, esto es, brindar varias copias de un servidor

2.2.5 Carga del Sistema Cliente/Servidor

Hasta ahora se ha mostrado que los modelos Cliente/Servidor pueden ser clasificados a través del servicio que brindan Las aplicaciones Cliente/Servidor se pueden clasificar a demás por cómo está distribuida la aplicación entre el cliente y el servidor figura 2.8. El modelo servidor robusto pone más funcionalidad en el servidor. La funcionalidad o reglas del negocio se refieren al código programado donde se llevan la lógica de la aplicación. El modelo cliente robusto hace lo contrario. Servidores Groupware. Transacción y Objeto son ejemplos de servidores robustos Servidores Base de Datos y Archivo son ejemplo de clientes robustos.

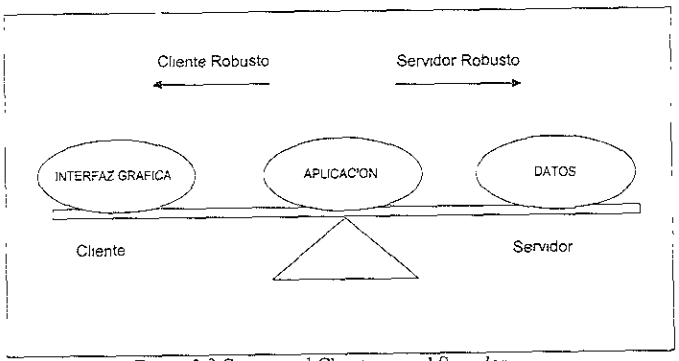


Figura 2.8 Carga en el Cliente o en el Servidor

Los clientes robustos son la forma más tradicional de arquitectura Cliente/Servidor El grueso de la aplicación se lleva a cabo del lado del cliente En ambos, tanto en los modelos de servidor archivo y el servidor base de datos los clientes saben como están organizados y almacenados los datos en el servidor cliente. Los clientes robustos se usan como apoyo a decisiones y software personal Brindan flexibilidad y oportunidades para crear herramientas de interfase grafica que permiten a los usuarios finales crear sus propias aplicaciones

Las aplicaciones de servidor robusto son más faciles de manejar y desplegar en la red porque la mayoría de código se lleva en los servidores. Los servidores robustos intentan minimizar los intercambios de la red creando niveles de servicio más abstractos Servidores Transacciones y de Objetos, por ejemplo. encapsulan la

base de datos. En lugar de exportar datos simples o datos primarios, exportan los procedimientos (o métodos en metodología orientada a objetos) que operan sobre los datos El cliente en el modelo de servidor robusto provee la interfase gráfica e interactúa con el servidor a través de llamadas de procesos remotos (o invocaciones de métodos)
Cada modelo Cliente/Servidor tiene sus propios usos En muchos casos los modelos se complementan unos a otros y no es extraño hacer que coexistan en una aplicación.

2.3 Transición de 2 Capas a 3 Capas

Una vez que se ha mostrado la arquitectura Cliente/Servidor tradicional, hablaremos de la revolución en que esta arquitectura se ha desarrollado, que la ha transformado de 2 capas a una arquitectura 3 capas, esto es, la necesidad de involucrar una capa intermedia entre el Cliente y el Servidor. El impacto de este cambio algunos lo consideran más dramático que el paso de aplicaciones monolíticas a Cliente/Servidor. Esto ha surgido por la necesidad de hacer que las aplicaciones Cliente/Servidor trabajen para las aplicaciones "mejoradas" (Internet, intranet, objetos distribuidos, y de misión crítica)
Se usará el término de capas para describir la partición lógica de una aplicación a través de los clientes y servidores. Separar la carga de proceso es un concepto central de Cliente/Servidor Las capas describen dos opciones de arquitectura:

2-Capas: divide la carga del proceso en dos La mayoría de la lógica de la aplicacion se ejecuta en el cliente, el cual generalmente manda peticiones *SQL* a un servidor de base de datos Esta arquitectura la definimos como de cliente robusto.

3-Capas Divide la carga del proceso en tres 1)clientes ejecutando la lógica de la interfase gráfica para el usuario, 2) servidor de aplicación, ejecutando la lógica de la aplicación, 3) la base de datos. Dado que la segunda capa lleva la lógica de la aplicación al servidor, se define como un servidor robusto

Por definicion, toda aplicacion Cliente/Servidor debe tener al menos dos capas la interfase de usuario que reside en el cliente y los datos compartidos almacenados en el servidor Como acabamos de ver, una aplicacion es de 2 capas o de 3 capas basado en la separación de la lógica de la aplicación, de la interfase gráfica y de base de datos

Esta separación es el diseño central que hace la gran diferencia en determinar el éxito de aplicaciones de misión crítica ya que un mal diseño puede no explotar las características de software dedicado a aplicaciones de 3 capas. Esta partición se ilustra en la figura 2.9.

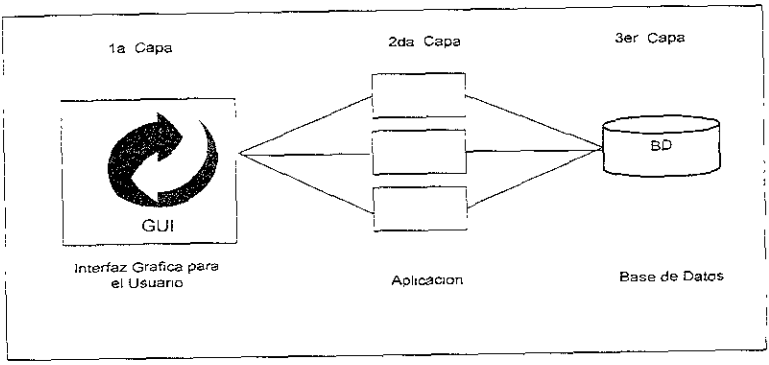


Figura 2 9 Aplicación 3 Capas

Los arquitectos, que son los encargados de diseñar correctamente una aplicación tomando en cuenta los productos de software con los que se cuentan y las características de la aplicación, descubrieron que la arquitectura y herramientas que usaron para aplicaciones 2-capas no soportaban aplicaciones de misión crítica. Aplicaciones que trabajaron perfectamente bien en prototipos e instalaciones fallaron cuando se enfrentaron a producción de gran escala. El modelo de 2 capas no es de misión crítica Por consecuencia, las aplicaciones fueron lentas. Aunque esto podría señalar una falla en la arquitectura Cliente/Servidor, lo que demuestra es que se tiene que hacer una transición. Ahora se desarrollan aplicaciones muy grandes y comercio electrónico Consecuentemente tenemos que dejar en el pasado el modelo 2-capas y separar las aplicaciones en componentes distribuidos a través de múltiples procesadores - aplicaciones de 3 capas (y N-Capas) .

2.3.1 Aplicaciones 2-Capas vs. Aplicaciones 3-Capas

En la figura 2 10 se muestra el porque se ha llevado las aplicaciones a N-capas, cómo es que las aplicaciones ha tenido que crecer para soportar un mundo interconectado

Número de clientes por aplicación	Menos de 100	Millones
Número de servidores por aplicación	1 o 2 servidores homogéneos	Más de 100,000, con servidores heterogéneos ejecutando diferentes rolles
Geografía	Local	Global
Interacciones servidor a servidor	No	Sí
Middleware	SQL y procesos almacenados (stored procedures)	Componentes en Internet e intranets, así como middleware especializados.
Arquitectura Cliente/Servidor	2-Capas	3-Capas
Actualizaciones Transaccionales	Poco frecuente	Siempre
Contenido multimedia	Bajo	Alto
Interfase gráfica (Front Ends)	Clientes Robustos	Clientes en demanda
Tiempo de Contexto	1885 y presente	1997-2000 y futuro

2.10 Cliente/Servidor Local vs. Cliente/Servidor Global

En el modelo de 3 capas, el cliente provee la interfase gráfica e interactúa con el servidor a través de servicios remotos. La lógica de la aplicación vive en la capa media y es de su dominio Aquí se maneja y despliegan los procesos separadamente de la interfase del usuario y de la base de datos La lógica de la aplicación ahora tiene su propia capa y corre en uno o más servidores

El modelo 3-capas es la área creciente de los sistemas Cliente/Servidor porque es donde se reúnen los requerimientos de aplicaciones de gran escala de internet e intranet y otras aplicaciones de misión crítica. Estas aplicaciones son más fáciles de manejar y desplegar en la red la mayoría del código corre en el servidor. Adicionalmente las aplicaciones de 3-Capas minimizan los intercambios en la red creando niveles abstractos de servicio. En lugar de interactuar con la base de datos directamente, el cliente llama a la lógica del negocio en el servidor La logica del servidor accede a la base de datos El modelo de 3-Capas sustituye muchas sentencias SQL por pocas llamadas a servidores , por lo que la ejecucion es mucho mejor que en 2-Capas También brinda mejor seguridad ya que no expone el esquema de la base de datos al cliente y puede delimitar autorización en el servidor

Cuando los modelos Cliente/Servidor crecen para correr aplicaciones de misión-crítica se pueden observar las siguientes características entre 2 y 3 capas, figura 2 11

Administración del Sistema	Compleja (más lógica en el cliente para administrar)	Menos Compleja (La aplicación se puede administrar centralmente desde el servidor- los programas de la aplicación se ha hecho con estándares)
Seguridad	Baja (Seguridad a nivel datos)	Alta (Bien optimizada a servicios o nivel de métodos)
Encapsulación de datos	Bajo (Las tablas están expuestas)	Alto (El cliente involucra servicios o métodos)
Ejecución (Performance)	Pobre (muchas sentencias SQL se envían sobre la red, los datos seleccionados tienen que ser analizados por el cliente)	Alta (Solamente las peticiones y respuestas son enviadas entre el cliente y el servidor)
Balanceo	Pobre (Administración limitada)	Excelente (puede distribuir cargas a múltiples servidores, balanceo de cargas)
Fácil Desarrollo	Alto	Mejorando (Se pueden usar herramientas estandares para crear los clientes, y existen herramientas que pueden usarse para desarrollar el cliente y servidor de la aplicación)
Infraestructura servidor a servidor	No	Sí (vía middleware)
Integración de la Aplicación	No	Sí (vía gateways encapsulados por servicios u objetos)
Soporta en Internet	Pobre	Excelente (Clientes no robustos son más fáciles de adecuar, existen aplicaciones distribuidas de servicios remotos que están en el servidor)

Soporte de base de datos Heterogéneas	No	Sí (Las aplicaciones de 3-capas pueden usar múltiples bases de datos dentro de la misma transacción)
Variadas opciones de Comunicación	No	Sí (soporta llamadas como RPC's, pero puede soportar mensajes sin conexión, encolamiento. publicación y suscripción de servicios y publicación de mensajes).
Flexibilidad de Arquitectura de Hardware	Limitado (se tiene un cliente y un servidor)	Excelente (todas las 3 capas pueden residir en computadoras diferentes, o la segunda y tercer capa puede residir en la misma computadora Los componentes de la 2da. Capa también pueden distribuirse en múltiples servidores).
Disponibilidad	Pobre (No tiene un servidor respaldo para fallas)	Excelente (pueden reestablecerse los componentes de la capa del middleware en otros servidores):

Figura 2.11 Tabla comparativa entre arquitectura 2 y 3 Capas

2.3.1.1 Cuando Usar las 3-Capas

Aunque las aplicaciones 2-capas son más fáciles de desarrollar que las de 3-capas, esto puede cambiar exponencialmente si la aplicación se empieza a volver más compleja

Se puede determinar que una aplicación no es de 2-capas, sino de 3-capas si se tiene alguna de las siguientes características

- Existen muchos servicios o clases aplicativos, más de 50
- Las aplicaciones están programadas en diferentes lenguajes o desarrollados por diferentes empresas.
- Existen dos o más bases de datos heterogeneas.
- La vida de la aplicacion es mayor a 3 años, especialmente si se espera muchas modificaciones o adiciones
- Existe gran volumen de peticiones, mas de 50,000 transacciones por día o más de 300 usuarios concurrentes en el mismo sistema usando la misma base de datos.
- Comunicacion significativa inter aplicación.
- La probabilidad de que el sistema crezca y pueda cubrir alguno de los puntos anteriores.

2.4. Segunda Capa

2.4.1. Componentes en la Segunda Capa

La segunda capa de la mayoria de las aplicaciones 3-capas no esta implementado como un programa monolitico En su lugar, está implementado como una colección de componentes que se usan en una variedad de transacciones de negocios iniciadas por el cliente, figura 2.12 .

Cada componente automatiza una función relativamente pequeña del negocio Los clientes frecuentemente combinan varios componentes de la segunda capa para una transacción sencilla Un componente puede llamar a otros componentes para completar una petición

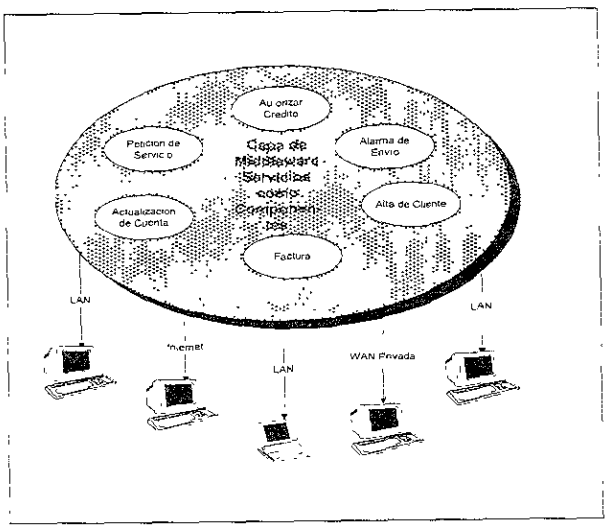


Figura 2 12 Segunda Capa

2.4.1.1 Beneficios de las Arquitecturas Basados en Componentes

- Se pueden desarrollar aplicaciones grandes en pasos pequeños. Las arquitecturas basadas en componentes permite desarrollar aplicaciones de misión crítica grandes como proyectos pequeños Cuando se usa este método de desarrollo por pasos, se puede generar versiones iniciales de la aplicación en producción más rápidamente Esto reduce el riesgo

- Las aplicaciones pueden reusar componentes A diferencia de los lenguajes orientados a objetos que se enfocan en el reuso del código fuente, aquí se pueden reusar los componentes como "cajas negras". Se pueden combinar estos componentes en formas variadas, dependiendo de la aplicación

- Los clientes pueden acceder a los datos y funciones de una manera fácil y segura. Los clientes mandan peticiones a los componentes para ejecutar una función en lugar del cliente. Los componentes del servidor "esconden" los detalles de la logica de la aplicación Los clientes no necesitan saber qué base de datos se está accediendo para ejecutar la petición. Y los cliente no necesitan saber si la petición fue mandada a otro componente u otra aplicación para la ejecución Esto comportamiento provee consistencia, seguridad acceso auditable a los datos y elimina actualizaciones no controladas de muchas aplicaciones al mismo tiempo

- La aplicaciones hechas a la medida pueden incorporar componentes no propios Existen componentes ya hechos que pueden convivir con otros hechos por un vendedor de software diferente

- Los componentes con el tiempo se hacen mejores. Los sistemas basados en componentes pueden crecer Se pueden añadir aplicaciones muy rápidamente construyendo nuevos clientes, añadiendo nuevos componentes en el middleware y reusando componentes existentes. Se pueden actualizar componentes sin cambiar los clientes

2.4.2 Definición de Middleware

Middleware es un término que cubre todo el software distribuido necesario para soportar interacciones entre los clientes y los servidores. Es el software que está en medio de los sistemas Cliente/Servidor Esta capa es la que le permite al cliente obtener un servicio del servidor El middleware empieza en el conjunto API del lado del cliente que es usado para invocar un servicio y cubre la transmisión de la petición sobre la red y de la respuesta del resultado. El middleware no incluye el software que provee el servicio actual -él software está en el dominio del servidor No incluye la lógica de la interfase de la aplicación -ésta está en el dominio del cliente

Para ubicar a la segunda capa dentro de una arquitectura de 3 capas, se muestra su posición dentro de la arquitectura base en la figura 2 13.

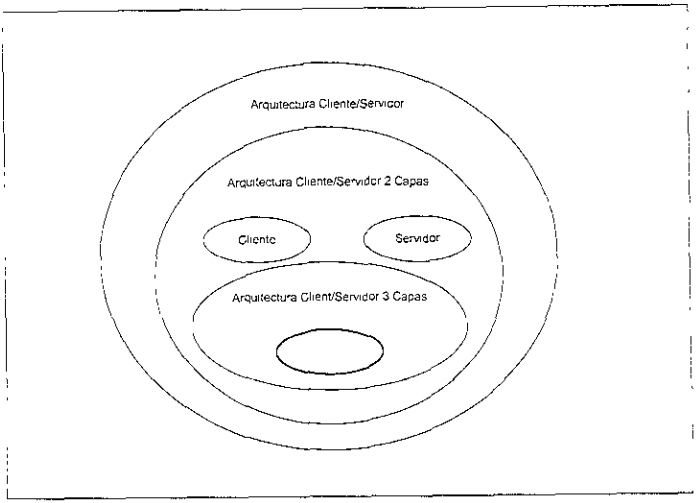


Figura 2 13 Posición del middleware en la arquitectura Cliente/Servidor

El middleware se puede ver agrupado en dos clases

1. Middleware General

Se refiere a la mayoría de las interacciones Cliente/Servidor. Esto incluye la comunicacion de stacks, directorios distribuidos, autentificación de servicios, tiempo de red. RPC's, y servicios de encolado. Esta categoría incluye también extensiones de red del sistema operativo como servicios distribuidos de archivos o impresión. Algunos productos que caen dentro de esta categoría incluye NetWare, LAN Server, LAN Manager, TCP/IP, y Net BIOS

2. Middleware Servicio-Específico

Dado a las características que tienen los middlewares de servicio específico se han dividido en los siguientes grandes grupos .

- **Middleware Específico-Base de Datos**

Permite a los cliente invocar servicios basados en SQL Estos estándares definen una interfase a nivel llamada para SQL, ejemplos de éstos son tal como ODBC, IDAPI, DRDA, EDA/SQL, SAG/CLI.

- **Middleware Específico-OLTP (On Line Transaction Protocol)**

Permite a los clientes invocar servicios a través de múltiples servidores transaccionales TP Monitores (Monitores Transaccionales) permiten a los diferentes servidores controlar sus recursos locales y cooperar con otros monitores de transacciones cuando ellos necesitan acceder a recursos que no son locales. Los Monitores transaccionales garantizan la integridad de todas las actividades dentro y a través de los servidores El middleware Monitor de Transacciones consiste de las RPC's transaccionales que permiten a los clientes una transacción específica y las llamadas necesarias entre servicios para coordinar una transacción compleja. Tal como el ATMI y /WS (workstation) de Tuxedo, RPC transaccionales de Encina, y el TxRPC y XATMI de X/Open

- **Middleware Específico-Groupware**

Permite a los clientes invocar servicios en un servidor groupware Ésto es una nueva área donde los estándares middlware todavía se están desarrollando
Tal como MAPI, VIM, VIC y llamadas Lotus Notes

- **Middleware Específico-Objetos**

Permite a los cliente invocar métodos que residen un un servidor remoto. La llave del middlware es el OMG de CORBA Un ejemplo es el Servidor de Aplicaciones de Oracle, cuyo middleware es el Web Request Broker implementado con CORBA.

- **Middleware Específico-Administración de Sistema**

Permite administrar estaciones para hablar a servicios administrados
Tal como SNMP, CMIP y ORBs.

- **Middleware Servidor a Servidor**

El middleware no incluye el software que provee el servicio actual Esto lo hace incluyendo el software que es usado para coordinar las interacciones inter-servidor. Las interacciones Servidor-a-Servidor son generalmente de naturaleza Cliente/Servidor –los servidores son cliente de otros servidores-. Estas interacciones requieren servicios especializados de middleware Por ejemplo, el protocolo "*two-phase commit*", puede ser usado para coordinar la transacción que ejecutan múltiples servidores La mayoría de software moderno siguen el paradigma cliente servidor Y al menos un monitor de transacciones va tan lejos como proteger a la interfase de usuario usando la disciplina transaccional "*two-phase commit*".

2.2 Arquitectura Aplicada al Sistema Nuevo

La Arquitectura muestra los equipos de cómputo, los productos y la interrelación que tendrán cada uno de ellos en el sistema La arquitectura a desarrollar para el "Sistema Nuevo" se muestra en la siguiente figura.

