UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"EXPERIENCIA PROFESIONAL EN EL DESARROLLO, SOPORTE E IMPLEMENTACIÓN DE SISTEMAS DE SOFTWARE"

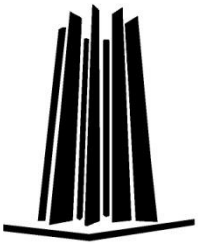
INFORME DEL EJERCICIO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

JUAN CARLOS MURO CORDERO

ASESOR:
ING. OSCAR ESTRADA GARCÍA



FES Aragón

MÉXICO 2014

Agradecimientos:

A mis padres Juan Muro Cedillo y Amparo Cordero Mendoza, por su incondicional apoyo, a mis hermanos Luis Javier y Cesar Mendoza Chávez, mis abuelitas María Mendoza y Fortunata Cedillo.

ÍNDICE

Pág.

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

INTRODUCCIÓN

El siguiente informe del ejercicio profesional resume las actividades que he desempeñado durante mi vida laboral, dentro de las distintas empresas en donde he tenido la oportunidad de participar, contribuyendo con esto a mi desarrollo profesional dentro del ramo de la ingeniería de computación y actividades relacionadas con el desarrollo, soporte e implementación de sistemas que han permitido reducir costos y tiempos, mediante la automatización de procesos aplicando una solución de software. Es importante mencionar que dentro de mi participación en las diferentes empresas, he tenido la oportunidad de crecer profesionalmente ya que en cada compañía de la que he sido parte, me he desempeñado honestamente y siempre asumiendo los retos derivados de las actividades asignadas.

Durante los siguientes capítulos se presenta un resumen de problemas reales que han requerido una solución de software junto con la participación de diferentes equipos de trabajo para llevar a buen término las situaciones que se han presentado, producto de las necesidades de cada compañía. A lo largo de las actividades realizadas se trata de transmitir mi participación en la solución de los problemas así como la forma en que los mismos han sido resueltos mediante el uso de herramientas de desarrollo, el buen análisis, la implementación de arquitecturas, el trabajo en equipo y la coordinación con todos los actores involucrados refiriéndome con esto, a los usuarios de los sistemas, administradores de proyectos, lideres técnicos, otros equipos de desarrollo, etc.

Los proyectos que serán descritos en los siguientes capítulos, fueron realizados con mi participación directa ya sea como consultor externo de sistemas o como empleado regular en las distintas compañías mencionadas realizando una breve descripción de la problemática presentada, para continuar después con un resumen de las actividades que se realizaron, teniendo que para cada asignación se muestra un organigrama del proyecto en el cual se muestra mi participación dentro del mismo (marcado en el cuadro rojo).

Las labores expuestas en este documento, incluyen la creación de piezas de software realizadas para la solución de problemas mediante su uso, desarrollo completo de sistemas utilizando distintas tecnologías, uso de herramientas de software de terceros que ayudan a implementar las soluciones, uso de productos de software robustos dedicados para actividades particulares como migración de datos, diseño de interfaces gráficas, codificación y compilación de lenguajes de programación.

Capítulo 1
Consultora de Sistemas Tornado Solutions.

Giro. Consultoría en Sistemas.
Posición. Consultor Jr.
Dirección**:** Bosque de Ciruelos No. 160 piso 6 B Col. Bosques de las Lomas México Distrito Federal

1.1 Proyecto Asignado PROFECO (Procuraduría General del Consumidor)

Giro**.** Empresa Gubernamental dedicada al apoyo y protección del consumidor frente a prácticas de carácter ilegal por parte de empresas privadas.
Posición Ocupada: Desarrollador CRM Oracle Jr.
Líder Técnico: Juan Carlos Azofeifa
Project Manager: Michel Richmond
Personas a mi cargo: Ninguna.

Organigrama del Proyecto:

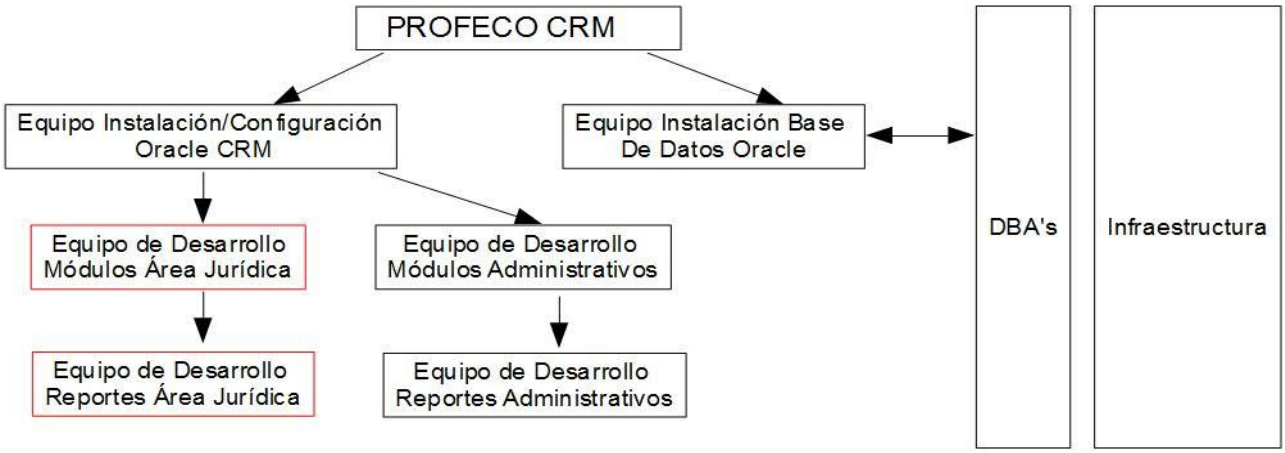


Fig. 1.1 Organigrama del Proyecto

1.1.1 Introducción General del Proyecto.

Durante este proyecto, se realizó la instalación de la suite de Oracle CRM (Customer Relationship Management) para estandarizar los procesos internos de la dependencia y que anteriormente se manejaban en forma dispersa en sus distintas áreas, debido a que no se contaba con una solución de software adecuada, causando un alto consumo de tiempo en las actividades diarias de los usuarios además de altos riesgos como la pérdida parcial/total de información.

Este proyecto se llevó a cabo con distintos equipos de trabajo cada uno con tareas específicas asignadas, por ejemplo, algunos equipos únicamente realizaron la instalación y configuración de la suite de la herramienta Oracle CRM, a otros les fue asignada la tarea de rastrear toda la información de las distintas áreas para la unificación de la misma en una sola base de datos única derivando en la creación de los diagramas entidad relación, que más tarde se convertirían en tablas de base de datos donde se persistirá toda la información relevante a los procesos de negocios de la dependencia así como la información de referencia necesaria de usuarios y catálogos necesarios, en este caso, el motor de base de datos de Oracle.
Entre las principales actividades asignadas a mi cargo, fue el de realizar juntas periódicas con los usuarios pertenecientes al área **Jurídica** dentro de la dependencia para recabar sus necesidades de software, manteniendo una retro-alimentación constante en cuanto a entender el negocio que se modelaría dentro del CRM y los requerimientos de manera más específica a desarrollar que servirían finalmente en sus labores diarias.
Una vez completado el proceso de levantamiento de información con los usuarios, se procedió a levantar la documentación de requerimientos utilizando principalmente casos de uso, los cuales terminarían con el visto bueno de directivos, jefes de área, subordinados y el equipo de desarrollo involucrado.
Es importante hacer mención que durante toda la fase de construcción del proyecto, solo 2 ambientes fueron habilitados: desarrollo y producción, esto afectó un poco el proceso de calidad de software, debido a que el no contar con ambientes de QA (Quality & Assurance) o UAT (User Acceptance Testing), no se detectaban errores o bugs hasta que eran reportados por el usuario dentro del ambiente de producción, generando retrasos en otros entregables debido a que se tenía que hacer la corrección inmediata de los problemas reportados.

1.1.1.1 Descripción de los Problemas Encontrados

A continuación, se enumeran algunos de los principales problemas detectados antes de la instalación y puesta en marcha de ésta herramienta de software (CRM) adquirida por la dependencia.