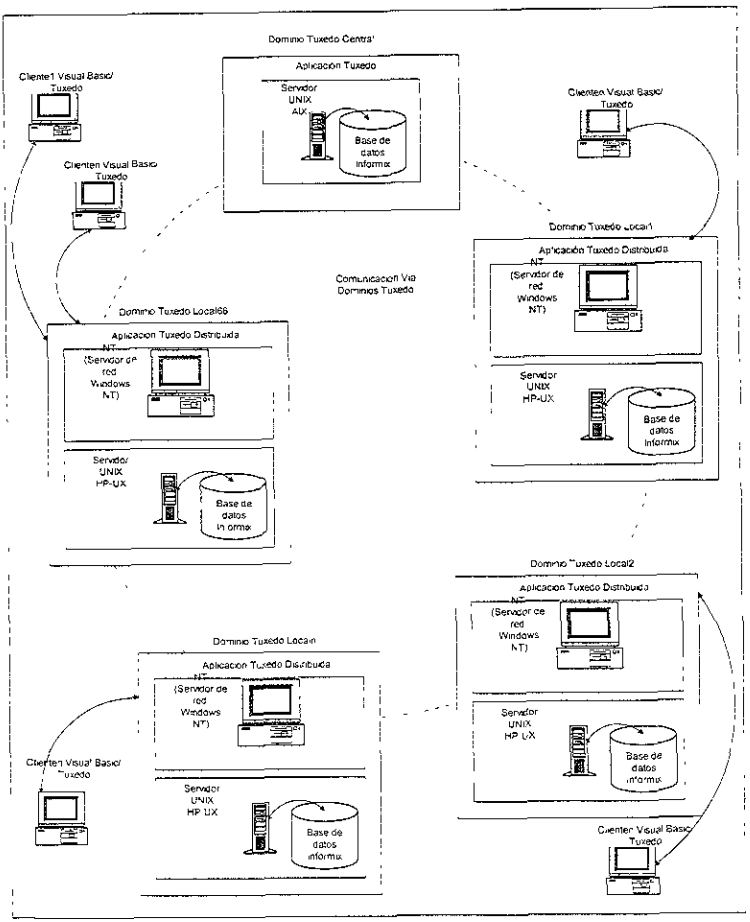


Figura 2.14 Arquitectura Global del Sistema Nuevo

En la figura 2.14 se muestran todos los elementos de la Arquitectura que están inmersos en el software de Tuxedo. Se ha puesto en el diagrama 5 nodos pero en la aplicación real existen 66 nodos distribuidos en toda la República Mexicana los cuales son independientes entre ellos Siguiendo el sentido de las manecillas del reloj, se muestra en primer lugar el Dominio Tuxedo Central, se muestra que es un nodo independiente el cual está diseñado como una aplicación Tuxedo no distribuida ya que toda la aplicación está en un equipo Unix a este nodo no se asocia ningún cliente ya que este nodo contiene principalmente una réplica de datos que se genera por cada operación hecha en los dominios locales. Los nodos Locales son, cada uno, un dominio local distribuido ya que la aplicación Tuxedo está configurada para expanderse en un Servidor NT y un servidor Unix HP. La administracion es controlada por el nodo maestro que se ha asignado al servidor NT. La definición de este tipo de aplicación Tuxedo distribuida fue un requerimiento por parte del cliente.

Todos los dominios Tuxedo pueden comunicarse unos con otros como lo muestra la línea punteada que los une, esto es posible ya que Tuxedo brinda esta característica indispensable para este sistema.

Fuera de la línea punteada se muestran los clientes construidos en Visual Basic y a los cuales se les ha instalado el producto WS Tuxedo (work station de Tuxedo) para poder hacer peticiones a los servicios Tuxedo. Las peticiones las hacen directamente al nodo local al cual están conectados y pueden hacer peticiones a servicios que no pertenecen a su dominio local vía dominios tuxedo. El que el servicio requerido esté o no en el dominio local es totalmente transparente para el cliente Visual Basic.

Esta Arquitectura muestra claramente un sistema distribuido de Cliente/Servidor 3 capas. La primera capa son los clientes Visual Basic, los cuales tendrán a su cargo ser la presentación al usuario, validar datos y llevar el diálogo entre el usuario y el sistema La segunda capa es Tuxedo en la cual se administra el sistema, y se construyen las reglas del negocio, estas dos funciones son las principales en el sistema y se detallarán en el capítulo siguiente La última capa es la base de datos la cual almacena todos los datos generados y requeridos por el sistema, en este caso es Informix

En esta gráfica se aprecia la utilización de la tecnología de la cual se habló a lo largo de este capítulo para resolver la problemática mostrada en el capítulo 1 A continuación se detallará los productos de la arquitectura utilizada en el Sistema Nuevo en cada una de sus fases de desarrollo.

2.2.1 Catálogo de Tecnología

Describe los estándares, productos y lineamientos para formar la infraestructura sobre la cual va a ejecutar la aplicación, tales como ambiente de desarrollo, servicios de red, sistema operativo, equipos, bases de datos, etc.

La figura 2.15 muestra a nivel global qué productos y sobre qué plataforma se integran.

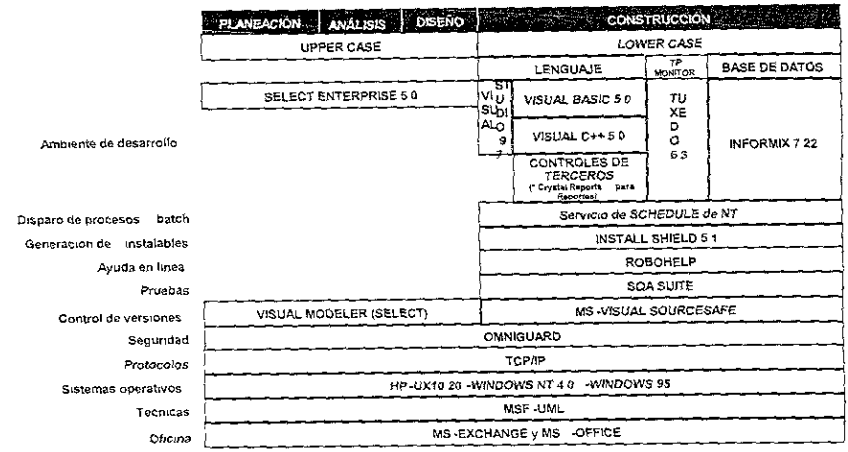


Figura 2.15 Catalogo de Tecnología

El uso de los elementos de la Figura 2.15, se explica a continuación:

Ambiente de Desarrollo

Herramienta CASE

Para las etapas de Planeación, Análisis y Diseño utiliza Select Enterprise 5.0

Aunque la herramienta estándar de Empresa es Paradigm Plus, mientras no exista un mecanismo para migrar transparentemente los datos de Select a Paradigm, los módulos se diseñan con Select

Lenguaje

Se utilizará Microsoft Visual Studio '97, para aprovechar la diversidad de funcionalidades que ofrece a través de los distintos lenguajes que lo integran, utilizando principalmente.

- Visual Basic 5 0 para la interfase gráfica, auxiliado por herramientas de terceros para facilitar el desarrollo, como Crystal Reports para la generación de reportes
- ANSI C para el desarrollo de los servicios y clientes Tuxedo (ver TP Monitor), utilizando el compilador de Visual C++

TP Monitor o middleware

Tuxedo está fungiendo como middleware –para trabajar con las bases de datos- .como Monitor de Transacciones, -para asegurar las transacciones- , así como la base de la aplicación, ya que la lógica del negocio se programará en servicios de Tuxedo, en C

Todos los accesos y actualizaciones a los datos, se deben realizar con servicios de Tuxedo, incluyendo las réplicas de datos entre servidores

El cliente (front-end) no accesa directamente a la base de datos, sino que lo hace a través de llamadas (directas o indirectas) a servicios Tuxedo

Se deben programar, además, servicios Tuxedo que funcionen como interfase entre Base de Datos Actual y , para asegurar que las transacciones afecten ambos sistemas y lograr los objetivos de la Convivencia Es requerimiento que las bases de datos del SIR y el Sistema Nuevo convivan

Manejador de Base de Datos

Como manejador relacional de bases de datos, el Sistema Nuevo utilizara Informix OnLine Dynamic Server, versión 7.22 (o superior), y convivira con la versión 5 02 que reside en la T-500 mientras no se actualice o mientras no se terminen de sustituir a los módulos del Sistema Actual Sólo se utilizara como manejador, pero no se utilizará el lenguaje 4GL que contenga

Disparo de procesos Batch

Dada la necesidad de realizar procesos a determinadas horas de determinados días, se programará el disparo de programas de manera Batch, con el servicio de Schedule de Windows NT. (No confundir con el Paquete de Schedule de Windows 95)

Generación de instalables

Herramienta para crear el "setup" de la aplicación, de manera que sea portable y facilite la instalación en cada

ALR Install Shield 5 1

Ayuda en Línea

RoboHelp, para desarrollar la ayuda que el usuario tendrá en pantalla

Pruebas

SQA Suite

Control de versiones

- Durante la etapa de Planeación, Análisis y Diseño, se utilizará Visual Modeler de Select
- Durante la etapa de Construcción, se utilizará Visual SourceSafe, de Microsoft

Seguridad

Se realizará a través de Omniguard

Protocolos

TCP/IP Las comunicaciones entre las oficinas locales y regionales y el centro de procesamiento nacional utilizan uno de dos protocolos, TCP/IP o X 25 La mayor parte de los enlaces son del primer tipo, y los restantes están siendo actualizados. Para poder utilizar adecuadamente el Sistema Nuevo en las oficinas Locales, es necesario que cuenten con el protocolo TCP/IP, de modo que pueda trabajar con la oficina Central

Sistemas Operativos

- Para los servidores de bases de datos, se utilizará HP-UX 10.20
- Para los servidores de Tuxedo, se utilizará Windows NT 4 0 y HP-UX
- Para los clientes de la aplicacion, se utilizará Windows 95

.2.2 Elementos de Software en el Sistema Nuevo

.a figura 2.16 muestra el conjunto de elementos que conforman el sistema Es decir, a nivel funcional qué se necesita en cada equipo para interactuar

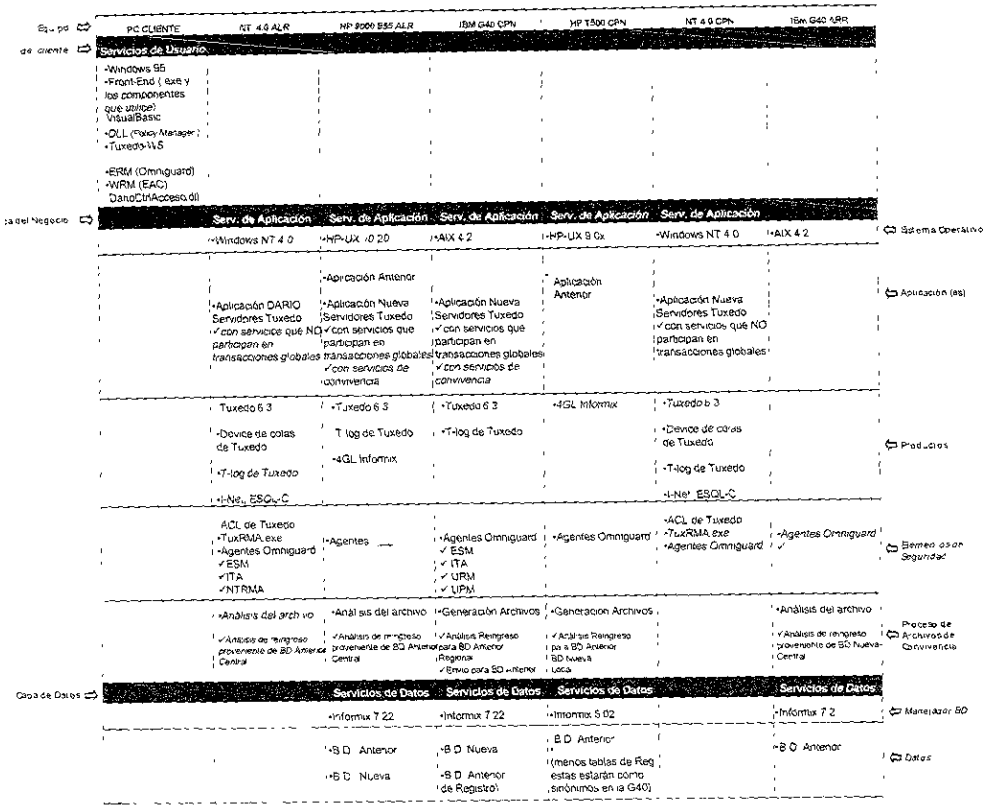


Figura 2 16 Elementos de software en cada Capa

3. TUXEDO - SEGUNDA CAPA

3.1 Definición de TUXEDO

"El Sistema TUXEDO es un conjunto de módulos de software que permite la construcción, ejecución y administración de alto rendimiento, de aplicaciones distribuidas" [1]. Originalmente TUXEDO fue planeado para crear este tipo de sistemas bajo el sistema operativo UNIX, pero el alcance creció y el desarrollo del propio producto hizo posible la construcción de sistemas distribuidos en un amplio rango de sistemas operativos, así las aplicaciones TUXEDO puede incluir diversas plataformas de cómputo, partiendo desde diversas estaciones de trabajo hasta múltiples servidores

3.2 Arquitectura Base

En el capítulo anterior se habló de la arquitectura Cliente/Servidor 3 capas, donde la segunda capa, el "middleware" podía ser del tipo de específico –OLTP On Line Transaction Protocol, TUXEDO entra en esta clasificación

TUXEDO es un producto de "middleware", el cual distribuye las aplicaciones en múltiples plataformas de cómputo, bases de datos y sistemas operativos, utilizando comunicaciones basadas en mensajes, y si se desea, proceso transaccional distribuido

Tuxedo también es clasificado como un monitor de transacciones ya que maneja un proceso transaccional distribuido Como monitor se especializa en el manejo de la transacción desde el inicio, que generalmente empieza en el proceso cliente y se lleva a cabo a través de uno o más servidores hasta volver al proceso cliente Cuando la transacción termina, Tuxedo se asegura que todos los sistemas involucrados en la transacción quedaron en un estado consistente, esto es, que se afectaron todas las bases de datos involucradas o que ninguna fue afectada

[1] The Tuxedo System, Juan M Andrade, Mark Carges, Stephen Felts, pág 24

La propia evolución del sistema TUXEDO lo ha ubicado como una extensión del propio sistema operativo Cuando existen aplicaciones muy grandes, es decir, que atienden a cientos o miles de clientes y existe concurrencia, el sistema se "alenta", esto es, el sistema tarda en responder a los usuarios debido a que los recursos que necesita el sistema siempre están ocupados Tuxedo, como una extensión del propio sistema operativo, puede conectar en tiempo real a miles de clientes a un grupo de procesos servidores compartidos optimizando así los recursos disponibles conexiones, memoria, procesos En la figura 3.1 se esquematiza esta optimización. Un ejemplo de este tipo de sistema que fácilmente podría degradarse sin un monitor de transacciones, es el de los cajeros automaticos de un banco, donde pueden ejecutarse miles de operaciones en un día crítico, en la quincena por ejemplo

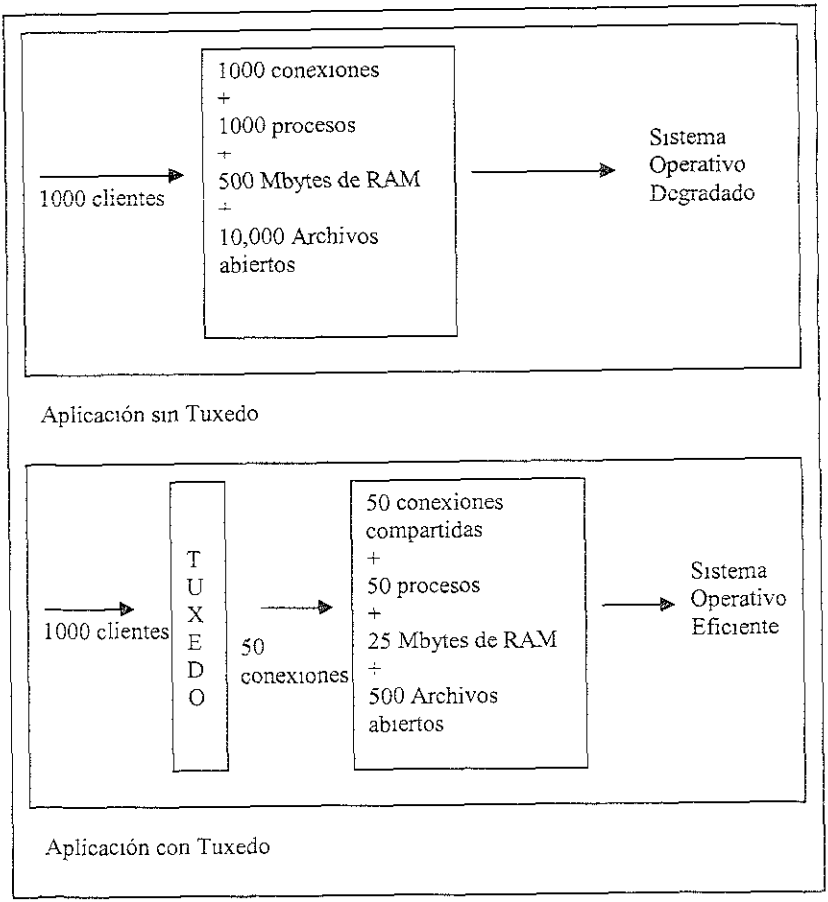


Figura 3 1 Sistema Operativo con y sin Tuxedo

El sistema TUXEDO brinda un estándar a la industria para la creación y administración de sistemas transaccionales en línea y distribuidos, en un medio ambiente heterogéneo Cliente/Servidor facilitando el desarrollo de aplicaciones, así como una infraestructura para la administración y mantenimiento de sistemas distribuidos confiables y de alto rendimiento

3.2.1 Historia y Evolución de Tuxedo

La construcción del sistema TUXEDO empezó en 1983 con el objetivo de habilitar al sistema operativo UNIX para la realización de transacciones. El proyecto inició con el nombre de UNITS (Unix Transaction System) dentro de la división Laboratorios Bell de AT& T

En los siguientes años, UNITS evolucionó hacia un sistema que brindara servicios hacia un alto número de usuarios, así como el soporte necesario para la creación de aplicaciones Cliente/Servidor, donde se integró un producto con nombre código TUX (Transacciones en UNIX), el cual posteriormente se extendió para realizar operaciones distribuidas, con la finalidad de introducir el producto en el mercado se le bautizó con el nombre de TUXEDO (TUX Extended for Distribute Operation), Transacciones en UNIX Mejorado para Operación Distribuida

La primera version de TUXEDO, se encontró disponible en 1984, este sistema permitía realizar aplicaciones Cliente/Servidor en una misma máquina, así, las aplicaciones podían ser divididas funcionalmente en procesos cliente y procesos servidor, éstas ocupaban una interfase rudimentaria denominada *API*, Aplication Programming Interface, Interfase de Programacion de Aplicaciones, la cual soportaba una interacción de forma requerimiento/respuesta, de esta forma el cliente localizaba al servidor, enviándole su requerimiento vía un servidor de nombres de alta velocidad llamado "*Bolletin Board*" [2]

En 1986 se introdujo al mercado la segunda versión de TUXEDO, la cual brindaba las primeras características de alta disponibilidad, esto quiere decir a prueba de fallas, para las aplicaciones, esta característica contemplaba el monitoreo y reinicio de las aplicaciones con falla, así como tambien las funciones de monitoreo entre sistemas y de auto diagnóstico

[2] Las palabras en letra cursiva, se definen más adelante en este mismo capítulo Los términos en idioma inglés no se han sustituido ya que la configuracion y programación requieren el término de esta modo, pero aunque no se sustituyen se definen enseguida en español

La tercera versión se encontró disponible en 1987 y brindaba las características de Cliente/Servidor en una computadora constituida de diversos elementos de proceso interconectados entre sí por medio de un canal de alta velocidad, es importante mencionar que cada elemento, contaba con su propio CPU y memoria privada Dentro de las características que se incluyeron en esta versión, se tenía la replicaciones del "Bulletin Board" en la memoria de cada elemento del proceso, permitiendo la operación de clientes y servidores en diversos equipos utilizando la misma interfase (*API*). Además, se incluyó un archivo central de configuración y herramientas de administración.

En la versión 4 de TUXEDO, se incluyó la funcionalidad de la version anterior extendiendo las plataformas de cómputo, creando de esta manera sistemas distribuidos en diversos sistemas operativos heterogéneos Dentro de las mejoras en la tecnología presentada por esta versión, se incluyó la distribución del "Bulletion Board", la introduccion del soporte transaccional y la interfase para monitoreo de aplicaciones transaccionales denominada *ATMI* (Application to Transaction Monitor Interface), Interfase para Aplicaciones de Monitor de Transacción. La introducción de transacciones llevó al establecimiento de la interfase *XA* para soportar el control de transacciones entre base de datos heterogéneas Además se introdujo el requerimiento de utilizar una librería común de interfase para la transmisión de diferentes tipos de datos entre computadoras heterogéneas, "*Typed buffers*"

La siguiente versión en liberarse fue la versión 4 2 en 1993, en la cual se incluyeron diversos tipo de clientes como las estaciones de trabajo y las computadoras personales, esta versión además incluyó la conexión a mainframes, ofreciendo así la posibilidad de crear sistemas distribuidos desde las estaciones de trabajo hasta la computadora central mainframe, soportando comunicaciones con base en colas y conversaciones, utilizando la misma interfase ATMI

La versión 5 de TUXEDO, introdujo el soporte de dominios "Domains", esta característica permite la interacción de aplicaciones entre diversos sistemas TUXEDO. Además, esta versión presentó la interfase de invocación de procedimientos remoto en forma transaccional "TxRPC" (Transactional Remote Procedure Call), como la sintaxis para la comunicación requerimiento/respuesta. Como resultado de esta versión, se lograron establecer configuraciones de gran escala, así como el proceso de transacciones entre compañías (Inter-enterprise transaction procesing)

En la versión 6 de TUXEDO, se implantó la arquitectura de administración para las aplicaciones, incluyendo el TMIB (Tuxedo Management Information Base), Información Administrativa Interna de Tuxedo, que accede a los datos administrativos del Sistema TUXEDO, con una interfase denominada TMIB API, y una interfase de presentación gráfica . Además dentro de ésta versión, se incluyeron controles de seguridad *ACL*

(Access Control List), Lista de Control de Acceso y se incluyó un disparador de eventos con la misma interfase ATMI.

Actualmente la versión de TUXEDO incluye, además de todas las facilidades anteriores, con una interfase al comercio electrónico, haciendo factible la realización de sistemas de función distribuida utilizando la internet

3.2.2 Elementos de Evolución

A continuación, se listan las características de TUXEDO que dieron origen a su evolución

- ATMI Es una interfase para la comunicación de sistemas distribuidos, Interfase para Aplicaciones de Monitor de Transacción , un subconjunto de ésta la adoptó *X/Open*, bajo el nombre de XATMI.