Descripción del problema.- No mantener la información en una base de datos única.
Al no tener una base de datos única para registrar las actividades diarias, como por ejemplo, de la demandas recibidas (u otras actividades), la información se registraba y guardaba en diferentes lugares como: Archivos planos, bases de datos SQL Server/Access, bases de datos Oracle, archivos de hojas de cálculo, documentos tipo Word, etc. Lo anterior, resulta poco confiable desde el punto de vista operativo, además de que la información dispersa en varios lugares causa demasiados problemas, solo por enumerar algunos de los que a los usuarios más preocupaban, era que la información guardada en las computadoras personales o en un servidor compartido, frecuentemente sucedía que si en las maquinas se presentaba un problema en el hardware o software (disco duro fallaba, algún virus, sistema operativo colapsado, etc.) entonces, se perdía completamente la información, teniendo que en la mayoría de los casos, no se contaba con ningún tipo de respaldo, en otras ocasiones con suerte, se lograba recuperar parcialmente la información teniendo que volver a repetir el trabajo para completar la información faltante. Lo anterior, dejaba una enorme inconsistencia en la información así como retrasos a los usuarios, sumándole a esto, la poca seguridad que se tenía con información sensible que se encontraba a la vista de cualquiera.

Otro problema fue, que en algunas ocasiones, los usuarios salían de vacaciones dejando sus equipos apagados, si algún otro usuario requería consultar alguna información en este equipo, simplemente era imposible, debido a que no contaba con su contraseña de seguridad, por lo que se tenía que esperar a que el usuario encargado del equipo regresará de vacaciones o con suerte, se le llamaba durante sus vacaciones para pedirle su contraseña y ubicación de los archivos requeridos, siendo esto una mala práctica desde el punto de vista de seguridad de la información.

Descripción del Problema.- No contar con una solución única de REPORTES
Al no haber una solución única para los reportes dentro de la dependencia, la impresión de formatos, reportes relativos al negocio u otra documentación diversa, ocasionaba que cada área tenía sus propias formas de resolver el problema, por ejemplo, usando plantillas en hojas de cálculo definidas por esa área o utilizando aplicaciones que imprimían directamente la información a la impresora creadas por terceras partes, pero sin ningún tipo de soporte para nuevas necesidades resultando en aplicaciones obsoletas, en otras áreas se utilizaban archivos tipo Word que se compartían entre todos los usuarios y áreas a través de un servidor compartido sin ningún tipo de seguridad, ya que generalmente, todos los usuarios podían acceder sin ninguna restricción y peor aún, con permisos para borrar los archivos compartidos.
Lo anterior, resultaba en un caos total, debido a que si alguna área necesitaba modificar alguna plantilla para agregar información adicional a algún reporte, por ejemplo, imprimir una fecha en formato “MM/DD/YYYY”, entonces actualizaba la plantillas dentro del servidor compartido, lo anterior sin avisar a otras áreas, resultando con esto, que cuando otra área utilizaba el mismo reporte, se encontraba con nueva información o formatos, pensando entonces, que los reportes eran erróneos o peor aún, ellos no necesitaban mostrar nueva información y solicitaban que las viejas plantillas siguieran vigentes.
Era necesario unificar en una solución común todos los reportes de la dependencia que sirviera de manera general a todos los usuarios, por lo que fue adquirida la herramienta Oracle Discoverer Reports, la cual permite dar una solución de reportes dentro de una arquitectura Cliente/Servidor, en donde, los reportes quedan publicados dentro de un servidor y los usuarios con los apropiados permisos, pueden ejecutarlos desde su computadora, únicamente instalando el programa cliente que viene con esta herramienta.

1.1.2 Resumen de las Actividades Realizadas.

A continuación, se detallan las actividades realizadas durante el proyecto, en las cuales mantuve participación directa como:

- Modelado del proceso de Ingreso de Demandas al Sistema.
- Generación de Alarmas
- Módulo de Reportes para el Área Jurídica

1.1.2.1 Modelado del Ingreso de Demandas al Sistema.
Como se mencionó anteriormente, los casos a desarrollar fueron los pertenecientes al área JURÍDICA, la cual es encargada de llevar a cabo el seguimiento de todas las demandas recibidas hasta su proceso de resolución, anulación en caso de no proceder, amparo promovido por el demandado y todos los recursos legales requeridos por la ley, siendo con esto, una de las áreas principales dentro de la dependencia.

A manera de inducción sobre la suite de Oracle CRM, ésta herramienta sirve para configurar un flujo de trabajo (mejor conocido por su nombre en inglés Workflow), el cual se puede definir como una serie de tareas que modelan situaciones junto con su información adicional (dependiendo lo que se esté modelando) por lo que se debe tener al menos una tarea configurada, en donde a cada tarea se le asocian estados (al menos 1 debe ser configurado en cada tarea), cada cambio realizado al estado de una tarea, puede llevar a invocar la generación de nuevas tareas o la terminación del flujo. Por lo tanto, a la totalidad de tareas configuradas para un caso que se modela, se puede entender como la totalidad de un flujo de trabajo, el cual, no necesariamente tiene el mismo comportamiento ya que dependiendo los estados elegidos en las tareas dependerá su comportamiento, dichos estados son cambiados de forma manual por algún usuario autorizado o bien, puede haber procesos en segundo plano como alarmas, que también afectan de manera automática a un flujo de trabajo cambiando los estados de sus tareas y por ende, afectando el flujo de trabajo en donde están configurados.

Esta configuración de tareas y estados, se realiza de manera manual dentro del administrador de flujos de trabajo (Workflow Administration) de la herramienta, en donde se pueden dar de alta tareas y sus estados. A manera de ejemplo para hacer más claro lo anterior, si se requiere modelar el flujo de trabajo destinado a la recepción de una demanda (el cual fue parte de nuestras actividades), se puede hacer de la siguiente manera:

Configuración de la información general del flujo a modelar.
De acuerdo con el proceso general de recepción de demandas al sistema, se debe generar un folio único alfanumérico junto con los datos necesarios en una demanda, por ejemplo el nombre y datos personales de la persona que interpone la demanda, entidad demandada, domicilio legal de la entidad demandada, fecha de ingreso al sistema, estatus de demanda, nombre del abogado asignado a la demanda, usuario que registra la demanda, fecha de recepción de la demanda, etc.
Estos datos se ingresan al sistema mediante una interfaz de usuario (con sus respectivas validaciones) permaneciendo visibles durante todo el flujo de trabajo permitiendo ser consultados en todo momento.

La siguiente parte, es modelar las tareas que complementan el flujo de trabajo con sus respectivos estados asociados, cabe mencionar, que esta parte dentro de la interfaz de usuario, se muestra dentro de una grilla de datos, mostrando las tareas asociadas en forma de lista y para cada tarea se integra un control de selección (combo box) en donde el usuario cambia los estados.

Por ejemplo, para el caso de modelar la recepción de una demanda, se puede pensar en las siguientes tareas (el caso real que se configuro, tiene alrededor de unas 100 tareas con al menos 20 posibles terminaciones):