- Typed Buffers Es una tecnología para identificar colecciones de datos a ser transformados cuando existe la comunicación entre diversos equipos con diferente representación de datos

- FML Consiste en una estructura de datos y métodos de acceso asociados, con el propósito de comunicar un número variable de parámetros de tipo definido entre aplicaciones cooperativas

- XA. Es una interfase de comunicación entre el administrador transaccional (TM) y los componentes de software denominados administrador de recursos (RM) Esta interfase se planteó con base en el modelo de X/Open.

- /WS. Consiste en la implementación del cliente de la interfase ATMI, y puede ser utilizada en computadoras de escritorio con sistemas operativos Windows, Mac Os, OS/2, DOS y Unix.

- /Q Consiste de un sistema de colas entre aplicaciones en forma transaccional, el cual opera como un "resource manager" de la aplicación, pudiéndose establecer en la misma computadora o entre diferentes computadoras Bajo esta forma operativa, la comunicación entre los módulos de software involucrados es independiente del tiempo, cabe señalar que el sistema cumple con el estándar XA e incorpora typed buffers en su funcionalidad.

- /HOST y SNA Domains Este subsistema provee la interfase ATMI con elementos para la interacción con sistemas IBM CICS.

- /OSI-TP. Este subsistema provee de la interfase ATMI para la interacción con sistemas basados en el protocolo OSI-TP estándar de ISO.

- Event Broker Consiste de un sistema de comunicaciones de publicación y suscripción con semántica transaccional Se puede establecer el envio de mensajes cuando un evento determinado ocurre.

- TMIB API Interfase de programación hacia la arquitectura de administración TUXEDO Mediante esta interfase podemos acceder a información interna de la aplicación TUXEDO y hacer más eficiente el trabajo de optimización de la aplicación.

3.2.3 Propietarios de TUXEDO

En cuanto a las compañías que han sido dueñas de TUXEDO posteriormente a su desarrollo por los laboratorios Bell son: UNIX System laboratories (USL), posteriormente Novell y actualmente BEA Systems por lo que en los nuevos productos se antepone **BEA**, por ejemplo **BEA** TUXEDO En este trabajo se omitirá el prefijo

3.2.4 TUXEDO en el Mercado Mexicano

El desarrollo de Aplicaciones TUXEDO empezó hace varios años y en México empezó aproximadamente en 1995 Principalmente comenzó con grandes empresas y con aplicaciones muy aisladas. En este momento, por la necesidad de participar en sistemas de aplicaciones de 3-capas de misión critica, ha aumentado el número de desarrollos con TUXEDO

Las siguientes empresas cuentan en este momento con aplicaciones TUXEDO y además muchos de ellos siguen licitando nuevos proyectos son

- Banamex Sistema de Fraudes. -Sistema Interfase Juridico-Hipotecario, Sistema de Tarjetas y Canal Inteligente
- Telmex Inventario de la planta telefonica
- Serfin Proyecto de Divisas
- Atlántico
- IFE

- Gigante
- Bancomer
- InverMéxico. Proyecto de Mercado de Dinero

Los Sistemas Tuxedo siguen desarrollándose actualmente en nuestro país dado que las necesidades de conectividad entre sistemas de diferentes plataformas exigen la utilización de herramientas de alto rendimiento capaces de comunicarse confiablemente entre todos los elementos participantes en estos sistema

3.3 Características de TUXEDO

Dentro de las principales características y beneficios del sistema TUXEDO se encuentran incluidas las siguientes

- Modularidad
- Rendimiento
- Independencia de datos
- Transparencia de localizacion
- Independencia de funciones
- Manejo de transacciones
- Escalabilidad
- Alto rendimiento
- Confiabilidad en misión crítica
- Soporta estandares
- Ruteo dependiente a datos
- Balanceo de cargas
- Manejo de prioridades
- Replicación de servidores
- Manejo del modelo Cliente/Servidor mejorado
- Tolerancia a fallas
- Manejo de diversas bases de datos
- Soporte en diferentes lenguajes
- Seguridad

Modularidad

La organización de la aplicación está distribuida en unidades bien definidas; el cliente y el servidor son independientes.

Rendimiento

Se conservan los recursos del sistema, dado que TUXEDO no abusa de los recursos Se reduce el tamaño de los clientes y de las interacciones.

Independencia de los datos

Los métodos de acceso a datos y base de datos son transparentes para el cliente

Transparencia de localización

La comunicacion es independiente de la localización, los servidores pueden estar distribuidos en diferentes máquinas y las peticiones hechas por el cliente, serán transparentes a la localización del servidor que requiere, ya que los llamados a los servicios lo hace por medio de un nombre, el cual es el mismo para cualquier cliente. Internamente TUXEDO se encarga de "rutear", esto es enviar, la petición del cliente al servicio correspondiente.

Independencia de funciones

El servidor se ubica entre el cliente y DBMS Siendo el servidor el responsable de la comunicación entre ellos El cliente interactúa con el usuario final y hace la petición a servicios por su nombre

Escalabilidad

La distribución de clientes y servidores en múltiples plataformas heterogéneas provee un escalabilidad lineal, la cual es necesaria para soportar la operación de muchos usuarios, ademas permite agregar funcionalidad en unidades pequeñas

Alto rendimiento

El balanceo de carga de trabajo y la afinacion de rendimiento, permiten obtener un rendimiento transaccional alto con tiempos de respuesta cortos, además el ruteo de transacciones con base a los datos y el acceso concurrente a las bases de datos, brinda el soporte de largos volúmenes de datos

Confiabilidad de misión crítica

TUXEDO brinda un alto nivel de integridad de datos, un alto nivel de tolerancia a las fallas y una disponibilidad 7 x 24 (7 días a la semana las 24 horas), además de poder manejar llamadas concurrentes

Soporta estándares

TUXEDO se adhiere a los estandares de X/Open e ISO

Ruteo dependiente de los datos

Esta caracteristica permite a un cliente hacer la llamada a un servicio (mismo nombre) utilizando diferentes rutinas, dependiendo del contenido de los parámetros enviados por el cliente Uno de los usos más comunes para utilizar ruteo es cuando se requieren particionar horizontalmente la información de una base de datos Es decir que los datos estén particionados en diferentes equipos Los servicios son procesados de manera idéntica (misma lógica) pero en diferentes equipos Aunque también es factible que los servicios realicen diferentes operaciones dependiendo de los datos de entrada.

Balanceo de cargas

Los servicios ofrecidos en una aplicación TUXEDO pueden estar contenidos en uno o varios servidores, estos servicios pueden tardar desde milisegundos hasta segundos, es decir, si un cliente realiza una solicitud hacia un servicio que se tarda segundos, y otro lanza una petición a uno que se tarda milisegundos y los dos están contenidos en el mismos servidor, uno estará bloqueando la petición del otro, el balanceo permite que cuando el segundo cliente lance la petición, TUXEDO la asigne al servidor que tenga menos trabajo pendiente por realizar.

Manejo de prioridades por cliente y por servicio

TUXEDO permite poner prioridades tanto a los servicios como a los clientes en escala de 1 a 100, siendo la de mayor prioridad la 100, en este caso si dos peticiones llegan concurrentemente se resolverá la que tenga una mayor prioridad. Esta prioridad puede ser definida por servicio o para un mismo servicio por cliente (la misma transacción pero dependiendo de quien la envie tendra una prioridad mayor)

Replicación de servidores

Esta caracteristica consiste en poder crear una imagen de un servidor que tiene mucha demanda, es decir replicar el proceso para poder ofrecer el servicio desde diferentes puntos de entrada y así poder atender a más usuarios como resultado obtener mejores tipos de respuesta

Manejo del modelo Cliente/Servidor mejorado

Los servidores TUXEDO se desarrollan con la filosofía de poder atender a más de un usuario, cuando dos transacciones son enviadas concurrentemente, el servidor atiende a una y la otra la pone en su cola de espera, de esta manera un servidor puede atender hasta 'n' clientes con un solo servidor, el número de servidores depende de cada aplicacion Como beneficio de este modelo tenemos que si se esta accediendo a alguna base de datos, no es necesario crear una conexion por cada cliente que lance una petición hacia la base sino que el servidor TUXEDO es el que se conecta a la base de datos, obteniendo una minimización en el número de conexiones hacia ésta, lo cual trae por resultado beneficios tanto en costo como en rendimiento Por ejemplo, si se tienen 100 clientes concurrentes implicaria 100 conexiones hacia la base de datos con TUXEDO se

podría tener 1 servidor o sea 1 conexión hacia la base de datos para atender a los 100 clientes (esto puede depender de la aplicación)

Tolerancia a fallas

Dentro de la arquitectura de TUXEDO, se puede implementar diferentes mecanismos para desarrollar una aplicación tolerante a fallas, entre los principales se encuentran:

- Dentro de una aplicación distribuida se puede definir un nodo de respaldo en caso de que el nodo maestro se quede fuera de línea, el nodo de respaldo toma el control y éste se convierte en el nodo maestro.
- Reconexiones automáticas En el caso de que un cliente pierda la conexión durante el envío de una transacción, TUXEDO trata de realizar la reconexión, en el caso que se haya perdido totalmente la comunicación, el problema deberá tratarse de otra forma (almacenamiento físico, "*2-Phase Commit (2-PC)*) Igualmente esta reconexión la realiza si se pierde la comunicacion entre un servidor y la base de datos
- Restablecimiento automático de servidores Si un proceso servidor de aplicación llega a fallar y se perdiera la conexión del servidor por cualquier razón, TUXEDO intenta restablecerlo de manera automática
- Establecer una línea de espera de transacciones, comúnmente definido como "encolar la transacción", En el caso de que una transacción no pueda realizarse en línea, se puede poner en una cola, esto es almacenar físicamente en disco las peticiones para proceder a procesarlas posteriormente

Manejo de diversas bases de datos

TUXEDO puede acceder varias bases de datos Informix, Oracle y Sybase

Soporte en diferentes lenguajes

Tuxedo soporta diferentes lenguajes para su construccion como lo son C, C++, Cobol Visual Basic, etc

Seguridad

TUXEDO maneja diferentes niveles de seguridad, el más utilizado es *ACL's* La seguridad se describirá en un punto más adelante.

3.3.1 Elementos del Sistema TUXEDO

De forma general, los elementos que componen al sistema TUXEDO son los siguientes:

Bulletin Board (BB)

Este elemento constituye la base del nodo TUXEDO, y contiene una lista de todos los servicios, incluyendo los mapas de los servidores que los brindan formando de esta manera un mapa de navegación. Existe una copia del Bulletin Board por cada nodo, o máquina incluida en la aplicación, así un requerimiento necesita únicamente incluir el nombre del servicio y sus parámetros asociados, cabe señalar, que no requiere incluir al servidor El Bulletin Board, provee la dirección del servidor que ofrece el servicio ruteando el requerimiento a éste y en el caso de encontrar el servicio en otra máquina, es enviado a través de un canal de comunicación

Bulletin Board Liaison (BBL)

Este elemento es un proceso administrativo, el cual monitorea continuamente el estado de los componentes del nodo TUXEDO, así como también, maneja la coordinación de funciones para todos los componentes del sistema.

Distinguished Bulletin Board Liaison (DBBL)

Este elemento es responsable de propagar los cambios de la información global de administración del sistema TMIB Además coordina el estado de las diversas máquinas involucradas en una aplicación, existe sólo un DBBL para toda la aplicacion, y ésta puede migrar a otra máquina en caso de falla.

Bridge

La comunicación entre diversas máquinas servidores conectadas en red, es optimizada por el establecimiento de canales de comunicación multiplexados entre éstas, el elemento que lleva a cabo estas funciones, así como la de administrar el tráfico es el Bridge. Este elemento en principio, envía y recibe requerimientos de servicios, utilizando los transportes estándar de comunicacion y "ruteándolos" a las colas locales de servicios. Además, este elemento realiza la función de codificar y decodificar los datos entre máquinas.

Message Queues

Este componente está apoyado y forma parte del sistema operativo, tiene la función de realizar la comunicación entre clientes y servidores del sistema.

UBBCONFIG

Este elemento es físicamente un archivo ASCII, el cual contiene la configuración de la aplicación y de los diversos nodos que la conforman, después de compilarse el archivo UBBCONFIG se genera un archivo binario denominado TUXCONFIG, utilizado para la construcción del BBL

Transaction Manager Server (TMS)

El sistema TUXEDO contiene un mecanismo optimizado para la administración de transacciones, éste explota el proceso *asíncrono* para el manejo de la terminación de la transacción en servidores administrativos en lugar de servidores aplicativos. Este mecanismo permite que los servidores aplicativos libremente procesen requerimientos de servicios con nombres de diferentes transacciones, sin tener que esperar por la terminación de cualquier transacción Así la conclusión de las transacciones es controlada por el administrador de transacciones TMS

Workstation Listener (WSL)

Este componente acepta los requerimientos de conexión realizados por las estaciones de trabajo TUXEDO (/WS), además este componente administra el conjunto de procesos manejadores de estaciones, activandolos o desactivándolos dependiendo de la demanda Esto permite que los procesos clientes puedan residir en equipos diferentes al equipo que contiene la aplicación Tuxedo

Workstation Handler (WSH)

El workstation Handler o manejador de estaciones de trabajo, realiza la función de manejo de una o más conexiones desde y hacia las estaciones de trabajo TUXEDO (/WS), realiza la conexión con los servicios aplicativos, administrando las transacciones realizadas y regresando las respuestas. Como se mencionó estos procesos son activados y desactivados por el WSL

Cliente

Los procesos cliente, son programas ejecutables, donde el flujo de control es escrito por el programador de la aplicación y es generalmente manejado por la interaccion del usuario, comunicaciones, o alguna otra forma de entrada a TUXEDO, estos programas, realizan llamadas a servicios vía TUXEDO, dependiendo de la lógica del negocio El código de la aplicación interactúa por medio de la interfase *ATMI* y tiene entre sus características principales.

- Ejecuta interacción con el usuario
- Envía requerimientos de servicio por nombre usando *ATMI*.

Servidor

Estos procesos proveen de uno o más servicios el control del flujo de la lógica está definido por TUXEDO, así, un servidor espera un mensaje en su cola asignada, de forma que cuando un mensaje llega a la cola, este es asignado a un servicio, que lo procesa y regresa el resultado a TUXEDO para ser enviado al cliente que hizo la petición Cada servidor publica uno o más servicios, los cuales pueden ser invocados por los clientes, además, los procesos servidores se arreglan en grupos administrativos, de forma que cada grupo tenga acceso a un administrador de recursos y sus caracteristicas principales son

- Es un proceso activo que acepta requerimientos y los canaliza a las rutinas de servicios correspondientes.
- Pueden estar disponibles muchos servicios.
- Un mismo servicio puede estar disponible en varios servidores
- Varios servicios pueden ser manejados por una sola rutina
- Un servidor puede convertirse en cliente

La figura 3.1 ilustra gráficamente los elementos principales del "middleware" TUXEDO los cuales se describieron en esta sección.

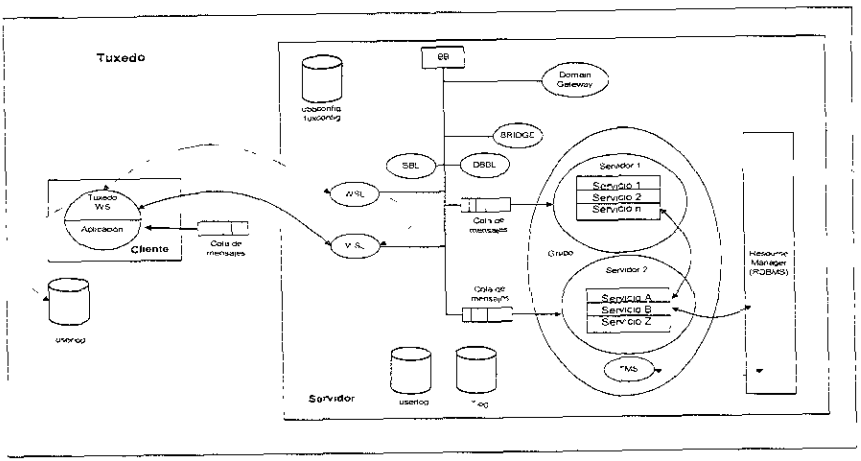


Figura 3.1 Elementos Principales de Tuxedo

3.3.2 Tipos de Requerimiento y Respuesta TUXEDO

Requerimiento / Respuesta

Este tipo de interacción es realizado por medio del modelo de colas empleado por TUXEDO, como se mencionó previamente, cada proceso servidor posee una cola de comunicación de mensajes denominada cola de requerimiento y cada cliente posee una cola llamada cola de respuesta La operación es del tipo sin conexión, esto es, en lugar de establecer una conexión por cada cliente, el servidor requiere solamente "mirar" el contenido de la cola de requerimientos y de igual forma el proceso cliente "mirará" la cola de respuesta Los requerimientos pueden ser de los siguientes tipos:

- **Requerimientos síncronos**
  En este tipo de requerimiento el proceso cliente se detendrá a esperar por la respuesta del servicio.

- **Requerimiento asíncrono**
  Este tipo de requerimiento a diferencia del anterior, el proceso cliente no se detiene a esperar la respuesta, sino que puede continuar su ejecución, verificando la cola de respuesta posteriormente

- **Requerimientos transaccionales**
  Estos requerimientos tienen su relación en el manejo del estado de las transacciones realizadas, así por ejemplo un cliente podrá iniciar una coordinación de una transacción que involucra diversos servicios en la misma o en otra computadora

- **Requerimientos de forwarding**
  Este esquema de interacción consiste en el paso del requerimiento recibido a otro servicio el cual lo realizará, así, los procesos servidores no son bloqueados, sin ser necesario esperar por la respuesta ya que el segundo servicio regresará directamente al cliente

- **Conversacional**
  Esta modalidad a diferencia de las anteriores, establece una conexión virtual entre un cliente y un servidor, así, se pueden intercambiar datos entre éstos, hasta que se solicite la terminación de la conversación. Este tipo de interacción bloquea un servidor a un cliente por la duración de la conversación

3.4 Modelo X/Open DTP

El modelo X/Open DTP (Distributed Transaction Processing), se define como el modelo arquitectónico estándar abierto para el desarrollo de sistemas de procesamiento transaccional distribuido, éste plantea los componentes integrantes, su funcionalidad, y las diversas interfaces necesarias para su operación, en un contexto general para sistemas de esta naturaleza. Cabe mencionar, que para la definición del modelo, se consideró la intervención de la gran mayoría de empresas proveedoras de tecnologías, por ejemplo IBM, BEA, HP, etc La mayoría de los monitores de transacciones siguen las especificaciones establecidas por X/Open y de ISO (International Standard Organization), Organización Internacional de Estándares para poder integrar diversas plataformas a un sistema distribuido A continuación se describirán los principales elementos involucrados en este modelo

Transacción

El modelo X/Open DTP, define a una transacción, como una unidad lógica de trabajo, la cual agrupa un conjunto de cambios en el sistema, los cuales, deberan completarse "todos" exitosamente y convertirse de esta forma en permanentes. Ahora bien, si existiera alguna falla, "todos" fallarán de forma que el sistema se restaure a su estado previo La unidad lógica de trabajo y la transacción de negocio, la cual representa, es la unidad fundamental, u operación, la cual toma al sistema de un estado consiste y persiste a otro estado Una transacción consiste en una unidad de trabajo que presenta las siguientes propiedades, también conocidas como las propiedades de ACID

- Atomicidad (Atomicity)
- Consistencia (Consistency)
- Aislamiento (Isolation)
- Durabilidad (Durability)

La transacción deberá ser capaz de realizar los cambios, ***"commit"***, o retroceder operación, ***"rolled back"***, de una transacción El commit significa que los cambios a recursos compartidos, como puede ser el caso de una base de datos, tienen un efecto permanente

Transacción global distribuida

Una transacción global o transacción distribuida, se define como la transacción que podrá ocurrir entre múltiples recursos compartidos (Resource Managers). La decisión de efectuar "commit" o retroceder "rollback" debe considerar el estado del trabajo ya realizado en cualquier lugar por la transacción, esta operación es completada utilizando un protocolo de "commit" de dos fases o "2PC" (two phase commit). Este protocolo de describe en el siguiente punto de este capítulo.

El modelo en su componente denominado aplicación, se integra principalmente de los programas servidores y servicios, los cuales utilizan la funcionalidad brindada por los demás componentes y su principal función consiste en realizar la lógica aplicativa o de negocio. Por ejemplo, la transacción de "Cambio de Domicilio del Cliente", donde esta transacción puede dividirse en los siguientes servicios verificar datos nuevos, asignar nueva dirección, borrar de la sucursal anterior, todos estos servicios deben ejecutarse correctamente para que la transacción sea exitosa.

El componente CSI (Client Server Interfase)

Tiene como principal función, el realizar y administrar la relación de la aplicación con la arquitectura cliente servidor, por medio del establecimiento de un protocolo que implementa la funcionalidad por encima de los protocolos de comunicaciones. La interfase brindada por este componente se denomina TP, por lo que, las llamadas del tipo tp como tpcall(), pertenecen a esta interfase

El componente TM (Transaction Manager)

Este es el responsable de controlar el proceso de las transacciones, realizando su coordinación por medio del protocolo de "two-phase commit", y realizando, en el caso, su recuperación Este componente interactua con el componente aplicativo mediante la interfase denominada TX, como es el caso del txcommit(), pertenecen a esta interfase.

El componente denominado RM

Administrador de recursos o "Resource Manager", como lo indica su nombre, realiza la administracion de elementos como las bases de datos sistemas de colas, etc , interactuando con el componente aplicativo por medio de su interfase nativa, como un ejemplo se puede citar el lenguaje SQL

Interfase XA

En una transaccion global, el "Resource Manager", por ejemplo Informix, es controlado por medio de la interfase XA, la cual implementa el protocolo de "Two-phase commit". XA brinda una interfase de programación que permiten el control y la coordinación de una transacción global realizada por el Monitor de la Transacción, TUXEDO La utilización de esta interfase, permite la comunicación de TUXEDO con diversos "Resource Managers", en este caso la base de datos, con la finalidad de realizar la coordinación de una transacción global

En la siguiente figura se muestra la relación que guardan los principales elementos del Modelo X/Open descritos en este tema.