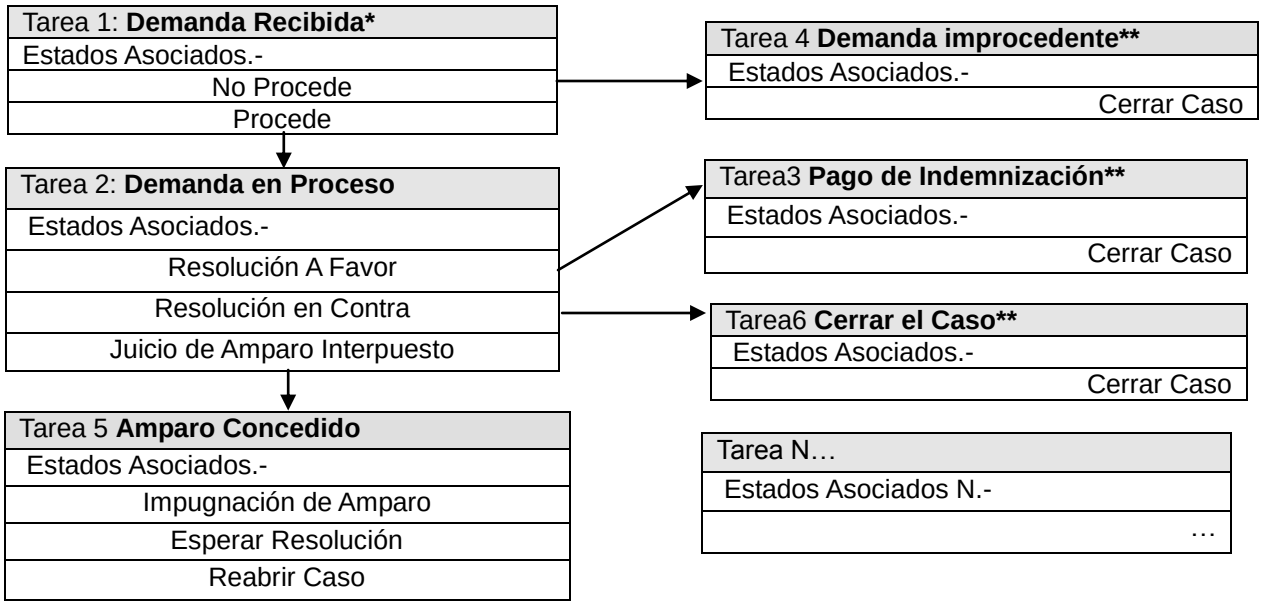


Fig. 1.2 Tipos de Tareas y Estados Asociados

*Tarea Inicial
**Tareas de Terminación de Flujo de Trabajo

Como se observa, se han creado un grupo de tareas, con sus posibles estados, el flujo puede tener distintos caminos, dependiendo que estados se vayan eligiendo, por ejemplo, si se ingresa una nueva demanda al sistema, después de que el usuario da de alta los datos generales, entonces la(s) primer(as) tarea(s) configurada(s) se muestran automáticamente, por lo que se le debe asignar alguno de sus estados disponibles, suponiendo que la Tarea #1 del ejemplo anterior es la única tarea inicial dentro del flujo, entonces se mostrara cuando el usuario inicie su operación, supongamos que el estado que se elige en la tarea es “Procede”, entonces de acuerdo al figura anterior, la Tarea #2 “Demanda en Proceso” será inmediatamente mostrada y así sucesivamente hasta llegar a una tarea de finalización la cual termina con el flujo de trabajo.

Parte del trabajo realizado, fue personalizar la interfaz de usuario que ofrece la herramienta CRM ya que por default, solo se agregan algunos campos básicos para la captura de la misma, lo cual, es poco útil, por ejemplo, el ID que se genera por default en la herramienta es de tipo numérico, lo cual puede no ser útil o requerido, teniendo que para nuestro modelo de ingreso de demandas, se pidió que el ID fuera un campo alfanumérico basado en ciertas reglas, también la herramienta asigna por default algunas cajas de texto, las cuales resultan poco practicas e insuficientes para registrar toda la información requerida como el nombre del demandante y demandado, abogado que lleva el caso, etc., finalmente, otro inconveniente es que al momento de guardar la información, se registra la fecha de creación, la cual puede utilizarse como parte del requerimiento como la fecha de ingreso de la demanda, sin embargo, la herramienta no muestra dicho registro.

Para resolver esta problemática, se debe modificar y adaptar la interfaz de usuario, de aquí en adelante abreviada por sus siglas en inglés UI (User Interface) junto con su código fuente, lo cual, es permitido hacer dentro de la herramienta siguiendo ciertas reglas, ya que de hacer cierto tipo de modificaciones, como algún tipo de re-ingeniería utilizando un compilador externo como C u otro tipo de cambios, se puede perder la garantía del producto, haciendo invalido cualquier tipo de soporte directamente con el proveedor. Parte del trabajo, fue el de personalizar las UI con los datos de entrada requeridos para adaptarlos a las necesidades de los usuarios, por lo que dentro de la suite del producto se tiene designado un espacio de trabajo dedicado para la construcción y modificación de interfaces de usuario para poder adaptarse a los requerimientos solicitados.
Cabe mencionar que las reglas dentro de cada UI como validaciones de campo, eventos a botones, formatos de fecha y hora se deben realizar como parte de la codificación de las mismas, la arquitectura esta basada en dos capas, front-end (la UI) y un back-end que contiene el código asociado codificado en lenguaje PL/SQL e interactuando directamente con la base de datos pudiéndose editar con cualquier editor de texto, aunque para este proyecto, se utilizó la herramienta TOAD para ORACLE, una vez editado, los cambios se ven reflejados después de que el proceso de compilación de la herramienta es ejecutado, dicho proceso, es ejecutado por el administrador de la suite, generalmente una persona de cada equipo tiene asignado dicho permiso, por lo que cada cambio era rápidamente puesto dentro del ambiente de pruebas para su revisión, en caso de falla durante la compilación, se debe buscar dentro del archivo de registro de errores (archivo .log) la descripción del error arrojado y corregirlo.

1.1.2.2 Generación de Alarmas.

Una vez que se desarrolló el flujo de trabajo del área jurídica y los usuarios dieron su aprobación, se procedió a desarrollar el proceso de alarmas, dicho proceso consiste en la ejecución de tareas en segundo plano dentro del sistema, no se deben confundir esto con las tareas configuradas explicadas anteriormente, siendo conceptos distintos, las alarmas son fragmentos de código (PL/SQL) que se ejecutan dentro de un plano independiente de acuerdo a condiciones establecidas afectando directamente a un flujo de trabajo o bien haciendo otro tipo de actividades como enviar correos electrónicos.
La necesidad de alarmas surge de automatizar ciertas situaciones dentro de algún flujo de trabajo, por ejemplo, supongamos que se tiene la siguiente tarea configurada (fig. 2) y el usuario manualmente le ha asignado alguno de los estados abajo descritos:

Tarea: **Revisión de demanda**
Posibles Estados:
Solicitar Nueva Revisión
Improcedente

Tabla. 1.1 Tarea Revisión Demanda con posibles estados.

Supongamos que dentro del proceso, tenemos que una demanda puede ser declarada como improcedente, pero también puede tener una segunda revisión, siempre y cuando la parte demandante solicite dicha revisión en el período de tiempo exigido por la ley, pero que de no ser solicitada, entonces el caso continua como improcedente sin ninguna opción legal para que el demandante solicite una nueva revisión, entonces es necesario para los abogados, notificar a su cliente que dicho recurso de revisión está disponible, pero de no ser solicitado, entonces el caso podría no tener una resolución a su favor.

La alarmas son una pieza clave para realizar actividades que ayudan en ciertas circunstancias en las cuales es muy complicado llevar un control manual de alguna situación, por ejemplo, regresando a la situación planteada anteriormente, puede ser útil configurar una alarma que envié un correo electrónico a los abogados inmediatamente que algún caso pueda ser declarado improcedente y con esto, el abogado que lleva el caso puede notificar a su cliente si desea interponer el recurso de una segunda revisión que legalmente está permitido dentro de la ley.
La configuración de las alarmas, fue parte de las actividades asignadas, dicha configuración viene dentro de la suite del CRM teniendo su propio marco de trabajo, en el cual, se elige un flujo de trabajo existente, se muestran sus tareas asociadas de manera gráfica a través de un diagrama y se elige la(s) tarea(s) y bajo qué estado(s) de la misma se puede disparar una alarma. La herramienta abre un formulario solicitando los datos del tiempo en que se debe iniciar la ejecución de la alarma, tipo de alarma (re-cursiva, única), configuración de correos electrónicos de las personas que serán notificadas una vez que la alarma se lance y opcionalmente código asociado a la alarma, el cual según sea el caso, puede cambiar el estado de alguna otra tarea, generar nuevas tareas, e incluso alterar el flujo de trabajo en sus datos generales.

1.1.2.3 Módulo de Reportes Área Jurídica.

Finalmente, el módulo de reportes fue una de las asignaciones más laboriosas, para ello se utilizó como ya se mencionó anteriormente la herramienta Discoverer Reports de Oracle, la cual permite asociar los resultados de una consulta SQL directamente a la UI cliente de la herramienta, en donde se muestran los resultados con distintos tipos de ordenamientos aceptando además filtros de entrada. Se crearon alrededor de 40 reportes, algunos de ellos bastantes complejos, debido a que el usuario comenzó a pedir cierta lógica un tanto compleja, para explicarlo mejor, a veces solicitaba, que un reporte de acuerdo a sus parámetros de entrada mostrará uno u otros resultados, escondiera campos o agregara dinámicamente otros, poner valores por default para ciertos usuarios, reaccionar a ciertos parámetros de entrada como una fecha la cual al validarse si cumplía con ciertas condiciones mostrara ciertos resultados o en caso contrario otros, etc. Este tipo de requerimientos generó mucho roce con los usuarios, debido a que no entendían los alcances de la herramienta, siendo que no es un entorno de programación capaz de responder a diferentes eventos de entrada como para realizar lógica compleja.
Afortunadamente un 80% de los requerimientos e inquietudes solicitadas por los usuarios fueron cubiertas de manera satisfactoria, para el 20% no completado, se tuvo que explicar las limitaciones de la herramienta y por qué ciertos casos, no eran posibles de realizar en la manera que ellos tenían pensando, sin embargo se lograron acuerdos para resolver de una manera diferente sus inquietudes obteniendo finalmente su visto bueno y aprobación.

La manera de trabajar en el desarrollo del módulo de reportes, fue acordando construir 10 reportes por semana teniendo una junta con los usuarios al inicio de cada semana para su revisión y aprobación, durante estas juntas, se presentaba la ejecución de los reportes con datos tomados desde una copia del ambiente de producción para que la información fuera lo más cercana a la realidad, en dicha junta, se cotejaba la información, formato, parámetros de entrada, ordenación e imágenes para que los usuarios dieran su visto bueno u sus observaciones, las cuales quedaban pendientes para la siguiente semana y en ese momento tenerlas resueltas.

Un problema que se presentó durante la fase de desarrollo y pruebas de reportes, fue que la ejecución de algunos de ellos comenzó a ser bastante lenta. Al analizar el problema, a pesar de que las consultas eran correctas y optimizadas, los reportes tomaban un tiempo de generación muy alto debido a que obtenían una gran cantidad de información histórica siendo esto la principal causa de la tardanza, la primer solución fue crear objetos de base de datos denominados “Vistas” a partir de las consultas que tenían una demora significativa, pero a pesar de que esto redujo considerablemente los costos en cuanto a tiempos, los reportes seguían demorando en su ejecución, por lo tanto, se optó por otra solución permitida por el manejador de base de datos, siendo el crear otro tipo de objeto, denominado “Vistas Materializadas”, las cuales son muy similares a una Vista pero con la diferencia de que estas tienen un proceso de carga en su definición, es decir, cuando se crean, se debe definir cada vez que se actualiza su información, por lo que estos objetos no requieren ningún tipo de pre-compilación durante su consulta, quedan totalmente fuera de cualquier transacción que pudiera hacer algún bloqueo dentro de los registros y pueden ser tratados como si fuesen una tabla más, por lo que es posible también configurar índices que optimizan significativamente búsquedas y ordenamientos. La configuración de recopilación de información en las vistas materializadas se configuro diariamente a la media noche, para que al siguiente día el usuario tuviese todo el histórico menos el día actual, lo cual no causo ningún inconveniente ya que la mayoría de los reportes que se basaron en este tipo de objetos, no requerían información actual e incluso del día en curso.

1.2 Proyecto Asignado HIR CASA Inmobiliaria

Giro. Empresa dedicada a Préstamos personales y créditos para adquisición de autos e inmuebles.
Posición Desempeñada: Desarrollador Visual Basic Jr.
Líder Técnico: Juan Francisco Rosales
Project Manager: Eduardo Pineiro
Personas a mi cargo: Ninguna.
Organigrama del Proyecto:

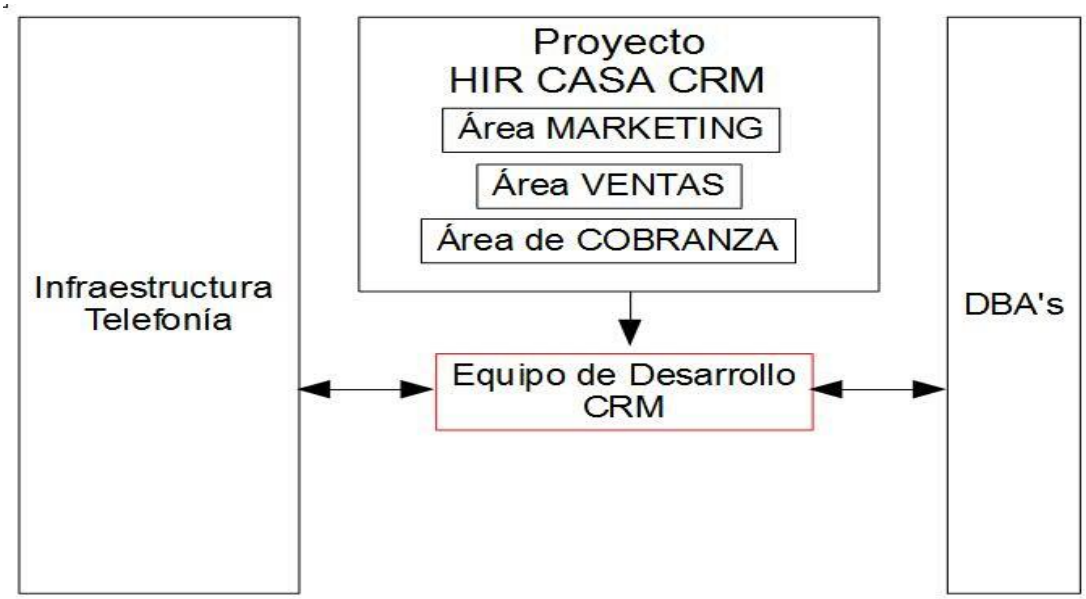


Fig. 1.3 Organigrama Proyecto HIR CASA

1.2.1. Introducción General del Proyecto.

Se desarrolló para esta empresa privada, un CRM (Customer Relationship Management) personalizado, el cual modelaría flujos de trabajo para sus distintas áreas, de las cuales, se destacan las siguientes:

Marketing.- En esta área se busca aperturas de nuevos créditos, mediante campañas que permitan atraer nuevos clientes que buscan una solución crediticia para los servicios que la compañía ofrece, siendo estos, principalmente los préstamos para adquirir una casa, personales o para un automóvil.

Ventas.- Se da seguimiento a los clientes que ya están interesados en alguno de los productos o ya tienen un crédito aprobado dentro de la compañía, se le da seguimiento a sus crédito(s) hasta la finalización de los mismos.

Cobranza.- Esta área se dedica a realizar la labor de cobranza a todos los clientes morosos que tienen una deuda que ya comenzó a generar intereses considerables debido a lo atrasado en el pago de sus créditos.

Tecnología Utilizada.
El sistema se construyó utilizando lenguaje de programación Microsoft Visual Basic con un motor de base de datos SQL Server 2000 corriendo en un servidor con sistema operativo Windows 2000 Server.

1.2.2 Resumen de las Actividades Realizadas.

A continuación se enumeran las actividades realizadas durante mi estancia en el proyecto, en las cuales participe directamente:

- Desarrollo del Módulo de Campañas.
- Desarrollo del Módulo de Campañas Automáticas.
- Proceso de Migración a la Base de Datos.
- Recepción de Listas para Carga de Campañas de Marketing.
- Módulo de Reportes.

1.2.2.1 Desarrollo del Módulo de Campañas.

Definición de Campaña.- Una campaña se define al acto de contactar directa o indirectamente por alguno de los siguientes medios: llamada telefónica, correo electrónico, correo convencional, mensajes de texto vía celular, etc. a los clientes o potenciales clientes con los cuales se mantiene o desea mantener una relación derivada de la adquisición de alguno de los distintos productos ofrecidos dentro de la compañía.
Para el desarrollo del módulo de campañas a mi cargo, únicamente se trabajó la campaña con contacto mediante llamada telefónica y correo electrónico. Básicamente los supervisores de cada área asignaban a sus operadores una lista de clientes haciendo una distribución de acuerdo a su experiencia, por lo que cada operador, una vez que se firmaba al sistema, abría la UI de campañas, en donde se mostraba una lista de potenciales clientes junto con la siguiente información: el nombre del cliente a contactar, su teléfono de contacto preferido, un teléfono secundario, horario preferido para llamar, correo electrónico, un icono para enviar un correo electrónico (si aplica), un botón de tipo check-box que se ajusta a verdadero una vez que se ha hecho una llamada satisfactoria o se ha enviado un correo electrónico, una vez hecho lo anterior, el registro es eliminado de la lista y el operador continua con los registros restantes hasta terminar con su lista de clientes asignados (o su campaña). Una vez que un operador cumple con su lista, el supervisor del mismo es notificado en tiempo real por medio de un correo electrónico, por lo que entonces, puede re-asignarle más registros, los cuales se toman de otros operadores que pertenecen a la misma campaña y que aún no han terminado con su lista, esto también en tiempo real.
El requerimiento para la distribución de listas a asignar, fue manteniendo una preferencia de asignación de acuerdo a la experiencia de cada operador, por ejemplo, los operadores con alta experiencia generalmente terminan sus listas en menor tiempo que los operadores de nuevo ingreso, por lo que se agregó un atributo especial en los operadores denominado ranking, el cual es un numero entero que va del 1 al 3, teniendo que el 1 representa mayor experiencia y el 3 menor, con este número, el supervisor asignaba dentro de los parámetros globales del sistema el porcentaje de peso que cada ranking debía tener para una campaña específica, por ejemplo, el ranking 1 puede tener un peso de 80%, el 2 el 15% y el 3 tanto solo del 5% restante, por lo que en base a lo anterior, se construyó un algoritmo de distribución, dando prioridad durante la asignación de listas a los operadores, tomando su ranking y el peso asignado al ranking, a continuación se muestra un ejemplo del algoritmo:

Campaña para el Área de Marketing: **“Prestamos Crediticio para Automóviles con tasa del 3% anual”**
Peso al Ranking 1: 80%
Peso al Ranking 2: 15%
Peso al Ranking 3: 5%

Tabla. 1.2 Campaña con Rankings.