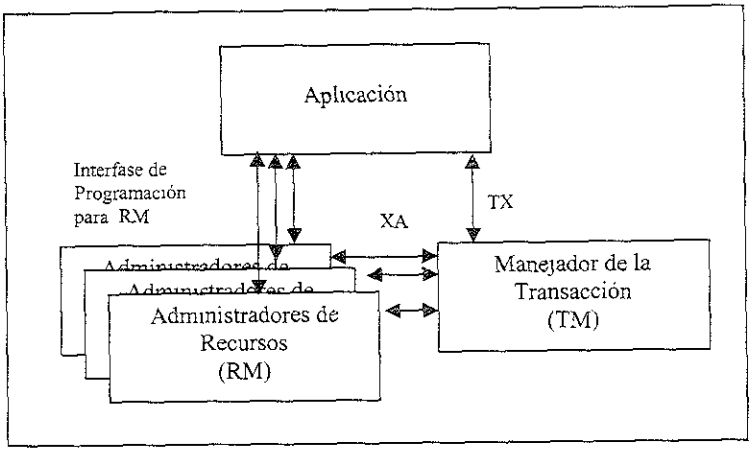


Figura 3.2 Relación de los principales elementos que involucra el modelo X/Open

A continuación se muestra una pequeña muestra de código de un servicio transacción construido en Tuxedo afectando únicamente una base de datos con la finalidad de ejemplificar una pequeña transacción. Para inicializar la transacción, en este caso, se utiliza la directiva de tx_begin y para finalizarla la sentencia de tx_commit. si hubiese un error se restablecerá la base de datos con un tx_rollback. En cada sentencia se despliega el error que pueda generase

```
if(tx_begin() != TX_OK)
 {
   userlog ("Error tpbegin al conectarse a DB XA"),
   userlog ("tpstrerror[%s] tperrno %d",tpstrerror(tperrno), tperrno);
   userlog(" Mensaje de error >>> [%s] ", sqlca.sqlerrm sqlerrmc );
 }
 EXEC SQL INSERT INTO TPrueba values ('PruebaFINAL_01', 'Tran'. 1);
 if(sqlca.sqlcode != 0)
 {
   userlog ("C604_PruebaTran: error al insertar TPrueba"),
   userlog(" sqlca sqlcode %d". sqlca.sqlcode );
   userlog(" Mensaje de error >>> %s ", sqlca sqlerrm.sqlerrmc ),
   tx_rollback().
   tpreturn(TPSUCCESS, 0, rqst->data, 0L. 0),
 }
 tx_commit(),

 tpreturn(TPSUCCESS, 0, rqst->data, 0L. 0),
```

figura 3 3 Código de un servicio Transaccional

3.4.1 Two Phase Commit (Commit de Dos Fases)

Antes de definir este punto se definen tres conceptos escenciales

Precommit se refiere a afectar temporalmente a la base de datos local, esto es, se puede insertar, borrar o modificar algún dato de la base de datos y este cambio sólo lo verá el programa que lo ejecute y tiene la opción de deshacer el cambio

Commit se refiere a hacer el cambio en la base de datos permanente, esto es, que para todos los que accedan a la base de datos verán la nueva información y no podrá deshacerse el cambio

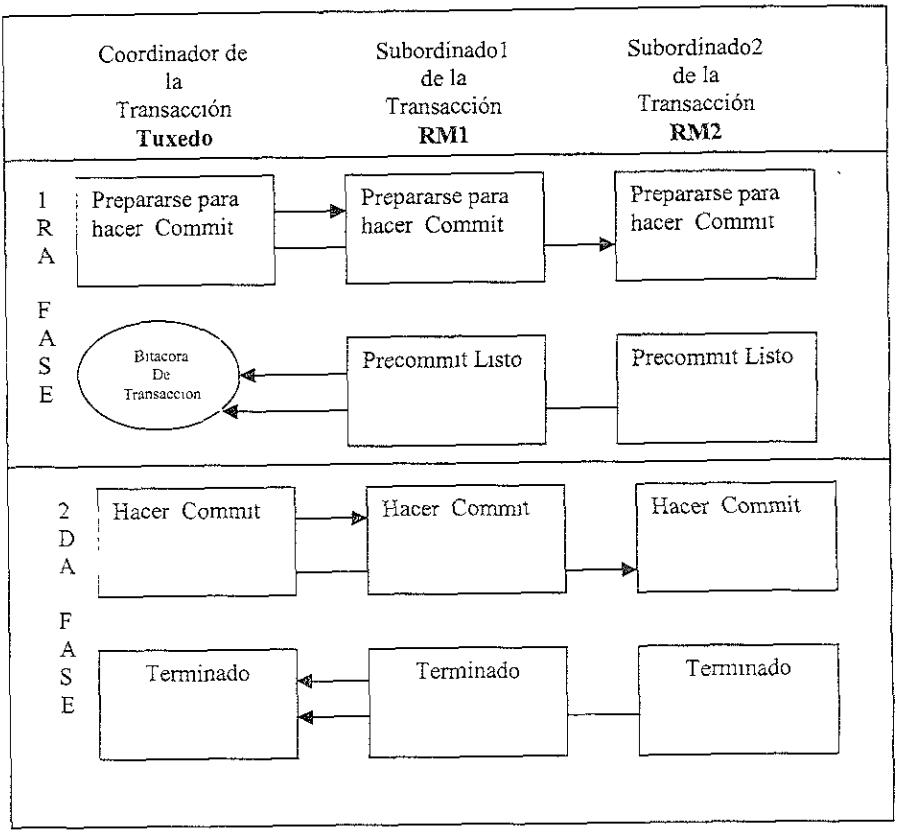
Roll back: se refiere a deshacer los cambios hechos a la base de datos , en la etapa del precommit

En el proceso de two-phase commit, la aplicación es la iniciadora del commit La petición es originada por los programas de la aplicacion por medio de la llamada a la funcion tpcommit() Esta llamada ejecuta un algoritmo del sistema TUXEDO y en este momento TUXEDO juega dos roles, ejecuta el algoritmo e interactúa con los participantes de la transacción, los cuales actúan de manera subordinada ya que TUXEDO es el coordinador de la transacción (Transaction Manager) y se comunica con todos los administradores de recursos (Resource Managers), por ejemplo, las bases de datos que están participando en la transacción. Así es el coordinador de la transacción y es también un subordinado de la misma

Los pasos para un commit global son los que siguen

1 Un commit es requerido por la aplicación Esto es que se haga permanente un cambio, por ejemplo el alta de un nuevo cliente
2. El TUXEDO mandará una petición a todos los administradores de recursos (RM) para que hagan un precommit a sus transacciones
3 Los administradores de recursos ejecutarán el precommit El precommit significa hacer los cambios con opción a recuperar los datos anteriores (abortar)
4. Los administradores de recursos mandaran la respuesta a TUXEDO, diciendo si pudieron hacer el precommit o no
5 Si algún precommit falló, TUXEDO les dira a todos los administradores de recursos que aborten sus transacciones locales (rollback) Si ningún precommit falla, TUXEDO guardará en la bitacora de transacciones (registro de transacciones) la información de la transacción asignándole un identificador
6 TUXEDO dice a todos los administradores de recursos que hagan commit a sus transacciones Cuando los administradores de recursos han hecho commit exitosamente, la función tpcommit regresa a quien lo invocó y el cambio en las bases de datos sera permanente

Enseguida se muestran el commit de dos fasesinvolucrando a dos administradores de recursos (RMs) coordinados por Tuxedo



3 4 Diagrama del mecanismo de las Commit de Dos Fases ("Two Phase Commit")

3.5 Clientes, Servicios, Servidores y Grupos de Servidores

TUXEDO incluye programas que tal vez sólo hagan peticiones a servicios, estos son los CLIENTES . El programa que brinda servicios es el SERVIDOR TUXEDO permite que los servicios hagan peticiones a otros servicios, lo cual permite que un servicio pueda ser también un cliente Un servidor puede brindar uno o un conjunto de servicios, el que un servidor pueda brindar varios servicios debe seguir un análisis para no generar un servidores de bajo rendimiento.

En la figura 3 5 se muestra la relación de clientes, servicios, servidores y grupos de servidores. Un servicio debe brindar una función particular Un grupo de servidores es identificado por TUXEDO como un conjunto que usa un administrador de recursos (Resource Manager), por ejemplo una base de datos particular u otras características particulares

La línea punteada significa que pertenece al recuadro del sistema TUXEDO pero se ha sacado del recuadro para mostrar su relación con la base de datos .

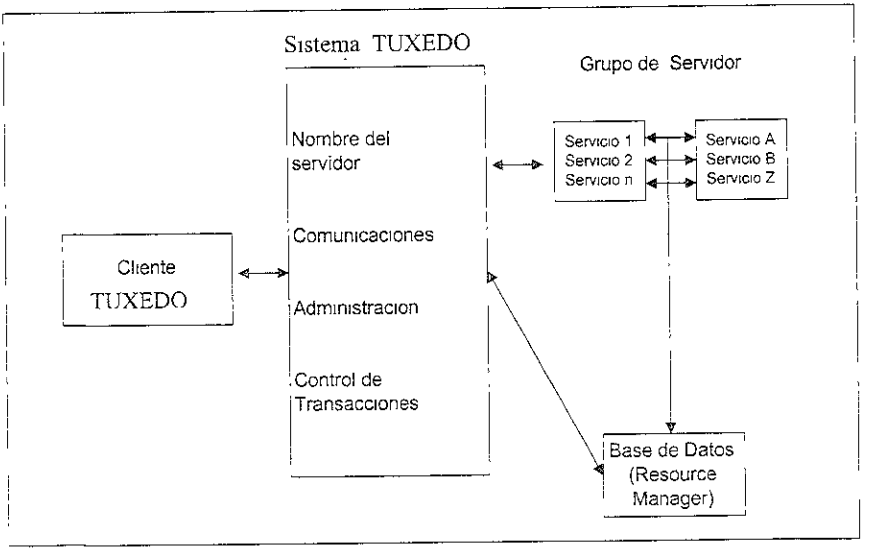


Figura 3 5 Clientes, servicios, servidores y grupos

Para acceder a la misma base de datos por más de un grupo de servidores, se asocia la base de datos con cada grupo de servidores. En la misma aplicacion TUXEDO pueden usarse múltiples tipos de bases de datos (bases de datos heterogéneas), por lo que en la aplicación deben establecerse múltiples grupos de servidores

Todos los servidores que pertenecen a un mismo grupo deben residir en la misma plataforma y la base de datos debe estar también en la misma plataforma Algunas bases de datos proveen características distribuidas, así que la base de datos actual puede que no esté en la misma plataforma del grupo de servidores Pero si la base brinda una interfase que lo permita, TUXEDO considerará que la base de datos es congruente a la del grupo

3.5.1 Archivos de Configuración para Aplicaciones TUXEDO

Los archivos que constituyen la configuración de una aplicación Tuxedo son:

3.5.1 UBBCONFIG

El sistema TUXEDO usa un archivo simple para definir un dominio TUXEDO o una aplicación TUXEDO al momento de levantar la aplicacion El nombre genérico para este archivo es tuxconfig, el cual es un archivo binario utilizado sólo por el software TUXEDO. El archivo fuente del tuxconfig, generalmente se llama ubbconfig El ubbconfig puede ser creado y mantenido usando cualquier editor de texto Este archivo es compilado con el comando tmloadcf para generar el archivo binario

El archivo de configuración ubbconfig esta dividido en diferentes secciones, a continuación se muestran las importantes.

RESOURCES

La sección RESOURCES contiene información de todos los recursos en el dominio, como son el maximo número de grupos, servidores, y servicios, el maximo número de procesos que pueden acceder a un nodo de TUXEDO al mismo tiempo Esta sección también define seguridad.

MACHINES

ción de MACHINES contiene información sobre la plataforma (el servidor nodo, esto es , la máquina donde residirá la aplicación) También incluye la ruta completa de las variables TUXDIR=directorio : está instalado TUXEDO, TUXCONFIG=ruta donde se encontrará el archivo binario de la aplicación, y)IR=ruta completa donde residirá la aplicación, esto es donde se encuentran los servidores

OUPS

sección de GROUPS provee informacion que se refiere a todos los servidores en un grupo y el resource nager asociado con el grupo, o el que no tiene asociado resource manager

ETWORK

a sección de Network provee información de la red, como la dirección para un dominio multinodo, esto se efiere a una aplicación TUXEDO donde un dominio está distribuido en más de una máquina física

SERVERS

En la sección de Servers se listan los nombres de los servidores en un grupo además de información acerca de este servidor. Esta sección puede ser usada para controlar el orden en que se van a "levantar" los servidores

SERVICES

La sección de Servicios contiene los nombres de servicios y pueden ser asociadas con el nombre del servidor al cual pertenecen, también puede ir información extra sobre el servicio, dependiendo del tipo de aplicación Los nombres en esta sección también se pueden omitir

ROUTING

La sección de ROUTING define el dato que se evaluará para el criterio de ruteo así como la definición de este criterio.

La figura 3 6 muestra un ejemplo de archivo ubbconfig y a continuacion de describe cada una de sus secciones Es importante mencionar que las secciones están definidas en idioma Inglés y en este caso no se sustituirán ya que es la sintaxis estricta para crear los archivos de la aplicación TUXEDO

```
RCES
                123465
NID             Sistema1
ER              NODO1
ACCESSERS       4
SERVER          2
SERVICES        4
DEL             SHM
AL              N

IACHINES
EFAULT:
                APPDIR="ruta del directorio de la aplicación"
                TUXCONFIG="ruta del archivo binario/tuxconfig"
                TUXDIR="ruta donde está instalado Tuxedo"

Servidor        LMID= NODO1

*GROUPS
GROUP1
                LMID=NODO1   GRPNO=1   OPENINFO=NONE

*SERVERS
DEFAULT:
                CLOPT="-A"

Servidor1       SRVGRP=GROUP1    SRVID=1

*SERVICES

GeneraCuenta
```

3 6 Ejemplo de ubbconfig

Dominios TUXEDO

)O provee una característica que permite la comunicación de dos o más aplicaciones TUXEDO, o ıa aplicación TUXEDO y otras aplicaciones definidas por otros productos que sigan el estandar OSI y n.

ntinuación se definen algunos términos relevantes para entender el funcionamiento de los dominios :EDO

licación TUXEDO

ıa aplicacion TUXEDO está delimitada por un archivo TUXCONFIG Una aplicación TUXEDO puede omunicarse con otra aplicación TUXEDO a través de un grupo gateway domain.

Dominio TUXEDO

Es una aplicación que es administrada independientemente.

Grupo de *Gateway*[3]

Una colección de procesos que proveen servicios de comunicación entre un dominio y otros dominios

Dominio Local

Es subconjunto de servicios de la aplicación, que están disponibles a otros dominios Un dominio local es siempre representado por un grupo de gateway domain

Servicio Local

Un servicio de un dominio local que está disponible a otros dominios a través de un grupo gateway.

Dominio Remoto

Es una aplicacion Tuxedo diferente a la aplicacion local y a la que se puede acceder transparentemente a través de un grupo gateway, esto significa que se pueden utilizar servicios remotos que éste brinda

---

[3] Gateway se traduce como "Pórtico, salida", en este trabajo se refiere a los procesos que permiten la comunicación transparente entre diferentes aplicaciones Tuxedo (dominios Tuxedo) Se manejará con el vocablo inglés ya que en los archivos de configuración es estricto su uso en la sintaxis.

.emoto

io de un dominio remoto que está disponible para la aplicación local a través de un grupo de

.2.1 Características de Dominios

> que las empresas y sus aplicaciones crecen, es necesario organizar los diferentes segmentos del negocio conjuntos de funcionalidad. Estos segmentos requieren una administración independiente pero al mismo npo se requiere que compartan servicios e información, generalmente estos segmentos podrían estar en erentes máquinas, en diferentes localidades geográficas y probablemente en diferente plataforma. A cada io de estos conjuntos podríamos llamarlos "dominios del negocio"

TUXEDO tiene la capacidad de proveer software para permitir la interoperabilidad entre dominios de negocio que estén construidos con su software. De esta manera una aplicación TUXEDO puede utilizar servicios transparentemente sin saber si está siendo atendido localmente o en algún otro lugar geográfico.

Las características de una configuración con dominios son las siguientes

- Independencia Cada dominio es administrado independientemente
- Control El administrador de cada dominio puede controlar el acceso de otros dominios
- Seguridad Existen tres tipos de seguridad exportación de servicio, control de acceso y password de dominio
- Administración de Transacción Los programadores de la aplicación pueden llamar a un servicio remoto dentro de una transacción. Los dominios coordinan la transaccion remota y la local como una sola y pueden hacer commit o rollback de estas transacciones
- Multiplexión de peticiones Un gateway de dominio puede recibir y procesar peticiones de servicios mandados desde un cliente de máquinas TUXEDO o de dominios remotos

La figura 3.7 muestra un esquema simple de una aplicacion configurada por dominios En esta aplicación participan dos dominios los cuales son de diferente plataforma, esto quiere decir que sin Tuxedo, no serían capaces de intercambiar información. Cada maquina contiene la configuración de su dominio y es administrada localmente Un cliente del dominio 1 podría utilizar un servicio brindando por el dominio 2 y éste no se enteraría que está accediendo a otra aplicacion

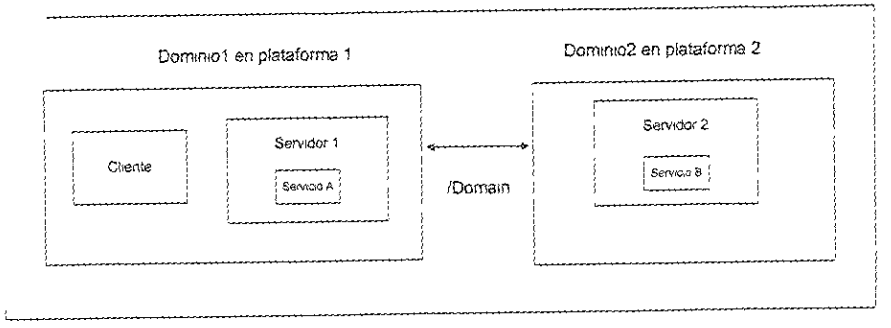


Figura 3 7 Dominios TUXEDO

5.2.2 DMCONFIG

Este archivo de configuración es necesario cuando nuestra aplicación TUXEDO va a interactuar con otra u otras aplicaciones, generalmente otras aplicaciones TUXEDO Es importante mencionar que las secciones estan definidas en idioma Inglés y en este caso no se sustituirán ya que es la sintaxis estricta para crear los archivos de la aplicación TUXEDO Las secciones se describen a continuación

DM_LOCAL_DOMAINS

Esta seccion identifica el dominio local y su grupo gateway asociado. Se especifica el nombre del grupo del servidor gateway y también se indica con que tipo de dominios se puede comunicar éste En esta sección también se define el identificador de este dominio

DM_REMOTE DOMAINS

En esta sección se identifica al conjunto de dominios remotos y sus características, como lo son el tipo de dominio y su identificador

DM_TDOMAIN

En esta sección se define la información de direcciones de cada dominio local y remoto Se especifica la dirección de red usada por un dominio local o un dominio remoto para aceptar conexiones de otros dominios TUXEDO.

DM_LOCAL_SERVICES

Esta sección provee informacion acerca del servicio exportado por cada dominio local Exportar un servicio significa ponerlo disponible para que otros dominios puedan utilizarlo

SERVICES

vee información acerca de los servicios importados y disponibles en los dominios remotos.

s que este dominio puede llamar aunque no pertenezcan a él

n, en la figura 3.8, se muestra un ejemplo de archivo dmconfig

```
AL_DOMAINS
AL                  GWGRP=GWADM_GRP
                    TYPE=TDOMAIN
                    DOMAINID=" DOMLOCAL "
                    DMTLOGDEV="ruta de log de transacciones\DMTLOG"
                    DMTLOGNAME=DMTLOG
REMOTE_DOMAINS
R1                  TYPE=TDOMAIN
                    DOMAINID="DOMR1"
MR2                 TYPE=TDOMAIN
                    DOMAINID="DOMR2"
DM_TDOMAIN
DOMR1     NWADDR="// dirección del servidor.puertoxx"

DOMR2     NWADDR="// direccion del servidor puertoxx"
          CMPLIMIT=600

DMLOCAL           NWADDR="// dirección del servidor:puertoxx"
          CMPLIMIT =600
*DM_LOCAL_SERVICES
ServicioLoc1
ServicioLoc2
*DM_REMOTE_SERVICES
ServicioRem1
ServicioRem2
```

Figura 3 8 Ejemplo del archivo de dmconfig

```
)MAINS
        GWGRP=GWADM_GRP
        TYPE=TDOMAIN
        DOMAINID=" DOMLOCAL "
        DMTLOGDEV ="ruta de log de transacciones\DMTLOG"
        DMTLOGNAME=DMTLOG
```

acción vemos la definición del dominio local, que está asociado a un grupo de es del tipo TDOMAN, esto es un dominio Tuxedo, el nombre del dominio es .OCAL", y por último se muestra dónde estará la bitácora de transacciones llamada OG.

```
REMOTE_DOMAINS
R1          TYPE=TDOMAIN
                DOMAINID="DOMR1"

MR2         TYPE=TDOMAIN
                DOMAINID="DOMR2"
```

En esta sección se definen los dominios Remotos con los cuales se podrá interactuar. En este caso son dos dominios remotos llamados "DOMR1" y "DOMR2".

```
*DM_TDOMAIN
DOMR1     NWADDR="//dirección del servidor puertoxx"

DOMR2     NWADDR="// direccion del servidor puertoxx"
        CMPLIMIT=600

DMLOCAL         NWADDR="// direccion del servidor puertoxx"

CMPLIMIT =60
```

En esta sección se define principalmente las direcciones IP y el puerto por el cual se mantendrá la comunicación tanto del dominio local como del dominio remoto Se define en este caso el parámetro CMPLIMIT comprimir los mensajes que lleguen con un tamaño mayor o igual al definido.

```
SERVICES
```

ción se listan los servicios que el dominio local "DOMLOCAL" brinda a los remotos para que pueden usarlos transparentemente.

```
MOTE_SERVICES
Rem1
oRem2
```

a sección se listan los Servicios que podrá utilizar el dominio local y que aunque no en ese dominio quien haga la petición no tendrá que configurar nada más y no sabrá si siendo atendido local o remotamente.