Teniendo los siguientes datos:

Número de Clientes a contactar para esta campaña: 1000
Número Total de Operadores: 10
Número de Operadores Ranking 1: 5
Número de Operadores Ranking 2: 3
Número de Operadores Ranking 3: 2

Porcentaje de peso para los distintos rankings:
Ranking 1= 80%
Ranking 2= 15%
Ranking 3= 5%

De acuerdo a lo anterior, se tiene una campaña con una lista de 1000 clientes a los cuales 10 operadores deben contactar, de esos 10 operadores 5 tienen ranking de 1, 3 tienen un ranking de 2 y finalmente los últimos 2 restantes tienen un ranking de 3, de acuerdo a las necesidades del supervisor, para esta campaña, se debe generar una distribución de asignación del 80% a los operadores de ranking 1, el 15% para los operadores del ranking 2 y finalmente un 5% para el ranking 3, teniendo entonces el siguiente resultado del Algoritmo de Distribución de Acuerdo al porcentaje de Peso de acuerdo al ranking:

Distribución de clientes asignada por ranking:
Ranking1=800 (80%)
Ranking2=150 (15%)
Ranking3= 50 (5%)

Distribución asignada por Operador:
Operadores de Ranking1=160
Operadores de Ranking2=50
Operadores de Ranking3=25

Tabla. 1.3 Resultado de Campaña con Rankings a operador.

Por lo tanto, los 5 operadores que tienen ranking1, tendrán en su área de trabajo 160 registros que contienen los nombres de los clientes a quienes deben llamar, los de ranking 2 50 y finalmente los de más bajo ranking solo 25. Si se hace la multiplicación de operadores por número de registros asignados, sumando finalmente los 3 rankings se tendrá:

(160x5) + (50x3) + (25x2)=1000

que es el número de registros total que se solicitó distribuir para la campaña.

Lo anterior es tan solo un ejemplo, obviamente en el algoritmo original se tienen validaciones y tratamiento a posibles excepciones, por ejemplo que no haya operadores de cierto ranking, verificación de que exista una configuración de porcentaje de pesos por campaña, verificar de que exista un ranking configurado para el usuario, redondeo de números no enteros al asignar, etc.
La interfaz gráfica de este módulo se dividió en dos partes, una parte para el rol de operadores, que como ya se explicó anteriormente es en donde se muestran los nombres de los clientes, correo electrónico y números telefónicos, entre otros datos, para que los operadores pudieran contactarlos de acuerdo a sus lista asignada y otra parte, es para el uso exclusivo de los supervisores, dicho modulo se explicará a continuación, pero solo para finalizar, para las campañas realizadas por el departamento de Marketing, se agregó a la UI existente un apartado conocido como el “Speech”, dicho apartado es otra interfaz que surgió de la necesidad de no omitir información importante durante un proceso de llamada a los potenciales clientes, una vez que un operador estaba en una llamada, la UI de speech podía ser abierta mostrando textos con información importante sobre la campaña en la que se estuviese trabajando, como datos de promociones, descuentos, fechas de inicio y fin etc.

Para la UI diseñada para los supervisores, una vez que ingresan al sistema, se muestra la información de todos los operadores que están bajo su cargo (subordinados) en una vista de tipo árbol pudiendo realizar las siguientes operaciones:

- Ver el nombre de los operadores conectados, el horario en que se conectaron y la IP (nombre) de la máquina en donde están conectados. Para esto solo se colocó un icono con una imagen en verde que le indicaba que su operador está conectado y un icono similar pero en color rojo que indica que esta desconectado. Esto sirvió, para evitar el uso de duplicidad entre los operadores, ya que solo se permite que un usuario inicie una sesión por equipo, si el sistema detecta que un mismo ID trata de conectarse en más de un equipo, se envía un correo al supervisor para tomar las medidas correctivas.

- Asignación manual de listas mediante un evento arrastrar-soltar (drag & drop). Como se mencionó anteriormente, un supervisor distribuye o redistribuye las listas de sus campañas, pero además de esto, se agregó la funcionalidad para que una lista pueda ser asignada manualmente al operador, el supervisor toma los registros que desea mediante un evento arrastrar soltar con el mouse y arrastra los registros soltándolos sobre el icono del operador deseado sin tomar en cuenta su ranking, una vez realizado esto, el operador podrá ver en tiempo real que le fueron asignados más registros a su lista para trabajar con ellos.

1.2.2.2 Desarrollo del Módulo de Campañas Automáticas.

Se requirió un módulo denominado campañas automáticas, las cuales eran principalmente útiles para el área jurídica. Este tipo especial de campaña, consiste en hacer llamadas telefónicas sin necesidad de un operador, por ejemplo, tener la posibilidad de agendar contacto con los clientes de una lista a cualquier hora del día (por ejemplo de las 2AM a 5AM), tiempo en el cual, se realizan las llamadas telefónicas. Este tipo de campaña tuvo su origen dentro del área jurídica debido a que ellos requerían hacer este tipo de campañas para todos sus clientes morosos en horas de la madrugada con la finalidad de causarles molestia y presionarlos para realizar sus pagos atrasados.

Para este nuevo módulo se agregó una nueva tabla de configuración de campañas automáticas al sistema, pero a diferencia de las que se realizan con la intervención de un operador, se agregó un campo que persiste el ID del código de la grabación que se ejecutara durante cada llamada, fecha/hora de inicio y fin durante el cual se ejecuta la campaña.
Al analizar este requerimiento, se determinó que este nuevo módulo no era necesario incluirlo dentro del sistema, dado que ningún operador sería requerido, por lo que se construyó fuera del mismo utilizando el lenguaje de programación JAVA mediante una aplicación de tipo consola (Standalone), la cual, se conectaba a la base de datos del sistema para localizar las campañas automáticas, la configuración de horario de ejecución y las listas asociadas para iniciar con el proceso de llamadas.

Esta pieza de software se realizó en conjunto con otro equipo de desarrollo, que nos proporcionó un API (application programming interface) pública que recibe parámetros de entrada del número telefónico a llamar (con los respectivos códigos de área para el marcación de números foráneos) y el código de grabación que se ejecutara durante la llamada telefónica, dicho equipo de desarrollo trabajaba con la empresa AVAYA quienes brindan este soluciones de redes telefónicas, faxes y conmutadores, las pruebas se realizaron en conjunto con dicho equipo, para garantizar la correcta funcionalidad del módulo.

1.2.2.3 Proceso de Migración a Base de Datos.

Parte de las tareas fueron la migración de la información a la base de datos relacional que fue diseñada y como se mencionó al inicio, es una base de datos Microsoft SQL Server, esto con la finalidad de unificar y tener la información en un solo lugar. Para el proceso de migración, se tuvo que hacer un inventario de los distintos programas que utilizaban los departamentos anteriormente, encontrando que principalmente, la mayor parte de la información estaba guardada en bases de datos Microsoft Access y otra parte en múltiples archivos de hojas de cálculo Microsoft Excel.
Para la migración desde Microsoft Access, no se tuvo mucho inconveniente, debido a que ambas herramientas (Microsoft SQL Server y Microsoft Access) pertenecen al una misma suite de productos Microsoft, por lo que se tienen utilerías de migración entre ambas, haciendo el trabajo relativamente sencillo, sin embargo para el proceso de migrar toda información proveniente de las hojas de cálculo, se tuvo que desarrollar una pieza de software capaz de obtener la información de las mismas para después insertarla a la base de datos, dicho software fue construido con el lenguaje de programación JAVA utilizando un Api denominada **jxl** que permite manipular hojas de cálculo, exponiendo una serie de simples métodos públicos, para obtener la información dentro de una hoja de cálculo, haciendo que el programa tan solo recorriera las filas y columnas de las hojas de cálculo mediante sentencias de repetición y una vez obtenida, se inserta a la base de datos ahorrando mucho tiempo que haciéndolo de manera manual.

1.2.2.4 Recepción de Listas para Carga de Campañas de Marketing.

Un problema que surgió durante la fase de pruebas al sistema por parte de los supervisores de las diferentes áreas, fue que la carga de campañas que contenían altos volúmenes de información era demasiado tardada y propensa a errores.
Esto genero la necesidad de crear alguna forma alternativa con la cual se pudieran cargar las campañas de una manera más rápida y sencilla que incluso pudiera ser independiente de la intervención de los supervisores, por lo que, al analizar el problema nos dimos cuenta que todas las listas de clientes para asociar con las campañas se recibían durante los fines de semana dentro de archivos tipo CSV (comma separated values) a través de un proveedor externo a la compañía, por lo que, esto fue la pauta para automatizar el proceso, haciendo un nuevo programa capaz de tomar y leer archivos CSV una vez que eran recibidos, para cargar masivamente toda la información directamente a la base de datos.
La solución fue automatizar el proceso mediante la ejecución de un programa (construido en Java) el cual se invoca dentro de un archivo “batch” del sistema operativo (archivo .bat) el cual buscaba los archivos enviados por el proveedor durante los fines de semana para procesarlos e insertarlos en la base de datos, con esto, los supervisores tenían las listas de clientes disponibles de manera automática sin la necesidad de insertarlas manualmente, en caso de un error durante el procesamiento de los archivos, el programa envía un correo electrónico a la gente de soporte y supervisores para tomar las acciones pertinentes.

1.2.2.5 Módulo de Reportes.

La empresa HIR Casa compro una herramienta de reportes teniendo gente especializada en la misma, por lo que todo el módulo de reportes quedo a cargo de dicho equipo, sin embargo, algunos reportes sobre información estadística de campañas fueron requeridos dentro del sistema, para su realización se utilizó la herramienta de software Crystal Reports con la cual, se pueden generar reportes a partir de consultas SQL aplicadas directamente en base de datos con parámetros de entrada para limitar los resultados y controlar el performance.

1.3 Proyecto Asignado IDV Información Digital Vértice, SA de CV

Giro**.** Publicidad.
Posición Desempeñada: Desarrollador Visual Basic Jr.
Líder Técnico: Juan Francisco Rosales Correa
Project Manager: Eduardo Pineiro
Personas a mi cargo: Ninguna.
Organigrama del Proyecto:

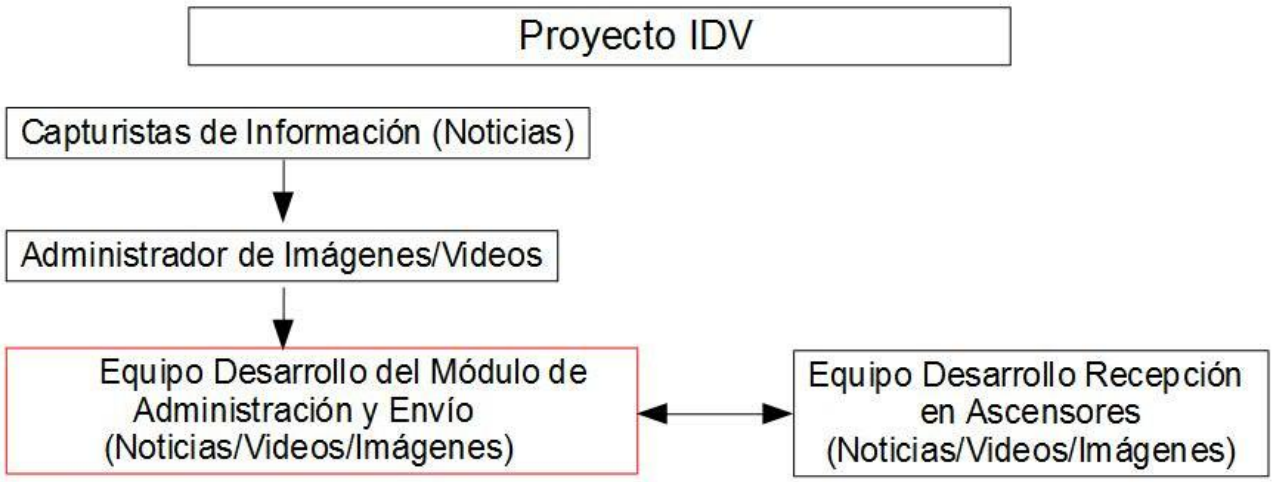


Fig. 1.4 Organigrama Proyecto IDV

1.3.1 Introducción General del Proyecto.

Empresa dedicada a la publicidad, cuya principal actividad es mostrar anuncios dentro de ascensores ubicados en distintos edificios dentro del área metropolitana, por ejemplo la Torre Optima 1 y 2 ubicadas sobre Paseo de las Palmas DF, el edificio OCE en la zona de Santa Fe, en los edificios corporativos de Oracle y Microsoft, etc.
La forma en la cual se presentan los anuncios, es mediante pantallas colocadas dentro de los ascensores giradas de forma vertical, la pantalla donde se muestra la información esta proporcionalmente dividida en dos partes, en una parte se tiene un video con publicidad diversa y en la otra parte se colocan noticias de distintos tipos, entre las que se destacan, Cultura, Deportes, Espectáculos, Finanzas etc.
Dentro de esta empresa se trabajó dividiendo el proyecto en 2 equipos, uno (del cual yo era parte) desarrollaría la construcción del sistema de administración, el cual consiste principalmente en el ingreso y envío de noticias, videos e imágenes a los distintos edificios

El segundo equipo se centraría en la construcción de otro sistema encargado de la recepción de la información proveniente desde el sistema de administración, dicho sistema se instaló dentro de cada servidor en cada edificio y estaría construido con el sistema operativo Linux SUSE, debido a que con este tipo de sistema operativo, se pueden crear instalaciones que permite tener varias instancias con salidas a un monitor de manera virtual o físicas (Display Screens) con cualquier orientación, por lo que, se puede instalar un servidor físico en alguna parte del edificio y desde allí abrir varias monitores con su propia información haciendo que en las pantallas de cada ascensor se pueda mostrar información de manera independiente, a la que hay en otro ascensor.

1.3.2 Resumen de las Actividades Realizadas.

A continuación se enumeran las actividades realizadas durante mi estancia en el proyecto, en las cuales participe directamente:

- Desarrollo de los Catálogos del Sistema
- Módulos de Envío de Noticias, Imágenes y Videos.
- Proceso de Envío de Imágenes y Videos

1.3.2.1 Desarrollo de los Catálogos del Sistema.

La primera parte del proyecto, fue desarrollar las interfaces de usuario para la administración de los distintos catálogos dentro del sistema, la tecnología utilizada para este proyecto, es el lenguaje de programación Visual Basic, una base de datos SQL Server 2000 sobre un servidor Windows 2000 Server.

- **Módulo de Edificios.**

Dentro de este módulo, el usuario da de alta los edificios en donde se tiene operación de publicidad, junto con otra información relevante al mismo como su dirección IP (la asignada al servidor del edificio), puerto SFTP del servidor junto con su usuario y contraseña el cual es utilizado para el envío de archivos, información de los ascensores con los que cuenta el edificio, entendiéndose que por cada ascensor hay una pantalla, etc.

- **Módulo de Administración de Noticias.**

El siguiente catálogo desarrollado fue el de ingreso de noticias, en el cual los usuarios dan de alta noticias, subdivididas por un tipo como lo es: Cultura, Deportes, Finanzas, Espectáculos etc. También se da de alta una imagen de fondo preferida para cada tipo de noticias, por ejemplo, para las noticias de Deportes, se puede asociar una imagen con un balón de fútbol soccer, no así para las de finanzas, en las cuales podrían presentarse con un fondo azul y una imagen con un símbolo de pesos, etc, al final esta imagen asociada a cada tipo de noticia se podía cambiar en cualquier momento, por ejemplo, en época navideña se colocaban otro tipo de imágenes, con un fondo relevante a dicha temporada.