TUXEDO provee cinco niveles de seguridad en la aplicación Esto permite a los programadores el grado apropiado para la aplicación.

Nivel 1 Se refiere a los permisos de lectura, escritura y ejecución del sistema operativo UNIX. La aplicación tendrá los permisos que el administrador haya asignado al grupo UNIX al cual pertenece la aplicación

Nivel 2· Se refiere a una lista de control de Acceso (ACL). En la cual el administrador genera dos archivos listando usuarios y grupos de servidores y un tercer archivo que relaciona usuarios con grupos de la aplicación TUXEDO

- Nivel 3· Se refiere a seguridad mediante passwords. El administrador establece un password el cual permitirá conectarse a la aplicación

- Nivel 4· Se refiere a una autentificación del cliente Esto es mediante la validación del cliente mediante un servicio del sistema que se especializa en autentificar

- Nivel 5 Se refiere a una autentificación completa del cliente mediante ACL Esto lo hace a nivel de servicios, eventos y colas. a diferencia del nivel 2 que al validar al cliente y le permite utilizar toda la aplicacion

mportantísima de TUXEDO es que puede administrar los recursos de la aplicación así como ta, obteniendo información del sistema en línea, por ejemplo· como cuantos servicios hay, a qué ertenece, en qué estado se encuentran, entre otros datos. A continuación se muestra en la figura e información que se puede obtener del sistema mediante el comando tmadmin :

```
Service Name Routine Name    Prog Name     Grp Name  ID  Machine    # Done Status
-------------------------------------------------------------------------------
TMS          TMS             TMS_SYB_T-    GROUP1 30001 tux_endpo-  3 AVAIL
TMS          TMS             TMS_SYB_T+    GROUP2 30001 tux_endpo-  0 AVAIL
DMADMIN      DMADMIN         DMADM         LDMGRP  10   tux_endpo+  7 AVAIL
tux_endpoint GWS             GWADM         LGWGRP  20   tux_endpo+  0 AVAIL
S404_ActuRe- GWS             GWTDOMAIN     LGWGRP  30   tux_endpo+  9 AVAIL
S404_ValiCo- GWS             GWTDOMAIN     LGWGRP  30   tux_endpo+  0 AVAIL
TMS          GWS             GWTDOMAIN     LGWGRP  30   tux_endpo-  0 AVAIL
C604_ConsRe- C604_ConsRe-    C604resgi+    GROUP1  40   tux_endpo-  0 AVAIL
C604_ActuRe+ C604_ActuRe+    C604resgi+    GROUP1  40   tux_endpo+  0 AVAIL
C604_BajaRe- C604_BajaRe-    C604resgh+    GROUP1  42   tux_endpo-  1 AVAIL
```

Figura 3 9 Pantalla que muestra la información de la aplicación brindada por el tmadmin

En la figura vemos que la primera columna muestra todos los servicios disponibles, los dos primeros TMS son servicios administrativos de TUXEDO que van a llevar el control de las transacciones distribuidas, el DMADMIN es el servicio que permite la administración en línea de los servicios que asociados a dominios; tux_endpoint es servicio de gateway que permite la vía de acceso entre el dominio local y uno remoto en particular, en este caso solamente hay comunicación con un dominio remoto, los siguientes dos servicios son servicios aplicativos que residen en el dominio remoto, el siguiente TMS es el servicio que manejará las transacciones que involucren ambos dominios, los últimos tres servicios son aplicativos y residen en el dominio local

La segunda columna indica a qué rutina pertenece cada servicio, la tercera columna indica a qué servidor pertenecen, la cuarta columna indica a qué grupo de servidores está asociado, la quinta columna indican qué identificador de servidor tienen asociado, la sexta columna indica a que maquina están asociados, en este caso la aplicacion local unicamente se implementó en un equipo UNIX, la columna siete indica cuantas peticiones ha atendido cada servicio y por último se indica en que estado se encuentra el servicio, aquí se encuentran disponibles para atender peticiones

rramientas de administración y monitoreo, se muestra únicamente está ya que es la que se nte y brinda información concentrada y relevante Esta pantalla además de ser informativa r comandos de administración para resolver algunos problemas de la aplicación cuando ésta

oductos BEA

finalidad de lograr los objetivos de un producto robusto para aplicaciones heterogéneas distribuidas, DO se complementa con los siguientes productos:

* **Workstation (/WS).** Este producto provee el soporte completo cliente para sistemas operativos dows, OS/2, MAC, UNIX, y otros

ea Connect (/HOST, /OSI-TP, e-Link). Esta realmente es un afamilia de productos de conectividad, la cual ntegra a las aplicaciones distribuidas TUXEDO con aplicaciones en Mainframe

Bea Jolt Este producto provee acceso vía internet a las aplicaciones TUXEDO, de forma que Bea Jolt toma los requerimientos de clientes Java, traduciéndolos a requerimientos TUXEDO

Bea TUXEDO Builder Este producto consiste en un conjunto de herramientas para el desarrollo de aplicaciones TUXEDO, soportando el desarrollo con herramientas de terceros. Las herramientas de Builder extienden y mejoran herramientas como Microsoft Visual Basic, Sybase Power Builder, Microsoft Visual C++ y Rational Rose.

Bea Manager Esta es una herramienta de administración para operación del sistema TUXEDO

4. DISEÑO TUXEDO

4.1 Arquitectura Tecnológica

El diseño descrito se enfocará al middleware Tuxedo en lo que se refiere a arquitectura, configuración y desarrollo de reglas del negocio. Dado que la configuración y diseño TUXEDO depende en gran medida de la arquitectura tecnológica. se comenzará describiendo esta. Se dividirá la arquitectura en las 3 siguientes partes

1. Arquitectura de la Oficina Local
2 Arquitectura de la Oficina Central
3 Relación de Comunicación entre las Oficinas Locales y la Central

4.1.1 Arquitectura de la Oficina Local

Cada oficina local cuenta con el siguiente equipo de hardware y sistema operativo

- Un servidor NT, un servidor HP, sistema operativo Unix HP-UX
- Una base de datos del sistema anterior, Informix
- Una base de datos del Sistema Nuevo, Informix
- Computadoras personales, Windows 95

4.1.2 Arquitectura de la Oficina Central

La oficina Central cuenta con el siguiente equipo de hardware y sistema operativo.

- Un servidor IBM, sistema operativo Unix - AIX
- Una base de datos del sistema anterior, Informix
- Una base de datos del Sistema Nuevo, Informix

4.1.3 Comunicación entre Oficinas Locales y Oficina Central

Existe una comunicación compleja ya que para llevar a cabo cualquiera de las operaciones es necesario consultar los datos del cliente en la base de datos local y posteriormente en la base de datos Central Existe también el caso en el cual es necesario acceder la base de datos de otra oficina local Este tipo de comunicación es en línea y debe ser transparente para el usuario. La figura 4 1 muestra los tres puntos en que se divide la arquitectura, se retoma el diseño general que se definió en el capítulo de Arquitectura para detallarlo

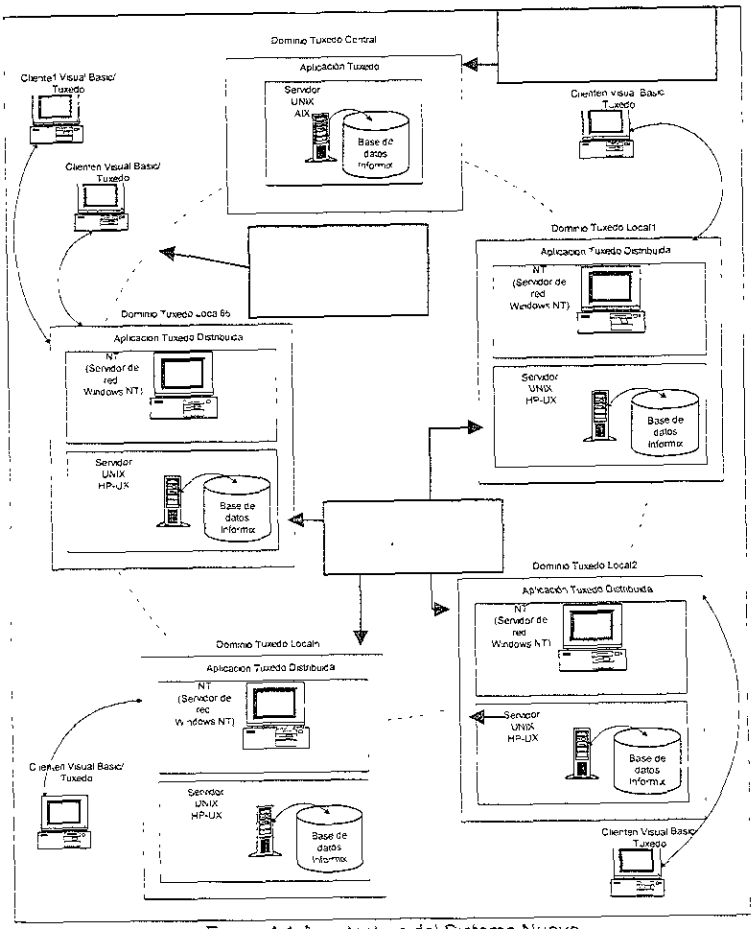


Figura 4 1 Arquitectura del Sistema Nuevo

La Arquitectura de la Oficina Central tiene los siguientes elementos:

1 Sistema Anterior: aparece ya que éste convivirá con el Sistema Nuevo hasta que se actualicen todos los datos
2 Servidor IBM: servidor que contiene la aplicación TUXEDO
3 Base de datos Anterior y que también es afectada por la aplicación
4 Base de datos donde se registra la información del Sistema Nuevo
5 Servicios TUXEDO que pertenecen a grupos transaccionales

La Arquitectura de la Oficina Local tiene los siguientes elementos:

1 Computadoras personales donde se ejecuta el cliente construido en Visual Basic
2 Servidor NT, que es el nodo principal en la aplicación TUXEDO distribuida
3. Servicios que están en plataforma NT y no tienen un Administrador de Recursos (Resource Manager) asociado, por lo que no es transaccional
4 Complemento de la aplicación distribuida, es el nodo respaldo en UNIX HP-UX. En este servidor se encuentran los servicios transaccionales
5 Base de datos anterior y se utiliza para convivencia.
6 Base de Datos nueva donde se registra la información del sistema local
7 Servicios TUXEDO que son transaccionales.

4.2 TUXEDO Distribuido

El sistema Global es un sistema distribuido, ya que interactúan diferentes plataformas y bases de datos distribuidas en toda la República Mexicana, pero cabe aclarar que cada dominio local es en sí mismo también es un dominio distribuido dado el requerimiento de utilización de dos servidores, de esta forma se define la configuración Local como TUXEDO distribuido, siendo el servidor NT el equipo primario y el servidor Unix, el equipo de backup, es decir que la configuración y administración se harán desde el servidor NT y el servidor UNIX está definido como respaldo del sistema, pudiendo migrar la aplicación al servidor UNIX por alguna contingencia

Existe un grupo de servidores XA en la Unix, esto es, que entraran en modo transaccional, y un grupo de servidores para operaciones no-XA en la NT, esto es, que seran operaciones que no serán controladas como una transaccion global También se define un grupo para los servidores de colas y un grupo de servidores administrativos para la implementación de dominios.

4.3 Dominios TUXEDO

Algunos datos en la Local, se deben reflejar en Central. Para realizar las transacciones, la Local y Central se comunican usando una configuración de dominios de TUXEDO, como se muestra mas adelante.

Debido a la inestabilidad de las lineas de comunicacion, los buffers se deben enviar a la oficina Central mediante colas (/Q), esto es, almacenarlos en disco hasta que el servicio correspondiente pueda atenderlo. Esto para las operaciones de replicación, ya que no necesitan ser ejecutadas en linea.

En la figura 4 2 se esquematiza el diseño del dominio Central y Local así como sus elementos En el primer cuadro se esquematiza uno de los 65 nodos Locales, los cuales son distribuidos y configurados por dominios. En el segundo recuadro se muestra el nodo Central, el cual brinda información a todas las locales en tiempo real vía dominios Tuxedo. También se muestran los dos archivos de configuración

El detalle de la configuracion Tuxedo de este diseño se expuso en el capítulo 3

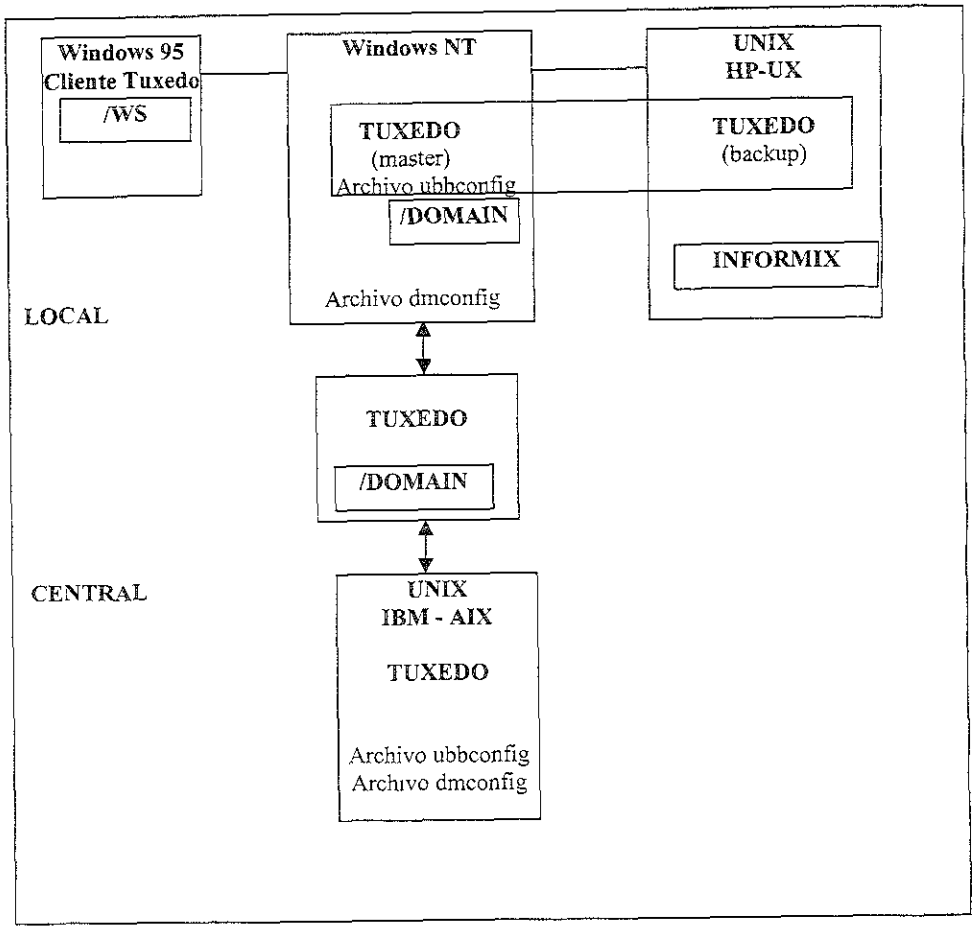


Figura 4 2 Diseño Tuxedo, Local y Central vía Dominios

4.4 Diagrama de Comunicación

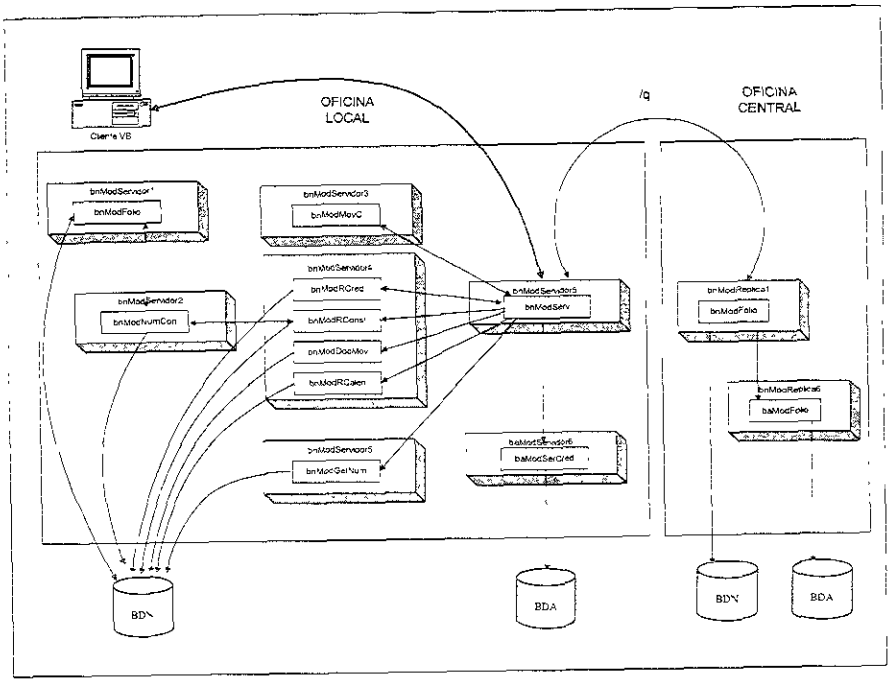
Transacción TUXEDO

La figura 4.3 muestra el tipo de transacciones que lleva a cabo el Sistema Nuevo, así como el llamado entre servicios para realizar la transacción lógica del negocio. Los recuadros dentro de los servidores son los servicios.

- Empieza con un evento en el lado del Cliente Visual Basic
- Se dispara una transacción TUXEDO llamando a un servicio que tiene la lógica del negocio
- Este servicio llama a un conjunto de servicios o funciones que acceden a la base de datos
- Se realiza el Two Phase Commit para asegurar la integridad de la transacción.
- Por último se encola el mensaje en la oficina Central para que ésta replique los cambios en su base de datos cuando tenga tiempo

Los nombres de los servidores y servicios siguen los estándares definidos

bn	Base de datos a la cual acceden
Mod	Módulo del sistema al que pertenecen
Folio	Identifica la operación aplicativa que realiza
BDA	Base de Datos Anterior
BDN	Base de Datos Nueva



4 3 Transacción Global TUXEDO

4.5 Archivos de Configuración

4.5.1 UBBCONFIG

El siguiente archivo es ubbconfig y éste se define como un dominio TUXEDO La configuración que presenta corresponde a una aplicación distribuida ya que administra dos equipos como si fuera un solo y accede a dos bases de datos diferentes.
Entre otros puntos se definen en este archivo

- Máquina Maestra. esto es, la que lleva la configuración
  NT
- Tipo de aplicación "Distribuida"
  NT - UNIX
- Se definen grupos, administrativos y aplicativos
  Grupos Administrativos : Work Station Handler Dominios y Colas
  Grupos Aplicativos: Grupos No Transaccionales NT, Grupos Transaccionales base de datos Anterior, Grupos Transaccionales asociados a la Base de datos Nueva.

- Se define el Resource Manager (Base de datos)
  Informix
- Se asocial el RM a los servidores grupos correspondientes
  TMS_INFX_TUX64
- Maximo número de servidores permitidos
  250
- Máximo numero de servicios permitidos
  650
- Se define el log (bitácora) de transacciones globales
  ULOG FECHA
- Se configuran reestablecimiento de servidores
  RESTART = Y
- Se configura ruteo para dominios
  Se ha definido el ruteo en el campo LOCACION del buffer FML Si el valor de LOCACION no es la del dominio actual, ira al archivo de dominios para saber a qué dominio redireccionarse

- Se definen servidores para colas.

TMQUEUE y TMQFORWARD

- Se definen servidores para comunicacion con clientes

  WSL

- Se definen servidores para comunicación entre dominios

  DMADM, GWADM y GWTDOMAIN

- Se definen los servidores aplicativos

  Ejemplo. Nvo_Consultas_DOMCENTRAL

  SRVGRP=NVONT_GRP : Grupo al cual pertenece

  SERVID=1090 Número único que lo identifica

  CLOPT="-A -- -BD BDNueva –INS DOM5_tcp" Parámetros enviados al servidor como la base de datos y la instancia a la cual conectarse.

- Se definen las direcciones y puertos de comunicacion

  Estas direcciones y puertos han sido modificados para seguridad del cliente

- Se definen criterios de ruteo

  Existen 4 criterios definidos para 4 grupos de servidores: NT base de datos Anterior, NT base de datos Nueva. UNIX, y colas

- No se definió seguridad ya que se implementó otro producto para este fin

La figura 4 4 muestra el archivo de configuración ubbconfig que se acaba de describir y que corresponde a un nodo Local de los 64 que componen la aplicación total.

```
# UBB OFICINA LOCAL
# Archivo - Configuracion Aplicacion TUXEDO
# Softtek - CSMC
# 05/14/1998

*RESOURCES
IPCKEY          40000
DOMAINID        DOM028_NUEVO
MASTER          DOM028NT,DOM028UX
MAXSERVERS      250
MAXSERVICES     650
MODEL           MP
LDBAL           Y
OPTIONS         LAN,MIGRATE
MAXGTT          1000
SCANUNIT        10

*MACHINES
ServidorNt      LMID=DOM028NT
                MAXACCESSERS=200
              APPDIR="c:\tuxapp\NUEVO"
          TUXCONFIG="c:\tuxapp\NUEVO\tuxconfig"
          TUXDIR="c:\TUXEDO"
          TLOGDEVICE="c:\tuxapp\NUEVO\TLOG"
          ULOGPFX="c:\tuxapp\NUEVO\ULOG"
          TLOGNAME=TLOG
          UID=0
          GID=0
          TYPE=WinNT


ServidorUnix            LMID=DOM028UX
        MAXWSCLIENTS=20
        MAXACCESSERS=140
        APPDIR="/usr/users/softtek/NUEVO"
        TUXCONFIG="/usr/users/softtek/NUEVO/tuxconfig"
        TUXDIR="/opt/TUXEDO"
        TLOGDEVICE="/usr/users/softtek/NUEVO/TLOG"
        ULOGPFX="/usr/users/softtek/NUEVO/ULOG"
        TLOGNAME=TLOG
        UID=860
```

```
      GID=859

#*************************************************
#                GROUP
#*************************************************
*GROUPS

GWADM_GRP    LMID=DOM028NT                GRPNO=200          OPENINFO=NONE

QSTK_GRP     LMID=DOM028NT                GRPNO=400          OPENINFO=NONE

#************** COLAS DE USO LOCAL Y ENVIO REMOTO ***************************

QENQUE_GRP   LMID=DOM028NT  GRPNO=500
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c \tuxapp\NUEVO\DEVQUE,STK_ENQUE_SPACE"

QINSAE_GRP   LMID=DOM028NT  GRPNO=600
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c \tuxapp\NUEVO\DEVQUE,SPACIO1"

QCMCAN_GRP   LMID=DOM028NT  GRPNO=700
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c \tuxapp\NUEVO\DEVQUE,ESPACIO2"

QAA_GRP              LMID=DOM028NT  GRPNO=800
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c \tuxapp\NUEVO\DEVQUE,ESPACIO4"

QOTRO_GRP    LMID=DOM028NT  GRPNO=900
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c \tuxapp\NUEVO\DEVQUE,ESPACIO5"

QSPACE_GRP   LMID=DOM028NT  GRPNO=1000
             TMSNAME=TMS_QM TMSCOUNT=2
             OPENINFO="TUXEDO/QM,c  tuxapp\NUEVO\DEVQUE,QSPACE"

NVONT_GRP    LMID=DOM028NT                GRPNO=1100         OPENINFO=NONE

ANTNT_GRP    LMID=DOM028NT                GRPNO=1200         OPENINFO=NONE
NT_GRP2      LMID=DOM028NT                GRPNO=1300         OPENINFO=NONE
             TMSNAME=TMS      TMSCOUNT=2
```

```
WSL_GRP         LMID=DOM028UX                   GRPNO=1700              OPENINFO=NONE

#*** GRUPOS DE UNIX *********

NVOUX_GRP       LMID=DOM028UX           GRPNO=1400
                  TMSNAME=TMS_INFX_TUX64 TMSCOUNT=2
                  OPENINFO="INFORMIX-ONLINE  bdNueva "

SERIALUX_GRP            LMID=DOM028UX           GRPNO=1450
                  TMSNAME=TMS_INFX_TUX64 TMSCOUNT=2
                  OPENINFO="INFORMIX-ONLINE bdNueva"


ANTUX_GRP       LMID=DOM028UX           GRPNO=1500
                  TMSNAME=TMS_INFX_TUX64 TMSCOUNT=2
                  OPENINFO="INFORMIX-ONLINE bdAnterior"

UX_GRP          LMID=DOM028UX           GRPNO=1600

#*************************************************
#                   NETWORKING
#*************************************************
*NETWORK
DOM028NT    NADDR="//11 11 111 111 92000"
          NLSADDR="//11 11 111 111 92002"

DOM028UX    NADDR="//22 22 222 222 92000"
            BRIDGE="/dev/lan0"
            NLSADDR="//22 22 222 222 92002"
#*************************************************
#                   SERVIDORES
#*************************************************
*SERVERS
DEFAULT  RESTART=Y          MAXGEN=5      REPLYQ=Y

# ************************************************
# SERVIDORES DE ADMINISTRATIVOS
# ************************************************

 WSL                            SRVGRP=UX_GRP SRVID=10
                          CLOPT="-A -- -n //22 22 222 222 24002 -d /dev/lan0 -x 4 -m 5 -M 10 -p 4050 -P
4080"
```

```
# ***** SERVIDORES DOMINIOS ********

DMADM                   SRVGRP=GWADM_GRP SRVID=20    REPLYQ=N
GWADM                   SRVGRP=GWADM_GRP SRVID=30    REPLYQ=Y
GWTDOMAIN               SRVGRP=GWADM_GRP SRVID=40    REPLYQ=Y

# ***** SERVIDORES DE COLAS ********

TMQUEUE                 SRVGRP=QENQUE_GRP SRVID=50      GRACE=0 CONV = N
MAXGEN=10
                        CLOPT="-s STK_ENQUE_SPACE TMQUEUE --" REPLYQ=N
TMQFORWARD              SRVGRP=QENQUE_GRP SRVID= 60     GRACE=0 RESTART=Y CONV=N
MAXGEN=10 REPLYQ=N
                        CLOPT="-- -i 2 -n -q STK_ENQUESVC "

TMQUEUE                 SRVGRP=QINSAE_GRP SRVID=70GRACE=0    RESTART = Y
CONV= N MAXGEN=10
                        CLOPT="-s ESPACIO1 TMQUEUE --" REPLYQ=N
TMQFORWARD              SRVGRP=QINSAE_GRP  SRVID=80     GRACE=0 RESTART=Y CONV=N
MAXGEN=10 REPLYQ=N
                        CLOPT="-- -i 2 -n -q ESPACIO1 "

TMQUEUE                 SRVGRP=QCMCAN_GRP SRVID=90      GRACE=0  RESTART
= Y CONV= N MAXGEN=10
                        CLOPT="-s ESPACIO2 TMQUEUE --" REPLYQ=N
TMQFORWARD              SRVGRP=QCMCAN_GRP  SRVID=100    GRACE=0 RESTART=Y CONV=N
MAXGEN=10 REPLYQ=N
                        CLOPT="-- -  2 -n -q ESPACIO2 "

TMQUEUE                 SRVGRP=QAA_GRP   SRVID=110 GRACE=0    RESTART = Y
CONV= N MAXGEN=10
                        CLOPT="-s ESPACIO3 TMQUEUE --" REPLYQ=N
TMQFORWARD              SRVGRP=QAA_GRP       SRVID=120          GRACE=0  RESTART=Y
CONV=N MAXGEN=10 REPLYQ=N
                        CLOPT="-- -i 2 -n -q ESPACIO3 "

TMQUEUE                 SRVGRP=QOTRO_GRP  SRVID=120      GRACE=0  RESTART
= Y CONV= N MAXGEN=10
                        CLOPT="-s ESPACIO4 TMQUEUE --" REPLYQ=N
TMQFORWARD              SRVGRP=QOTRO_GRP   SRVID=130   GRACE=0 RESTART=Y CONV=N
MAXGEN=10 REPLYQ=N
```

```
                    CLOPT="-- -i 2 -n -q ESPACIO4 "

########## SERVIDORES DE NT/INFORMIX ######################

 Nvo_TG_DOM          SRVGRP=NVONT_GRP SRVID=1010  CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
 Nvo_TG2_DOM         SRVGRP=NVONT_GRP SRVID=1020  CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
 Nvo_Consultas_DOM           SRVGRP=NVONT_GRP SRVID=1030  CLOPT="-A -r -- -BD BDNueva -INS
DOM5_tcp "
                             MIN=1 MAX=5 RESTART=Y MAXGEN=5
 Nvo_Comun_DOM               SRVGRP=NVONT_GRP  SRVID=1040  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
                             MIN=2 MAX=5 RESTART=Y MAXGEN=5
 Nvo_TG1_DOM         SRVGRP=NVONT_GRP SRVID=1050  CLOPT="-A "
 Nvo_TG3_DOM         SRVGRP=NVONT_GRP SRVID=1060  CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
 Nvo_GeneraRfc_DOM           SRVGRP=NVONT_GRP  SRVID=1070  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
 Nvo_ValidaRfcDOM            SRVGRP=NVONT_GRP  SRVID=1080  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
 Nvo_ValidaRfc       SRVGRP=NVONT_GRP SRVID=1090  CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
 NvoConCump_DOM              SRVGRP=NVONT_GRP  SRVID=1100  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
 Nvo_Consultas_DOMCENTRAL    SRVGRP=NVONT_GRP  SRVID=1120  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
                             MIN=1 MAX=5 RESTART=Y MAXGEN=5
 Nvo_Cat_DOM         SRVGRP=NVONT_GRP SRVID=1130  CLOPT="-A -- -BD NUEVO -INS DOM5_tcp "
 Nvo_Consultas1_DOM  SRVGRP=NVONT_GRP SRVID=1140  CLOPT="-A -- -BD NUEVO -INS DOM5_tcp "
                          MIN=1 MAX=5 RESTART=Y MAXGEN=5
 Nvo_NoTran_Serial_DOMCENTRAL SRVGRP=NVONT_GRP SRVID=1150  CLOPT="-A -- -BD BDNueva -INS
DOM5_tcp "
                             MIN=1 MAX=1 RESTART=Y MAXGEN=5
 Nvo_Repl_DOM        SRVGRP=NVONT_GRP SRVID=1160  CLOPT="-A -- -BD BDNueva -INS DOM5_tcp
"


 Ant_Conv_DOM        SRVGRP=ANTNT_GRP SRVID=1170 CLOPT="-A -- -BD BDAnterior -INS DOM5_tcp
"
 Ant_Conv2_DOM               SRVGRP=ANTNT_GRP  SRVID=1180  CLOPT="-A -- -BD BDAnterior -INS
DOM5_tcp "
 Ant_Conv1_DOM               SRVGRP=ANTNT_GRP  SRVID=1190  CLOPT="-A -- -BD BDAnterior -INS
DOM5_tcp "

 NvoNVServ           SRVGRP=NVONT_GRP SRVID=1195 CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
```

```
NvoCoServ          SRVGRP=NVONT_GRP SRVID=1197 CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
NvoProcesos        SRVGRP=NVONT_GRP SRVID=1198 CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
SRVDynamic         SRVGRP=NVONT_GRP SRVID=1200 CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
                           MIN=3 MAX=10 RESTART=Y MAXGEN=5
SRVDynamicConv     SRVGRP=NVONT_GRP SRVID=1210 CLOPT="-A -- -BD BDNueva -INS DOM5_tcp "
            CONV = Y

#********** SERVIDORES DE UNIX/TMS_INFORMIX **********************

Nvo_Tran_DOM       SRVGRP=NVOUX_GRP SRVID=1430 CLOPT="-A "
Nvo_Tran_Comun_DOM SRVGRP=NVOUX_GRP SRVID=1435 CLOPT="-A "
                           MIN=2 MAX=5 RESTART=Y MAXGEN=5
Nvo_Tran_CU_DOM            SRVGRP=NVOUX_GRP SRVID=1440 CLOPT="-A "
Nvo_Tran_Comun2_DOM SRVGRP=NVOUX_GRP SRVID=1450 CLOPT="-A "
                           MIN=2 MAX=5 RESTART=Y MAXGEN=5

Nvo_Tran_CUN2_DOM          SRVGRP=NVOUX_GRP SRVID=1460 CLOPT="-A "
Nvo_Tran_CUN1_DOM          SRVGRP=NVOUX_GRP SRVID=1470 CLOPT="-A "
Nvo_Tran_Comun3_DOM SRVGRP=NVOUX_GRP SRVID=1480 CLOPT="-A "
                           MIN=1 MAX=5 RESTART=Y MAXGEN=5

Nvo_Tran_CUN3_DOM          SRVGRP=NVOUX_GRP SRVID=1490 CLOPT="-A -e stdout "
Nvo_CU_DOM         SRVGRP=NVOUX_GRP SRVID=1500 CLOPT="-A "
Nvo_Tran_CUN4_DOM          SRVGRP=NVOUX_GRP SRVID=1510 CLOPT="-A "
Nvo_Tran_Serial_DOM  SRVGRP=SERIALUX_GRP SRVID=1520 CLOPT="-A "
Nvo_Tran_Serial_Folio SRVGRP=SERIALUX_GRP SRVID= 1525 CLOPT="-A "
Nvo_Tran_CU_DOMCENTRAL     SRVGRP=NVOUX_GRP SRVID=1530 CLOPT="-A "
Nvo_Tran_CUN_DOMCENTRAL    SRVGRP=NVOUX_GRP SRVID=1540 CLOPT="-A "
Nvo_Tran_Comun_DOMCENTRAL SRVGRP=NVOUX_GRP SRVID=1550 CLOPT="-A "
                           MIN=2 MAX=5 RESTART=Y MAXGEN=5
NvoSURetroOVDOM            SRVGRP=NVOUX_GRP SRVID=1580 CLOPT="-A "
Nvo_OV_TG                  SRVGRP=NVOUX_GRP SRVID=1590 CLOPT="-A "
NvoSUAB                    SRVGRP=NVOUX_GRP SRVID=1600 CLOPT="-A "
NvoPMLoc           SRVGRP=NVOUX_GRP SRVID=1610 CLOPT="-A "
NvoPMLocTemp       SRVGRP=NVOUX_GRP SRVID=1620 CLOPT="-A "
NvoPMasivo         SRVGRP=NVOUX_GRP SRVID=1630 CLOPT="-A "
NvoPMLocNac        SRVGRP=NVOUX_GRP SRVID=1640 CLOPT="-A "
NvoPMDarSir        SRVGRP=NVOUX_GRP SRVID=1650 CLOPT="-A "
NvoPMas_LN         SRVGRP=NVOUX_GRP SRVID=1660 CLOPT="-A "

#*****************************************
#              SERVICIOS
```

```
#**************************************
*SERVICES
NvoTraContriTG                                          #Nvo_TG_DOM
NvoBuscaREsc                    ROUTING="NUEVONT"       #Nvo_TG2_DOM
AntSuspDest                     ROUTING="ANTNT"             #Ant_Conv2_DOM
AntReanActDes                   ROUTING="ANTNT"             #Ant_Conv2_DOM
NvoTraContrib       AUTOTRAN=Y  TRANTIME=120            #Nvo_Tran_CUN4_DOM
NvoCamDom           AUTOTRAN=Y  TRANTIME=120            #Nvo_Tran_CUN4_DOM
NvoRHEstab          AUTOTRAN=Y  ROUTING="NUEVOUNIX"         #Nvo_Tran_CUN4_DOM
NvoRMovTR           AUTOTRAN=Y                          #Nvo_Tran_Comun3_DOM
NvoRRep             AUTOTRAN=Y  ROUTING="NUEVOUNIX"         #Nvo_Tran_Comun3_DOM
NvoRMovC            AUTOTRAN=Y  ROUTING="NUEVOUNIX"         #Nvo_Tran_Comun3_DOM

*ROUTING
NUEVONT         FIELD=LOCACION              BUFTYPE="FML"
                RANGES="'28' NVONT_GRP, * GWADM_GRP"
ANTNT           FIELD=LOCACION              BUFTYPE="FML"
                RANGES="'28' ANTNT_GRP, * GWADM_GRP"
NUEVONT_Q       FIELD=LOCACION              BUFTYPE="FML"
                RANGES="'28' QSTK_GRP, * GWADM_GRP"
NUEVOUNIX       FIELD=LOCACION              BUFTYPE="FML"
                RANGES="'28' NVOUX_GRP, * GWADM_GRP"
```

Figura 4 4 Archivo ubbconfig del Sistema Nuevo

4.5.2 DMCONFIG

Este archivo contiene la configuración necesaria para que la aplicación se comporte como un dominio Esto es, que pueda brindar servicios a otras aplicaciones remotamente así como permitir que la aplicación local puede utilizar servicios de otros dominios y que para el usuario sea transparente en donde se está llevando a cabo la operación

La siguiente configuración de dominios fue una de las tareas que tuve a cargo para configurar y probar. El siguiente archivo contempla tres dominios, aunque la aplicación en producción tiene configurados 65 dominios En la sección de servicios locales y servicios remotos se ha puesto una muestra de servicios publicados ya que se extendería demasiado esta sección para este trabajo

Los puntos que se configuran en este archivos se muestran en la figura 4 5 y se describen a continuación:

- Definir Dominios TUXEDO

  En la sección *DM_LOCAL_DOMAINS se define el dominio local

  En la sección *DM_REMOTE_DOMAINS se definen los dominios remotos.

- Definir direcciones y puertos de dominios TUXEDO

  En la sección DM_TDOMAIN se definen las direcciones de todos los dominios así como el puerto de comunicación. Este puerto debe ser el mismo para los dominios

- Definir servicios locales

  En la seccion DM_LOCAL_SERVICES se listan los servicios que el dominio local va a exportar a otros dominios

- Definir servicios remotos

  En la sección DM_REMOTE _SERVICES se listan los servicios que el dominio local va a importar y a que dominio va a redireccionar la petición

- Definir criterio de ruteo entre dominios

  Cuando se llama a un servicio remoto, existe la posibilidad de que éste se haya configurado con la propiedad de ROUTING, esto es, que este servicio lo puede ejecutar cualquiera de los 65 dominios de la aplicacion. En la seccion *DM_ROUTING se especifica a qué dominio se enviará la petición

```
# DOM 028
# Softtek –Sandra Sánchez Rangel - SSR1

*DM_LOCAL_DOMAINS
DOM028                  GWGRP=GWADM_GRP
                        TYPE=TDOMAIN
                        DOMAINID="DOM028"
                        DMTLOGDEV="c \tuxapp\NUEVO\DMTLOG"
                        DMTLOGNAME=DMTLOG
*DM_REMOTE_DOMAINS
CENTRAL                   TYPE=TDOMAIN
                        DOMAINID="CENTRAL"

DOM016                  TYPE=TDOMAIN
                        DOMAINID="DOM016"
*DM_TDOMAIN
DOM028          NWADDR="//11.11 111.111:4006"

CENTRAL         NWADDR="//33 33.333.333:4006"
        CMPLIMIT=600

DOM016          NWADDR="//44 44 44 444:4006"

CMPLIMIT =600

*DM_LOCAL_SERVICES

NvoGeneraRfc            LDOM=DOM028
NvoRMovAI               LDOM=DOM028
NvoRMovRI               LDOM=DOM028
NvoRMovC                LDOM=DOM028
NvoRMovN                LDOM=DOM028
NvoLUltFolio            LDOM=DOM028
NvoUFolioRuteo          LDOM=DOM028
AntRSirCENTRAL      LDOM=DOM028
```

```
NvoReplCanCont              LDOM=DOM028

#*********************************************************************
#                  SERVICIOS REMOTOS
#*********************************************************************
*DM_REMOTE_SERVICES

NvoTGRRegCont                    LDOM=DOM028          RDOM=CENTRAL
NvoClasCont                 LDOM=DOM028          RDOM=CENTRAL
NvoClasifCont               LDOM=DOM028          RDOM=CENTRAL
NvoClasMov                  LDOM=DOM028          RDOM=CENTRAL
NvoGetDOM                   LDOM=DOM028          RDOM=CENTRAL
NvoGetDOMI                  LDOM=DOM028          RDOM=CENTRAL
NvoGetDOMcc                 LDOM=DOM028          RDOM=CENTRAL
NvoGetRfc                   LDOM=DOM028          RDOM=CENTRAL

#       R U T E O
#INSCRIPCIONES
NvoCCEsc              LDOM=DOM028          ROUTING=CRITERIO
NvoHisClasCont        LDOM=DOM028          ROUTING=CRITERIO
NvoMovEscDOM          LDOM=DOM028          ROUTING=CRITERIO
NvoActNoSubsEs        LDOM=DOM028          ROUTING=CRITERIO
NvoMovNoSubsEs        LDOM=DOM028          ROUTING=CRITERIO
NvoUFolioRuteo        LDOM=DOM028          ROUTING=CRITERIO

*DM_ROUTING
CRITERIO        FIELD=RUTEO BUFTYPE="FML"
                RANGES="'0' CENTRAL, '16':DOM016"
```

Figura 4 5 Archivo dmconfig del Sistema Nuevo

4.6 Desarrollo TUXEDO

El siguiente código es una muestra del tipo de servicios que se programaron en el desarrollo TUXEDO. Estos servicios están construidos utilizado la infraestructura que desarrolló Softtek para optimizar el tiempo de construcción y minimizar los errores.