Todas los campos de la tabla tienen un borrado lógico, es decir, se activa o desactiva el registro mediante un campo booleano. Los usuarios de la empresa, en su mayoría capturistas, reciben mediante servicios tipo RSS la información de noticias, eligiendo las que serán proyectadas en las pantallas y descartando las que no, estas noticias pueden durar desde 1 día hasta una semana, dependiendo de su relevancia. Dentro de la UI, las noticias se dan de alta mediante una caja de texto múltiple (Multitext), en donde se escriben las noticias utilizando tags HTML válidos dando formato a su presentación, por ejemplo, sea la siguiente noticia deportiva:

```
MEXICO DERROTA A BRAZIL EN EL MUNDIAL DE FUTBOL
```

También puede escribirse usando tags HTML válidos como:

```
MEXICO <b>DERROTA</b> A BRAZIL EN EL MUNDIAL DE FUTBOL
```

O

```
<q>MEXICO <b>DERROTA</b> A BRAZIL EN EL MUNDIAL DE FUTBOL</q>
```

Por lo que la noticia se presenta de la siguiente forma en las pantallas como:

MEXICO DERROTA A BRAZIL EN EL MUNDIAL DE FUTBOL

MEXICO **DERROTA** A BRAZIL EN EL MUNDIAL DE FUTBOL

"MEXICO **DERROTA** A BRAZIL EN EL MUNDIAL DE FUTBOL"

- **Módulo de Administración de Videos.**

Este módulo es el más complejo del sistema, debido que aquí se realiza el contrato que genera el costo para las distintas empresas que se desean anunciar enviando los videos que serán proyectados dentro de los tiempos comprados.
Básicamente, las compañías anuncian sus productos o campañas promocionales por medio de videos que se proyectan en lapsos de tiempo bien definidos con una periodicidad variable según el contrato, existen diferencias entre un anuncio y campaña, ya que en el anuncio, la empresa envía un video con un tiempo límite de duración que se proyecta por el periodo contratado en el que es transmitido, no así en una campaña, en donde se realiza lo mismo que un anuncio, pero con la opción de transmitir diferentes videos a lo largo del periodo contratado, por lo que el anunciante envía la lista de videos y el orden en el cual deben presentarse, por ejemplo, si una compañía de autos quiere publicitar un nuevo modelo de auto a través de una campaña, entonces, puede tener una secuencia de videos en donde en los primeros días se exhibe un video anunciando la fecha del lanzamiento de su auto, en los siguientes días cambia el video y muestra algunos detalles técnicos del auto para concluir al final de su campaña con un video en donde se puede observar el auto, precios, promociones etc. Por lo tanto el módulo de administración de videos, es en donde se da alta el nombre del proveedor, fechas de transmisión (Inicio-Fin), tipo de transmisión (anuncio, campaña), orden de transmisión para el caso de campañas, nombre del video(s), correo electrónico del anunciante, costo de transmisión, restricción de edificios*.

*****Restricción a edificios. Ciertos videos no son permitidos para algunos edificios, como anécdota durante mi estancia en el proyecto, sucedió un día, que la empresa Microsoft anunciaba su sistema operativo Windows Server, por otro lado, en el edificio corporativo de SUN Microsystems (ahora Oracle Sun) se promocionaba su sistema operativo Solaris, cuando en ambos edificios se mostraron ambas promociones, se desato la molestia por ambas partes, dado que eran competencia directa, por lo que se pidió quitar de sus edificios dichos anuncios, respectivamente y fue entonces que se agregó el concepto de Edificios restringidos, en donde se agregaría el ID de los edificios en los cuales un video(s) no debe ser transmitido.

1.3.2.2 Módulo de Envío de Noticias, Imágenes y Videos.

Para enviar la información, a los distintos edificios, se construyó el módulo de envío de información, el cual era muy sencillo en cuanto a su interfaz gráfica ya que solo se tiene un botón de envío bajo una tabla de los edificios a los que se enviará la información, los cuales por default están seleccionados pero con la opción de que el usuario pueda elegir manualmente los edificios a los cuales se pueden enviar o no la información. Una vez que el usuario presiona el botón de envío, se abren 3 opciones a elegir: “Solo Noticias”, “Solo Videos” o “Todo”, el proceso es el siguiente.-

- Para el envío de noticias, el sistema construye en tiempo real las noticias que han sido dadas de alta para su envío dentro de un archivo XML, el cual, mediante una serie de tags va colocando recursivamente, por ejemplo:

```
<news>
       <notes>
           <type>Ciencia</type>
           <img>ciencia.png</img>
           <imgchecksum>1f369ed9eee756f449d4f2fd6e0011a3</imgchecksum>
            <note>
                <ID>1505</ID>
                <active>Y</active>
                  <order>1</order>
                <Text>
                  <![CDATA[ Descifrado a un 97% el genoma humano]]>
                </Text>
              </note>
              <note>
                <ID>1455</ID>
                <active>Y</active>
                <order>2</order>
                <Text>
                  <![CDATA[ TEST]]>
                </Text>
              </note>
       </notes>
       <notes>
       <type>Cultura</type>
          <img>cultura.png</img>
          <imgchecksum>9e367ed92ea756ffffd4f2f66e0022b5</imgchecksum>
          <note>
            <ID>1455</ID>
            <active>Y</active>
            <order>1</order>
                <Text>
                 <![CDATA[ Teatro de la Ciudad, presenta los <q>miserables</q> en el Museo del Chopo]]>
                </Text>
          </note>
          <note>
              <order>2</order>
              <Text>
                <![CDATA[Magna Exposición en el Castillo de Chapultepec, próximo viernes]]>
              </Text>
          </note>
          <note>
           <ID>1456</ID>
           <active>Y</active>
           <order>3</order>
           <Text>
              <![CDATA[ <q>Maquetas de la Ciudad<q>,museo del Chopo, solo <u>2 semanas</u> mas]]>
           </Text>
          </note>
       </notes>
</news>
```

Imagen Asociadas

Noticias

Fig. 1.5 XML para Construcción y Envío de Noticias.

Como se observa en la fig. 1.5, se muestra un fragmento XML, el cual se va construyendo por cada tipo de noticia, en el cual se agregan los tags que contienen el nombre de la imagen asociada de fondo, el texto de la noticia que será presentada, etc.

- Para los videos, el sistema envía dentro del mismo archivo XML bajo el tag <videos> los nombres de los videos que serán transmitidos, de forma similar al de las imágenes de fondo de las noticias arriba descrito, más adelante se explica porque se construye de esta manera:

```
<videos>
    <video>
      <name>vw.mov</name>
      <checksum>0f369ed9e98756f179d4f29d6e7755d3</checksum>
      <dueDate>10<dueDate>
      <campaingID>0</campaingID>
      <seq>0</seq>
    <video>
    <video>
      <name>palacioHierro.mov</name>
      <checksum>2ee904450ff378e57966630d6e4422e1</checksum>
      <dueDate>5</dueDate>
      <campaingID>0</campaingID>
      <seq>0</seq>
    <video>
    <video>
      <name>oracleSystems.mov</name>
      <checksum>7aae601c9f99b3759fbb2b385b7c8b6b</checksum>
      <dueDate>3</dueDate>
      <campaingID>0</campaingID>
      <seq>0</seq>
    <video>
</videos>
```

VIDEOS

Fig. 1.6 Archivo XML para envío de Videos

Como se observa, dentro del XML, para el caso de imágenes y videos, se envía el nombre del archivo, el checksum del archivo (imagen/video) y para el caso de videos, el número de días que será promocionado, si pertenece a una campaña y en su caso el número de orden en el que se deben presentar.
La definición de todos los tags de este archivo se realizó en conjunto con el equipo de construcción del sistema de recepción de información en los ascensores, revisando con ellos lo que necesitaban para interpretar la información de acuerdo a los tags definidos, mismos que serían interpretados por el software creado de su lado y que se ejecutaría en cada edificio al recibir el archivo XML desde nuestro sistema.

Con el archivo XML construido, se inicia el proceso de envío, el cual se realiza para cada edificio utilizando el protocolo SFTP, es por eso, que el sistema tiene configurado para cada edificio, su dirección IP, puerto SFTP, usuario y contraseña. En cada servidor de edificio se tiene asignada una ruta física en la cual se destinan los archivos de noticias, imágenes y videos. Una vez que el servidor recibe el archivo XML, lo primero que procesa son las noticias, para lo cual, revisa su directorio de noticias, el cual a su vez contiene subdirectorios de tipos de noticias (deporte, cultura, etc.), en caso de encontrar un nuevo tipo de noticia la cual no tiene un subdirectorio, simplemente se crea junto con la imagen de fondo con la cual se presentara.