4.6.1 Características de la Infraestructura

- Se implementa un lenguaje **nuevo** entre la aplicación TUXEDO, basado en librerías y clases.
- Se agregan parámetros propios de la infraestructura.
- Se elimina validación de errores TUXEDO, la infraestructura valida éstos, haciendo el desarrollo rápido y de mayor calidad
- El 95% de las instrucciones TUXEDO han sido sustituidas por funciones a las cuales únicamente se le envían los parámetros a procesar y datos definidos por la infraestructura
- No llama directamente a TUXEDO, utiliza funciones propias de la infraestructura
- La infraestructura valida y controla los errores TUXEDO

4.6.2 Características Generales

Las características generales son:

Servicios TUXEDO

- Existe un conjunto de librerías que manejan gran parte de las funciones TUXEDO.
- Estas librerías contienen las funciones necesarias para obtener datos del buffer, subir datos al buffer, llamar a otro servicio, alojar y liberar el buffer, copiar buffers, mandar un servicio a una cola, quitar de una cola un servicio, etc.
- La infraestructura maneja únicamente buffer FML
- Los datos requeridos por la infraestructura para el manejo del buffer son número de columnas, número de renglones, tipo de dato, nombre de bloque y mandar por lo menos un dato

4.6.3 Muestra de Servicio TUXEDO

```
/*****************************************************************
* Programa          NvoRPrincip.ec
** Proyecto         NUEVO
** Caso de Uso      Comun2
** Descripcion      Guarda la Relacion de una empresa Fusionada
*                              a una Empresa Fusionante
** Constructora     Softtek - Sandra Sanchez Rangel (SSR1)
** Fecha            23 febrero 1998
** Modificacion
** Fecha de Mod
*****************************************************************/
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <atmi.h>
#include <userlog.h>
#include <fml.h>
#include <GrlTuxBufLib.h>
#include <Grltmldyn_flds.h>
#include <GrlTuxTransLib.h>
#include <NVO_FuncionesBD.h>
#include <GrlTuxTransLib.h>
#include <stdlib.h>
#include <catTpClaves.h>
#include <constantes.h>
#include <GrlCatalogos.h>
#include <catMovimientos.h>
#include <catResultado.h>
#include <catClasificacion.h>
EXEC SQL INCLUDE SQLCA
EXEC SQL include datetime;
EXEC SQL INCLUDE "GrlEntidades.h"

#define cRFC_SIZE          13

#ifndef __DEBUG
/*#define __DEBUG*/
#endif

#ifndef iSQLSUCCESS
#define iSQLSUCCESS        0
#endif
```

```
#ifdef __cplusplus
extern "C"
#endif
#if defined(_STDC_) || defined(_cplusplus)
int dFRfcRPrincip(struct uRelacion *LuvRelacion, struct uMovimiento *LuvMovimiento)
#else
int dFRfcRPrincip(LuvRelacion, LuvMovimiento)
struct uRelacion *LuvRelacion,
struct uMovimiento *LuvMovimiento,
#endif
{
  /* declaracion de variables */

  FBFR    *LbuffRcv,
  TPSVCINFO Lrqst,

  EXEC SQL BEGIN DECLARE SECTION,
  struct          uClasCont LuvClasCont,
  char            LcFechaOp[nFECHATIME_LEN],
  char            LcYear[nFECHAYEAR_LEN],
  char            LcMonth[nFECHAMONTH_LEN],
  char            LcDay[nFECHADAY_LEN],
  char            LcHour[nFECHAHOUR_LEN],
  char            LcMin[nFECHAMIN_LEN]
  char            LcSec[nFECHASEC_LEN],
  char            LcFeRecepcion[nFECHADAY_LEN],
  int             LiNo_OperacionDos,
  int             LiFolioDos,
  int             LiDOMPrincipal=0,

  EXEC SQL END DECLARE SECTION,

  int  LiTipoAtt,
  long LiLen, LiLong
  int L-Ret,


#ifdef __DEBUG
  userlog("FUNCION  dFRfcRPrincip",
#endif

  LiDOMPrincipal=LuvRelacion->iDOMOperacion,

  /*
   * Informacion de entrada
   *
```

```
#ifdef __DEBUG
 userlog("dFRfcRPrincip  servidor >> EL RFC PRINCIPAL ES [%s]",LuvRelacion->cRfcPrincipal),
 userlog("dFRfcRPrincip  servidor >> EL DOM PRINCIPAL ES [%d]", LiDOMPrincipal),
 userlog("dFRfcRPrincip  servidor >> EL RFC SUB ES      [%s]",LuvRelacion->cRfcSubordinado),
 userlog("dFRfcRPrincip  servidor >> NUMERO             [%d]",LuvRelacion->iNoOperacion),
 userlog("dFRfcRPrincip  servidor >> EL DOM ES          [%d]",LuvMovimiento->iDOMOperacion),
 userlog("dFRfcRPrincip  servidor >> LA CLAVE DE MOV ES  [%d]",LuvMovimiento->iCvMovimiento),
 userlog("dFRfcRPrincip  servidor >> LA FECHA ES        [%s]",LuvRelacion->cFeInicio),
 userlog("dFRfcRPrincip  servidor >> EL USUARIO ES      [%s]",LuvMovimiento->cCvUsuario),
 userlog("dFRfcRPrincip  servidor >> Fecha Operacion    [%s]",LuvMovimiento->cFeOperacion),
#endif

 if((LbuffRcv = (FBFR *) tpalloc("FML", NULL, 4096)) == NULL){
   tpfree((char *)LbuffRcv),
          userlog("dFRfcRPrincip  Alojamiento del buffer"),
          return -1,
 }

 /*  Inicio de llamados a los servicios  */

                              *          RFC PRINCIPAL  INSERTA EN CLASCONT     *
                             /*  ruteando al nuevo DOM                 */
                             /*  replicar a CENTRAL                     *

                             LuvClasCont iCvTpclasificacion = iCV_MOV_SOCIEDAD_MERCANTIL,
                             LuvClasCont iCvClascont = iFUSIONADA,
      strcpy(LuvClasCont cSituacion  SITUACION_MARCADO),  /*M*

                             /*voy a definir columnas *
                             GrlTuxNewBlk(LbuffRcv, "EMPRESASESC"),
                             GrlTuxSetCols(LbuffRcv "EMPRESASESC",4)
                             GrlTuxSetRows(LbuffRcv,"EMPRESASESC",0),


#ifdef __DEBUG
 userlog("dFRfcRPrincip  los datos para NvoCcese principal son "),
 userlog("dFRfcRPrincip  LuvClasCont iCvTpclasificacion %i",LuvClasCont iCvTpclasificacion ),
 userlog("dFRfcRPrincip  LuvClasCont iCvClascont %i",LuvClasCont iCvClascont ),
 userlog("dFRfcRPrincip  LuvRelacion->cRfcSubordinado %s" LuvRelacion->cRfcPrincipal ),
 userlog("dFRfcRPrincip  LuvClasCont cSituacion %s",LuvClasCont cSituacion)
 userlog("dFRfcRPrincip  LuvRelacion->cFeInicio %s",LuvRelacion->cFeInicio),
#endif

                        *
      * definir datos
      *
                             GrlTuxDato(&LuvClasCont iCvClascont,      'n', 0),
                             GrlTuxDato(&LuvClasCont iCvTpclasificacion, 'n', 1);
```

```
                                GrlTuxDato(LuvRelacion->cRfcPrincipal,      's', 2),
                                GrlTuxDato(LuvClasCont cSituacion,          's', 3),
                                GrlTuxDato(LuvRelacion->cFeInicio,          's', 4),


                        /*
    * cargar el bufer datos
    */
                                if((GrlTuxAddRowBlk(LbuffRcv,"EMPRESASESC")) < 0)
                                {
                                 tpfree((char *)LbuffRcv),
                                 userlog("dFRfcRPrincip Carga de EMPRESASESC(RFC principal)"),
                                 return -1,
                                }

                                Lrqst data = (char *)LbuffRcv,
                                Lrqst flags = iFUNCION ,
                                LiRet = NvoCCEsc (&Lrqst),
                                if (LiRet == FALSE )
                                {
                                 tpfree((char *)LbuffRcv),
                                 userlog("NvoRPrincip  Llamado a NvoCCEsc (Principal)"),
                                 return -1,
                                }

#ifdef _DEBUG
userlog("dFRfcRPrincip  defino las columnas y renglones para el NvoMovN"),
#endif
                                GrlTuxNewBlk(LbuffRcv, "MOVIMIENTO"),
                                GrlTuxSetCols(LbuffRcv,"MOVIMIENTO",10),
                                GrlTuxSetRows(LbuffRcv,"MOVIMIENTO",0),

                                strcpy(LuvMovimiento->cFeMovimiento,""),
      strcpy(LcFechaOp,""),
      strcpy(LcYear,""),
      strcpy(LcMonth,""),
      strcpy(LcDay,""),
      strcpy(LcHour,""),
      strcpy(LcMin,""),
      strcpy(LcSec,""),

      if (GetAnyDate(LcFechaOp, LcYear, LcMonth, LcDay  LcHour, LcMin, LcSec) < 0){
                                        userlog("dFRfcRPrincip  Error pFechaANSI  Fallo la obtencion de Fecha"),
                                        return 1,
                        }

                                strcat(LcMonth,LcDay),
                                strcat(LcYear,LcMonth),
```

```
                        strcat(LuvMovimiento->cFeMovimiento,LcYear),

#ifdef __DEBUG
userlog("dFRfcRPrincip  LuvMovimiento->cFeMovimiento es -> %s",LuvMovimiento->cFeMovimiento),
#endif

                        LuvMovimiento->iCvMovimiento = iCLASIFICACION_CONTRIBUYENTE,
                        LuvMovimiento->iCvResultado = iOPERADO,
                        LiTipoAtt =ITP_ATENCION_PERSONALIZADA,
                        LuvMovimiento->iFolio = -1,
                        sprintf(LuvMovimiento->cBuzon "%05d", LiDOMPrincipal),
                        strcpy(LuvMovimiento->cOrigen,ORIGEN_PERSONALIZADA ),
                        strcpy(LcFeRecepcion,LuvMovimiento->cFeMovimiento),
                        /*
    * defino los datos del buffer para el siguiente servicio
    */
                        GrlTuxDato(&LuvMovimiento->iCvMovimiento,  'n',0)
                        GrlTuxDato(&LiDOMPrincipal,          'n',1),
                        GrlTuxDato( LuvMovimiento->cFeMovimiento,  's',2),
                        GrlTuxDato( LcFeRecepcion,           's',3),
    GrlTuxDato( LuvMovimiento->cFeOperacion,  's',4),
                        GrlTuxDato(&LuvMovimiento->iCvResultado,  'n',5),
                        GrlTuxDato( LuvMovimiento->cCvUsuario,    's',6),
                        GrlTuxDato(&LuvMovimiento->iFolio,        'n',7),
                        GrlTuxDato( LuvMovimiento->cBuzon,        's',8),
                        GrlTuxDato( LuvMovimiento->cOrigen,       's',9),
                        GrlTuxDato( LuvRelacion->cRfcPrincipal,   's',10),

                        if((GrlTuxAddRowBlk(LbuffRcv,"MOVIMIENTO")) < 0){
            userlog("dFRfcRPrincip  Fallo carga de MOVIMIENTO"),
                         return -1,
                        }
                        GrlTuxNewBlk(LbuffRcv, "CIMDI"),
                        GrlTuxSetCols(LbuffRcv,"CIMDI",0),
                        GrlTuxSetRows(LbuffRcv,"CIMDI",0),

                        GrlTuxDato(&LiTipoAtt,'n',0),

                        if((GrlTuxAddRowBlk(LbuffRcv,"CIMDI")) < 0){
            userlog("dFRfcRPrincip  Fallo carga de CIMDI"),
                         return -1,
                        }

#ifdef __DEBUG
userlog("dFRfcRPrincip  va a mandar a movimiento "),
userlog("dFRfcRPrincip  LuvMovimiento->iCvMovimiento [%d]", LuvMovimiento->iCvMovimiento)
userlog("dFRfcRPrincip  LiDOMPrincipal          [%d]", LiDOMPrincipal ),
```

```
userlog("dFRfcRPrincip  LuvMovimiento->cFeMovimiento [%s]", LuvMovimiento->cFeMovimiento),
userlog("dFRfcRPrincip  LcFeRecepcion          [%s]", LcFeRecepcion),
userlog("dFRfcRPrincip  LuvMovimiento->cFeOperacion  [%s]", LuvMovimiento->cFeOperacion),
userlog("dFRfcRPrincip  LuvMovimiento->iCvResultado  [%d]", LuvMovimiento->iCvResultado),
userlog("dFRfcRPrincip  LuvMovimiento->cCvUsuario    [%s]", LuvMovimiento->cCvUsuario),
userlog("dFRfcRPrincip  LuvMovimiento->iFolio        [%d]", LuvMovimiento->iFolio),
userlog("dFRfcRPrincip  LuvMovimiento->cBuzon        [%s]", LuvMovimiento->cBuzon),
userlog("dFRfcRPrincip  LuvMovimiento->cOrigen       [%s]", LuvMovimiento->cOrigen),
userlog("dFRfcRPrincip  LuvRelacion->cRfcPrincipal   [%s]", LuvRelacion->cRfcPrincipal),
userlog("dFRfcRPrincip  LiTipoAtt                [%d]", LiTipoAtt),
#endif

                                LiRet=dFRfcRMovN(LbuffRcv),
                                if (LiRet == FALSE )
                                {
                                 tpfree((char *)LbuffRcv),
                                 userlog("dFRfcRPrincip  Llamado a NvoRMovN (Principal)"),
                                 return -1,
                                }

    GrlTuxDato(&LiNo_OperacionDos,    'n',0),
    GrlTuxDato(&LiFolioDos.           'n',1),
    /*
     * Obtener el bloque RESPUESTA, esto es, bajar los datos del buffer
     *,
    if (GrlTuxGetRowBlk(LbuffRcv,"RESPUESTA",0)<0)
    {
      tpfree((char *)LbuffRcv)
      userlog("dFRfcRPrincip  Descarga RESPUESTA"),
                                return -1,
    }




    /*
     * RFC PRINCIPAL INSERTA EN HIST_CLASCONT
     *
                                *
     * ntcando al nuevo DOM
     *


#ifdef __DEBUG
userlog("dFRfcRPrincip  los datos para hist_Clascont principal son ")
userlog("dFRfcRPrincip  LuvRelacion->cRfcPrincipal     %s",LuvRelacion->cRfcPrincipal ),
userlog("dFRfcRPrincip  LLuvClasCont iCvClascont       %d",LuvClasCont iCvClascont ),
userlog("dFRfcRPrincip  LiNo_OperacionDos      %d",LiNo_OperacionDos),
userlog("dFRfcRPrincip  LuvMovimiento->iDOMOperacion   %d",LuvMovimiento->iDOMOperacion ),
```

```
userlog("dFRfcRPrincip  LuvRelacion->cFelnicio        %s",LuvRelacion->cFelnicio),
#endif

                                GrlTuxNewBlk(LbuffRcv, "HISTORICO"),
                                GrlTuxSetCols(LbuffRcv,"HISTORICO",4),
                                GrlTuxSetRows(LbuffRcv,"HISTORICO",0),


#ifdef __DEBUG
     userlog("dFRfcRPrincip  Se creo el BLK HISTORICO"),
#endif

                                GrlTuxDato( LuvRelacion->cRfcPrincipal,   's',0),
                                GrlTuxDato(&LuvClasCont.cCvClascont,      'n',1),
                                GrlTuxDato(&LiNo_OperacionDos,            'n',2),
                                GrlTuxDato(&LiDOMPrincipal,               'n',3),
                                GrlTuxDato( LuvRelacion->cFelnicio,       's',4),

                                if((GrlTuxAddRowBlk(LbuffRcv,"HISTORICO")) < 0)
                                {
                                  tpfree((char *)LbuffRcv),
                  userlog("dFRfcRPrincip  Carga de HISTORICO"),
                                  return -1,
                                }
#ifdef __DEBUG
     userlog("dFRfcRPrincip  Se cargaron los datos a HISTORICO"),
#endif

                                Lrqst data = (char *)LbuffRcv,
                                Lrqst flags = tFUNCION
                                LiRet = NvoHisClasCont(&Lrqst ),

                                if (LiRet == FALSE )
                                {
#ifdef __DEBUG
       userlog("dFRfcRPrincip  fallo en histclascont principal"),
#endif
       tpfree((char *)LbuffRcv),
       userlog("dFRfcRPrincip  Llamado a NvoHistClasCont "),
                        }


  tpfree((char *)LbuffRcv),
  return 0,
}
```

Figura 4 6 Código de Servicio Tuxedo

4.7 Participación Específica en el Proyecto

En este proyecto participé como miembro de Softtek en la parte del desarrollo Tuxedo en el Módulo de Registro, en la siguiente figura se ubica mi rol y mis tareas en el contexto de desarrollo

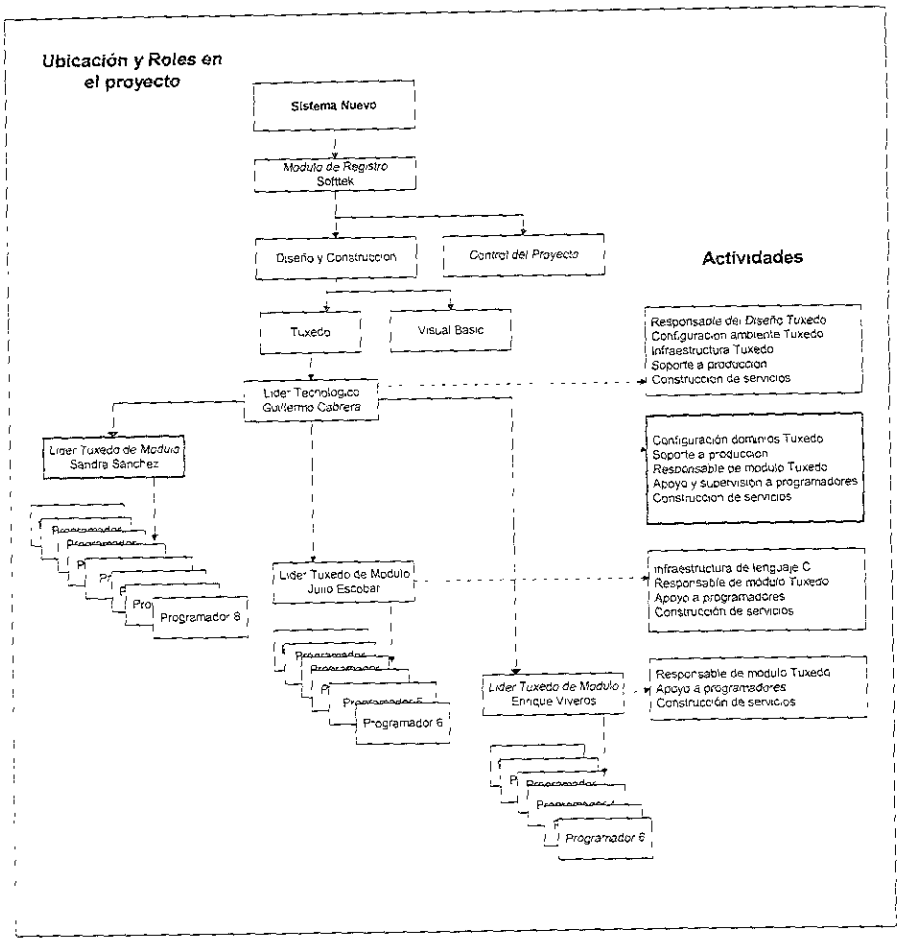


Figura 5 7 Contexto de mi rol desarrollo del Sistema Nuevo

La descripción de cada una de las tareas listadas en el diagrama es la siguiente:

- **Configuración de Dominios Tuxedo**

  Se ha señalado en el trabajo, capítulos 3 y 4, que los dos archivos que sostienen la configuración Tuxedo son el *ubbconfig* y el *dmconfig*, el segundo archivo es el que hace posible que la aplicación local pueda acceder a servicios remotos y de la misma manera permite ofrecer servicios a otras aplicaciones. Yo tuve a mi cargo el hacer las pruebas de concepto de comunicación entre dominios y configurar para pruebas el archivo dmconfig.

  Es importante mencionar que no existía ningun archivo de este tipo desarrollado y que no teníamos experiencia en la configuración, por lo que el que pudiera implementarlo tuvo gran valor para el proyecto

- **Asesoría y Responsable de Módulo de Construcción Tuxedo**

  Ya que Tuxedo no contaba con gran oferta de profesionistas expertos o con experiencia en proyectos de este tipo y con esta herramienta (1997-1998), no contábamos con suficientes programadores Tuxedo, por lo que se optó por asignar a programadores de lenguaje C formado módulos de construcción y asignar a un responsable que si conociera el API Tuxedo. De esta manera la estrategia de desarrollo fue que yo, como líder de módulo de construcción Tuxedo guiara a los programadores en cuanto al API Tuxedo, resolver dudas sobre el paradigma Cliente/Servidor y lenguaje C y revisar los productos en cuanto a programación. Por último era responsable de la productivad y calidad de los productos entregados por mi módulo.

  Cabe mencionar que en las estadísticas de control del proyecto fuimos el equipo que entrego mayor numero de servicios.

- **Construcción de Servicios Tuxedo**

  Como parte de mis actividades también generé archivos generales para la programación como el de constantes, estructuras de datos y construcción de servicios Tuxedo con pruebas unitarias e integrales

  Al inicio del proyecto fue poca mi participación el la programación porque la mayor parte del tiempo estaba resolviendo dudas o revisando programas de hasta 8 programadores. Se acentuó mi participación en la programación cuando fui asignada tiempo después a generar nuevos requerimientos aproximadamente 8 meses después

- **Apoyo a Producción de la Aplicación Tuxedo**

En esta etapa estuve asignada a en enseñar los procedimientos basicos de la administración de la aplicación Tuxedo al departamento de "Apoyo a la Producción del 'Sistema Nuevo' " de SHCP En esta etapa participamos 3 de los líderes de módulo Entre mis funciones estuvieron las siguientes

a) Entrenamiento en la administración de la aplicación a los responsables de ésta por parte del cliente (Departamento de Sistemas de la SHCP).

b) Apoyo a los responsables de la aplicación del Sistema Nuevo en las Oficinas Locales vía telefónica y en algunos de los casos vía un telnet a su servidor

c) Análisis y reasignación de problemas a los responsables específicos de éste, por ejemplo podían surgir problemas propios de la aplicación Tuxedo· direcciones incorrectas, procesos mal levantados, aplicaciones que no podían levantar (por mencionar algunos). este tipo de problemas eran los que nosotros resolvíamos; existían otro tipo de problemas y teníamos que identificarlos y reasignarlos, por ejemplo· falta de recursos en el sistema operativo, incongruencia de datos, etc.

Esta etapa fue tan enriquecedora como estresante para mi ya que fue la primera vez que estuve en apoyo a producción y el hacer frente a problemas y encontrar la solución optima lo más rapido posible, involucró a un equipo de varios especialistas los cuales teníamos que confiar en las decisiones de cada uno de nosotros y hacernos responsables de las nuestras. esto en un ambiente de gran presión Esta actividad me dejó la gran experiencia de trabajar bajo presión

CONCLUSIONES

Participar en el desarrollo de una aplicación de estas dimensiones representa una experiencia enormemente valiosa tanto en el sentido profesional, como en lo personal La magnitud del desarrollo aportó gran conocimiento en distintas áreas desde la parte de administracion de requerimientos, de recursos físicos, lógicos, así como la experiencia para resolver diversos problemas de forma eficaz y eficiente mediante el uso correcto de la tecnologia

Ha sido muy satisfactorio saber que el resultado obtenido cubrió en su totalidad las necesidades del cliente, logrando un sistema de alto rendimiento, capaz de administrar una aplicación distribuida de misión crítica, el cual resuelve múltiples operaciones concurrentes mediante aplicaciones aparentemente sencillas, que generan transacciones globales en línea entre varios dominios de forma transparente con buen desempeño.

La propuesta de solución presentada no sólo pretendió cubrir los requerimientos y necesidades de los usuarios operativos, analistas y autoridades, sino aportar una solución global con una visión evolutiva a largo plazo, facilitando el mantenimiento que permita responder a los cambios permanentes en el entorno. El fácil mantenimiento al código quedó demostrado dada la rapidez en la que se agregaron servicios Tuxedo independientes a los ya construidos y que fueron requeridos por cambios en las reglas del negocio derivados de modificaciones en la Leyes que impactaron a la SHCP

La tecnología utilizada fue un factor clave para el éxito de la aplicación y para lograr una solución robusta, segura, administrable, de fácil manejo, que cuente con información verdaderamente integral y optimizando los recursos

La implementación de una arquitectura Cliente/Servidor a tres capas, aún ha sido poco explotada en el mercado y no existen muchos desarrollos con estas características, donde se experimentan conceptos teóricos, de los cuales no muchos han sido puestos en práctica, este proyecto permitió llevar a aplicar estos conceptos, separando la aplicacion en componentes y distribuirlos en múltiples procesadores

Al inicio del proyecto existieron problemas que requirieron corregir productos funcionales, de programación y de configuración, lo que impone prestar especial atención en la parte de conceptuar de forma muy clara la problematica y los requerimientos así como la correcta definición de la arquitectura de solución Es un hecho

que la experiencia en más proyectos de este tipo disminuirán los errores cometidos al inicio del proyecto evitando el retrabajo que se tuvo que hacer en etapas de pruebas

Dentro de la arquitectura de 3 capas el middleware, TUXEDO, fue una de las partes más importantes, o quizá la más importante, ya que en esta se encuentra todo el control de las reglas del negocio. toda la administracion de la aplicación que permite: el balacear cargas, restablecer servicios, monitorear servidores y clientes para optimizar el desempeño y sí fue capaz de realizar transacciones involucrando diferentes bases de datos en diferentes dominios de la aplicación utilizando two-phase-commit con lo que los datos son congruentes en las bases de datos involucradas .

El participar en la administración, diseño, implementación TUXEDO abrió la oportunidad de utilizar de forma correcta la disciplina transaccional en diversas bases de datos así como múltiples dominios, ya que TUXEDO efectivamente es capaz de coordinar transacciones globales entre dominios y bases de datos distribuidas mediante two phase commit. TUXEDO también ha brindado mecanismos de administración pudiendo optimizar la aplicación poco a poco, definiendo réplicas de los servicios más utilizados, el restablecimiento de servidores automáticamente cuando tienen algún problema no previsto TUXEDO cumplió con lo requerido en cuanto a herramienta de administración y construcción para la Aplicación El participar en este proyecto me ha dejado una enorme experiencia en cuanto a la Arquitectura Cliente/Servidor 3 capas, específicamente en el middleware con TUXEDO, lo que me dio la experiencia de poder participar posteriormente como líder tecnológico en otros proyectos ya que conozco la arquitectura, lo que implica un buen o mal diseño de la solución y todas las bondades que puede o no puede brindar la herramienta También obtuve la experiencia de coordinar equipos de trabajo y la capacidad de resolver problemas en etapas de liberación y producción sin caer en la desesperación. Estas nuevas habilidades han contribuido enormemente a mi crecimiento profesional

En el desarrollo de este proyecto las personas de MAC tuvimos una participación muy importante. Por una parte. el equipo de desarrollo TUXEDO estaba formado por un líder tecnológico, tres líderes de módulo TUXEDO y 20 programadores, el líder es también egresado de la carrera de MAC y yo fui líder de Módulo TUXEDO Por otra parte en el equipo de análisis también participó una egresada de MAC. Nuestra participación en el proyecto dio también el reconocimiento a nuestra carrera dado nuestro muy buen desempeño

Por otro lado, el que este trabajo de titulación sea de interés para algunos compañeros de trabajo y amigos como fuente bibliográfica, ya que empiezan a trabajar con este tipo de arquitectura y realmente van a utilizarlo para fines laborales, es un interés que no esperaba al empezar el trabajo y esto me genera gran satisfacción laboral y personal

Acerca de nuestra comunidad universitaria, quiero resaltar que la mayoría de los estudiantes hemos desperdiciado la gran oportunidad de acercarnos a la tecnología de punta, conocerla y manejarla ya que en la UNAM contamos con Centros de Investigación que están a la cabeza de las Universidades del país en cuanto a tecnología. Por este motivo invito a mis compañeros de carrera a involucrarse en las actividades en donde estén utilizando la nueva tecnología y puede ser en las actividades que la UNAM organiza y cuya promocion publica la jefatura de MAC · convocatorias a Cursos, Planes de Becarios, servicio social en Institutos de Investigación y Servicio Social en Proyectos de Sistemas en empresas privadas para abrir puertas mas alla del salon de clases Cabe mencionar que el líder tecnológico y yo participamos anteriormente como becarios en la Centros de Investigación de la UNAM

Finalmente agradezco a Nuestra Máxima Casa de Estudios, la Universidad Nacional Autónoma de Mexico, la oportunidad que me brindó para poder participar en el desarrollo de este país como profesionista

INTRODUCCION

El desarrollo de la tecnología y la reducción de costos en ella, ha permitido que algunas empresas inviertan sumas importantes de dinero en productos tecnologicos poderosos que le permitan optimizar su operación y asi aumentar su productividad

Uno de estos productos es TUXEDO, un "middleware o monitor de transacciones de misión critica", estas definiciones significan, entre otras cosas, que es capaz de monitorear administrar y construir aplicaciones cuya operación es constante, esto es, que permite atender requisitos de millones de clientes en cualquier momento y al mismo tiempo (concurrencia) reduciendo conexiones a la base de datos Puede monitorear transacciones entre varias bases de datos distribuidas a nivel país y a nivel mundial, este monitoreo asegura que la integridad de datos sea correcta aunque las bases de datos sean diferentes, Informix con Oracle, por ejemplo, y estén en diferentes servidores sobre diferentes sistemas operativos y en diferentes localidades

Para entender la importancia de TUXEDO como un producto de solución poderoso, se describirá su implementación en un proyecto de impacto a nivel Nacional el Registro Federal de Contribuyentes, desarrollado para la Secretaría de Hacienda y Crédito Público (SHCP)

En el primer capitulo se describe la situación general de la SHCP antes de desarrollar el proyecto, así como el requerimiento general del "Sistema Nuevo" Se describen las características y problemática de la operación del negocio y a partir de esta informacion se describe la propuesta global de solución Es importante conocer la operación de SHCP porque a partir de estas características se justifica la utilización del producto TUXEDO para la solución

En el segundo capítulo se describe la arquitectura base de TUXEDO, esto es, qué lugar ocupa en la arquitectura Cliente/Servidor de 3 Capas y cuáles son las características de este tipo de arquitectura Se describirán tambien las diferencias principales entre arquitectura 2 capas y arquitectura 3 capas Por último se describe los estándares, productos y lineamientos para formar la infraestructura sobre la cual se desarrollo el proyecto tales como ambiente de desarrollo, servicios de red, sistema operativo, equipos y bases de datos De esta manera se asocia la problemática del SHCP con las caracteristicas de la Arquitectura Cliente/Servidor 3 Capas y se definen los productos seleccionados para la solucion del problema

En el tercer capítulo se detallarán las características de TUXEDO como producto de solución específica para aplicaciones de misión crítica en ambientes distribuidos Se expondrá la Historia del Monitor de Transacciones TUXEDO, también se describirá como surge y cómo se ha desarrollado. Se dará la definición de TUXEDO como middleware, sus características, su objetivo, su lugar en la arquitectura 3 capas También se describiran las utilerías que conforman TUXEDO

En el último capítulo se expone el diseño general de la aplicación, únicamente en lo que respecta a TUXEDO (segunda capa en la arquitectura de 3 capas) configuración TUXEDO, dominios TUXEDO y programas TUXEDO principalmente Se expondrá la solucion específica de TUXEDO
Tambien se describira mi participación en el proyecto como parte de la Consultoría en Sistemas Softtek

Por último quiero mencionar que este trabajo tiene varios propósitos el primero fue documentarme y entender el lugar que ocupa TUXEDO dentro de la tecnología y así tratar de plasmarlo en este trabajo Este trabajo también puede ser de utilidad a personas cuyo desarrollo profesional se ha desarrollado en otras Arquitecturas diferentes a Cliente/Servidor y quieran entender el paradigma de esta arquitectura en 2 y 3 capas También tiene el fin de servir como apoyo bibliográfico a la comunidad universitaria

Cabe mencionar que compañeros de trabajo y amigos de MAC, quienes se han enterado sobre este trabajo de titulacion, me han pedido que al terminarlo se los preste por los siguientes motivos Personas que lo quieren como apoyo bibliográfico ya que piensan migrar aplicaciones Cliente/Servidor de 2 a 3 capas Algunos otros están interesados en entender el paradigma Cliente/Servidor ya que su desarrollo profesional ha sido en otras arquitecturas diferentes a esta y están en transicion de cambio Otros compañeros empiezan su desarrollo como programadores TUXEDO por lo que muestran mayor interés en él Por ultimo y mas importante el principal propósito de este trabajo es el finalizar formalmente esta etapa en mi educación profesional con el título de la Licenciatura en Matemáticas Aplicadas y Computación

GLOSARIO

A

Administrador Tuxedo

Es la persona que instala el sistema Tuxedo, configura y monitorea la aplicacion Tuxedo y actualiza la informacion de la aplicación (por ejemplo nombres y localidades de computadoras)

Administrador de Recursos - Resource Manager (RM)

Una interfase y un software asociado que brinda acceso a una colección de datos y los procesa Un ejemplo es un sistema manejador de base de datos, Informix Los administradores de recursos tienen la característica de manejo de transacciones , éstos son las entidades que se acceden y controlan dentro de una transaccion global

Administrador de Transacciones -Transaction manager (TM)

Es un componente de software que administra una transacción global en la que participan varios programas de aplicación Un administrador de transacciones coordina comandos de los programas de la aplicación junto con los administradores de recursos para empezar y completar una transacción global por medio de la comunicacion con todos los administradores de recursos que están participando dentro de esta transacción Cuando un administrador de recursos falla durante la transacción global, el administrador de transacciones ayuda a los manejadores de recursos ya que les indica hacer el cambio (commit) o deshacer el cambio aparente (rollback) y a que estaba pendiente la confirmacion de este cambio

ALR

Administración Local de Recaudación

Aplicación

Se define como aplicacion de uno o mas dominios Tuxedo cooperando para apoyar una funcionalidad específica del negocio Cada dominio está conformado por servidores, servicios y un o más manejadores de recursos definidos en el archivo de configuración (TUXCONFIG)

Arquitectura

Es una plataforma de hardware y software por ejemplo La arquitectura del Dominio de una Oficina Local, se define a grandes rasgos por Equipo HP, Sistema Operativo UX10 20, con Software Tuxedo 6 3 como Monitor de Transacciones así como un equipo WINDOWS 95, y Tuxedo 6 3, esta es la arquitectura definida para los nodos locales únicamente para la parte del Middleware 2) La estructura e interrelacion de componentes en un sistema o en un ambiente

B

Bulletin Board

Es una colección de estructuras de datos compartidas que están diseñadas para monitorear una aplicación Tuxedo mientras esta corriendo. Contiene información de los servidores, servicios, cliente y transacciones que conforman la aplicación Tuxedo El bulletin Board es replicado en cada máquina lógica en la aplicación

Bulletin Board Liaison (BBL)

Es un proceso administrativo de Tuxedo responsable de mantener una copia del bulletin board de cada procesador particular El proceso BBL corre continuamente en cada maquina lógica de la aplicación mientras la aplicación esté levantada Este proceso administra los BB's que conforman una aplicación distribuida

C

Cliente

Es un programa que invoca operaciones llamando a los servidores Tuxedo Los clientes remotos son iguales a los clientes nativos Sus peticiones son manejadas diferente pero transparentemente

Cliente Robusto

En este trabajo se refiere a que la mayoría de las reglas del negocio del sistema se han implementado en el procesador cliente

CPN

Centro de Procesamiento Nacional, condensa toda la información de los contribuyentes y se encuentra en la Ciudad de Mexico

Concurrencia

Es la ejecución simultánea de más de un proceso o función Los procesos concurrentes pueden utilizar recursos comunes compartidos alternando su tiempo de uso

Commit

El commit significa que los cambios a recursos compartidos, como puede ser el caso de una base de datos, tienen un efecto permanente

D

Dominio Tuxedo (T/Domain)

Es una colección de sistemas Tuxedo que estan administrados independientemente Los componentes de Dominios Tuxedo permiten llamadas de servicios entre dominios permitiendo

así ejecutar transacciones El administrador del sistema usa el archivo UBBCONFIG así como el archivo BDMCONFIG para configurar el dominio Tuxedo 2) Es un conjunto de servidores, servicios y resource manager (administrador de recursos), definidos en un solo archivo de configuración UBBCONFIG

F

Front end

Es común utilizar las palabras del idioma Ingles ***front end*** para referise al softwere cliente, por ejemplo, Visual Basic que es la interfaz que ve el usuario final

I

Interfaz de programación de aplicaciones - Application Programming Interface (API)

1) Elementos y ambiente que existe a nivel aplicación para apoyar un producto particular de software de sistema 2) Un conjunto de código que permite al programador iniciar y terminar peticiones cliente/servidor dentro de una aplicación 3) Un conjunto de llamadas que define como invocar a un servicio Un conjunto de interfaces de programación bien definidos (entradas, parámetros requeridos para llamar alguna función y valores de retorno de éstas) por medio de las cuales un programa utiliza el servicio de otro programa

Interfaz de programación para Monitor de Transacciones - Application to Transaction Monitor Interface (ATMI)

Es la Interfaz de programación para el sistema Tuxedo, que incluye rutinas de transacción, rutinas de manejo de mensajes, rutinas de interfaces de servicios y rutinas de manejo de buffer

M

Middleware

1)El "middleware" divide a los programas aplicativos en componentes clientes y servidores, la función del middleware consiste en efectivamente distribuir el proceso en múltiples servidores, administrando las transacciones distribuidas, y la integración de múltiples bases de datos

2)El middleware tiene como funcion satisfacer las necesidades de desarrollos en red para crear, integrar y manejar aplicaciones de gran escala en ambientes heterogéneos y distribuidos

3) Es un conjunto de servicios para construir aplicaciones Cliente/Servidor distribuidas, como son servicios para localizar otros programas en la red establecer comunicación entre esos programas y pasar información entre aplicaciones Los servicios que brinda el middleware resuelve la comunicación entre diferentes plataformas y brinda un modelo uniforme de autorización que cumplen múltiples proveedores de software y múltiples sistemas operativos

Misión-crítica

Se refiere a sistemas los cuales son bombardeados por cientos o miles de peticiones y de una manera concurrente (varias peticiones al mismo tiempo), tienen por característica principal su

alta disponibilidad, esto es, que son capaces de atender todas las peticiones con un resultado eficiente y rápido

Monitor de Transacciones - Transaction Processing Monitor (TP-Monitor)

Es un middleware con la capacidad adicional de poder manejar transaccionalidad en ambiente distribuidos y heterogeneos

P

Plataforma

Es un término de carácter genérico que designa normalmente una arquitectura de hardware, aunque también se usa a veces para sistemas operativos o para el conjunto de ambos Los ordenadores VAX de la firma Digital, por ejemplo, serían una plataforma en la que se pueden soportar aplicaciones que, a su vez, corren en otras plataformas

Proceso

1)Un proceso es una entidad activa que tiene asociada un conjunto de atributos código, datos, *stack*, registros e identificador único 2)Un proceso es la entidad de ejecucion reconocida por el sistema de operación 3)Un proceso es un programa en ejecución Esta ejecución es secuencial 4) Un proceso requiere de recursos (memoria, CPU, dispositivos de Entradas/Salida, stack) para su ejecución Los procesos tambien son llamados frecuentemente *tareas* Un programa, por su parte representa una entidad pasiva Cuando un programa es reconocido por el sistema de operación y tiene asignado recursos, éste se convierte en proceso

Programación Cliente/Servidor

Es un modelo de programación en el cual los programas de la aplicación estan estructurados como clients o servidores para generar un procesos distribuido Un cliente es programa de la aplicación que requiere la ejecución de servicios Un programa servidor es una entidad que ejecuta rutinas de servicios para responder a los programas clientes Un servicio es un modulo de programa de la aplicación que ejecuta una o mas funciones específicas 2) La programación Cliente/Servidor puede ser configurada en una estructura de 2 ó 3 capas Una configuracion de 2 capas consiste de únicamente el cliente y el servidor Una configuracion de 3 capas incluye un cliente un servidor y un nivel intermedio (middleware)

R

Red

1) In grupo de nodos interconectados 2) La ruta de comunicación para comunicarse con un servidor

Reglas del Negocio

Reglas del Negocio

Toda aplicacion trata de reflejar parte del funcionamiento del mundo real, para automatizar tareas que de otro modo serían llevadas a cabo de modo más ineficiente, o bien no podrían realizarse Para ello, es necesario que cada aplicación refleje las restricciones que existen en el negocio dado, de modo que nunca sea posible llevar a cabo acciones no válidas A las reglas que debe seguir la aplicacion para garantizar esto se las llama *reglas de negocio*, o *business rules* Ejemplos de tales reglas son no permitir crear facturas pertenecientes a clientes inexistentes, controlar que el saldo negativo de un cliente nunca sobrepase cierta cantidad, etc

En realidad, la información puede ser manipulada por muchos programas distintos así, una empresa puede tener un departamento de contabilidad que controle todo lo relacionado con compras, cobros, etc , y otro departamento técnico, que esté interesado en relacionar diversos parámetros de produccion con los costos La visión que ambos departamentos tendran de la información y sus necesidades serán distintas, pero en cualquier caso siempre se deberan respetar las reglas de negocio. El hecho de que la información sea manipulada por diversos programas hace más difícil garantizar que todos respeten las reglas, especialmente si las aplicaciones corren en diversas máquinas, bajo distintos sistemas operativos, y están desarrolladas con distintos lenguajes y herramientas

Rollback

Se refiere a deshacer los cambios hechos a la base de datos en la etapa del precommit dentro de una transacción

S

Servidor

1)Es un programa que ejecuta una tarea requerida por un cliente 2) Un módulo de software que acepta peticiones de clientes o de otros servidores y regresa respuestas 3) Un programa de software que brinda uno o más servicios en una arquitectura Cliente/Servidor

Servidor Robusto

En este trabajo se refiere a que la mayoría de las reglas del negocio del sistema se han implementado en el procesador servidor, esto es, que la mayoria de la logica del negocio está codificada en los servicios

SIR

Sistema Integral de Recaudacion, que maneja la Administracion General de Recaudación de la Secretaria de Hacienda y Crédito Público para recaudar impuestos

Sistema TUXEDO

Es un middleware robusto diseñado por BEA Systems, Inc para el desarrollo y deploying de aplicaciones de misión crítica Cliente/Servidor Este sistema es capaz de manejar transacciones distribuidas, mensajes de la aplicación y cuenta con un complemento de servicios necesarios para la construccion, ejecución y administracion de aplicaciones complejas de misión crítica.

T

Tipos de Buffers

Es un nombre abstracto para un tipo de mensaje Tuxedo cuenta con 4 tipos de mensajes de comunicación FML, VIEW, STRING y CARRAY Estos tipos de buffer son codificados y decodificados transparentemente a través de una red de maquinas heterogéneas Cada aplicación puede definir tipos de buffers adicionales

Transacción Global

1) Una transacción que involucra a uno o más manejadores de recursos para llevar a cabo transacciones locales 2) Una transacción global coordina múltiples servidores o interfases de administradores de recursos dentro de una unidad atómica de trabajo

X

X/OPEN

Estándar para el manejo de transacciones, cuyo principal propósito de este modelo es definir los componentes de sistemas basados en transaccionalidad y proporcionar las interfaces entre ellos X/open permite a los manejadores de recursos estar asociados con una transacción global cuando la aplicacion los llama directamente

Bibliografía

The Tuxedo System
Juan M. Andrade, Mark Carges, Stephen Felts
Addison Wesley, 3ra edición
USA 1998

3-Tier Client/Server At Work
Jery Edwards with Deborah Devoe
John Wiley & Sons Inc
USA, 1997

Introduction to Client/Server Systems
Second Edition
Paul E. Renaud
John Wiley & Sons Inc
USA, 1996

Technical Foundations of Client/Server System
Carl L. Hall,
John Wiley & Sons Inc
USA. 1994

BEA TUXEDO 6.5
Aplication Administration Course
TUX-103-65-01
BEA Systems, Inc Educational Services
Febrary, 1999

BEA TUXEDO 6.5
Domains Administration Course
DOM-113N-65-01
BEA Systems. Inc Educational Services
May, 1999

Dirección en internet // beasys com