Como se explicó, cada noticia consiste de un archivo con extensión .HTML siendo el nombre del archivo el ID de la noticia, de esta forma se puede identificar si esa noticia existe o no, sino existe entonces se genera un nuevo archivo .HTML con ese nuevo id, si ya existe el ID de noticia, entonces revisa el atributo activa, en caso de seguir activa no se realiza ninguna acción, en caso de tener el atributo inactiva, entonces el programa borra el archivo .HTML y la noticia no es mostrada nuevamente en las pantallas.

1.3.2.3 Desarrollo del Proceso de Envío de Imágenes y Videos.

Anteriormente se mencionó que dentro del archivo XML, se coloca el nombre del video o imagen, sin hacer mención a ninguna ruta, esto es debido a que, dentro del servidor de recepción como ya se mencionó, ya se tiene un directorio designado para videos/imágenes, en ambos casos, se utiliza el atributo “checksum” que utiliza el algoritmo MD5, para identificar que el archivo con el que se tiene que operar es correcto, por lo tanto, el sistema de recepción verifica en el archivo XML que recibe, los nombres de imágenes y videos, primeramente que existan por medio de su nombre y después ejecuta un checksum para verificar que ambos sean iguales, en caso de ser iguales, no se realiza ninguna acción a excepción de revisar para el caso de los videos que el atributo de días en los cuales el video es presentado siga vigente, esto debido a que en algunas ocasiones, los días contratados por solicitud de los clientes, se incrementan o en otras se pueden dar de baja por ciertas circunstancias como falta de pago, para esto, simplemente se envía el atributo <dueDate> con valor de 0 y así el sistema de recepción da de baja dicho video. En el caso de que un archivo difiera con el checksum o simplemente no exista, entonces desde el servidor de recepción de ese edificio, se envía un mensaje de petición al sistema de administración con el nombre del video o imagen que necesita, una vez que se recibe, el administrador abre una conexión SFTP al servidor del edificio para depositar el archivo requerido. Lo anterior se realizó como mejora debido a que en primeras versiones, el administrador siempre enviaba videos e imágenes a los servidores de edificios, ocasionando bastante lentitud debido al tráfico de la red que esto conlleva, sobre todo para el caso de videos, así que con la finalidad de reducir la carga y aumentar el performance del sistema, se decidió que únicamente archivos que no existieran o con algún problema de integridad durante el envío a los edificios, serían enviados.

Finalmente, se agregó un nuevo tipo de noticia, denominada INTERNA, este tipo de noticia surgió de la necesidad de los administradores de los edificios, para ofrecer noticias relevantes a su administración interna, por ejemplo, la siguiente noticia, es relevante únicamente para el edificio en donde es presentada:

“POR REMODELACIÓN A LOS ESTACIONAMIENTOS, EL VIERNES 28 JUNIO, UNICAMENTE LOS SOTANOS 1,2 Y 3 SERÁN HABILITADOS, DEJANDO EL 4,5,6 CERRRADOS, TOME SUS PRECAUCIONES”

para esto, se realizó una modificación al catálogo de noticias agregándose un campo adicional que tendría el valor del ID del edificio además de un nuevo tipo denominado “INTERNA”, por lo que durante el proceso de construcción y envío del archivo XML a un edificio en particular, si el sistema detecta que el valor del nuevo campo es igual al del edificio a donde se va enviar, la noticia debe ser incluida, en caso contrario simplemente se ignora y no se envía a ese edificio.

Capítulo 2
Consultora TCS (TATA Consultant & Services)

Giro. Consultoría en Sistemas.
Posición. Desarrollador Sr.
Dirección. Insurgentes Sur 664, Piso 2 Col. Del Valle, México D.F. 03100

2.1 Proyecto Asignado NEXTEL.

Giro. Empresa de Servicios de Radiocomunicación
Posición Desempeñada: Líder de Proyecto Jr.
Líder Técnico Interno: Mayeri Hirata
Project Manager: Hitesh Chaturvedi
Personas a mi cargo: Equipo de desarrollo Offshore India.
Organigrama del Proyecto:

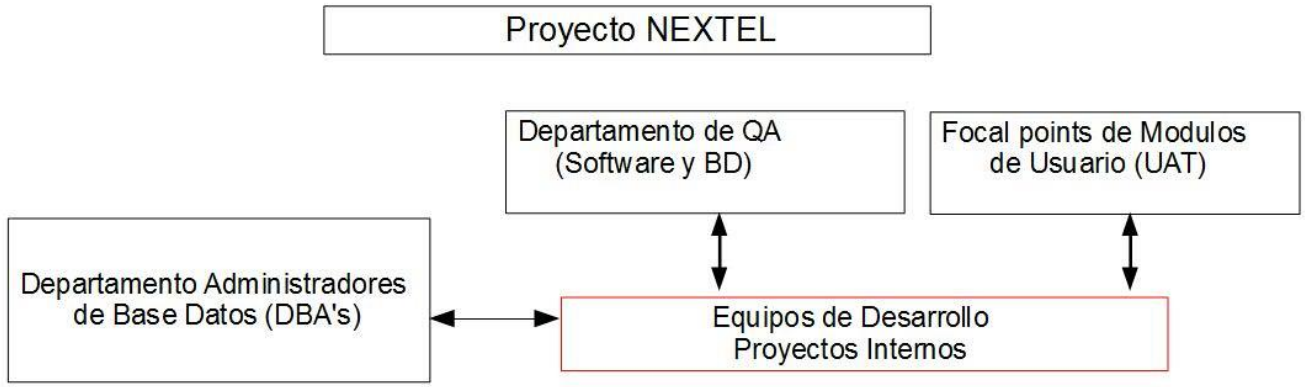


Fig. 2.1 Organigrama Proyecto Nextel

2.1.1 Introducción General del Proyecto.

Dentro del proyecto de Nextel la posición que ocupe fue la da Líder de Proyecto, teniendo a mi cargo a un grupo de desarrolladores en offshore, trabajando desde la India. Nuestro proyecto se integró a un sistema existente como un nuevo módulo funcional al cual se le dio el nombre de RENOCO (Sistema de Renovación de Contratos) respetando toda la arquitectura de programación y herramientas ya existentes, la cual, se puede resumir como la siguiente:

- Sistema de Software Multicapa, construido bajo tecnología Java:

Capa de Presentación	Framework de Struts (JSP,JScript)
Capa de Negocio	Enterprise Java Beans sin estado
Capa de Persistencia	Hibernate (DAOs)
Base de Datos	Oracle
Log del sistema	Log4J
Script de instalación	ANT
Servidor de Aplicaciones	Bea WebLogic Server

Tabla. 2.1 Arquitectura del Sistema

2.1.2 Resumen de las Actividades Realizadas.

Las siguientes tareas fueron parte de mis asignaciones durante el proyecto:

1) Supervisar la construcción e implementación del módulo de renovación de contratos, el cual tiene como finalidad obtener todos los clientes que cuentan con un servicio vigente con la compañía, pero que está a punto de vencer. Dicho vencimiento debe alertar a los operadores de marketing en sus colas de trabajo de manera automática con al menos 3 semanas de anticipación al vencimiento, para poder contactar a los clientes y ofrecer nuevos planes, equipos y tarifas atractivas que permitan continuar con su servicio de comunicación.

2) Hacer una carga masiva desde 3 sistemas diferentes, en donde se dan de alta contratos a las nuevas tablas creadas para el módulo de renovación de contratos a desarrollar, además que los sistemas de donde se tomara esta carga masiva, seguirán operando y vigentes como puntos de carga válida, por lo que se pidió también una solución para tener sincronizada la información proveniente de estos sistemas.

3) Implementar de manera adecuada el concepto de “Refurbish” en el módulo de renovación de contratos, el cual consiste en que si un cliente decide renovar su contrato, debe retornar sus equipos viejos, los cuales, deben agregarse al inventario como disponibles como respaldo. Lo anterior debido a que, cuando un cliente tiene algún percance con sus equipos (perdida, descompostura, robo, etc.) se le deben reemplazar a la brevedad con equipos similares, sin embargo, esto afecta seriamente al inventario, debido a que disponer de equipos directamente del inventario los agota para los nuevos clientes o a clientes próximos a renovar su contrato, por lo que se les pedía en la mayor de las veces, esperar a que llegaran sus equipos al almacén teniendo que hacerlos esperar algunos días. Por lo tanto, el agregar los equipos viejos al inventario para los casos de percance disminuyo la afectación, ya que estos equipos se destinaban para los clientes con incidentes de manera temporal sin dejar de prestarles el servicio que tienen contratado y además permite que los nuevos clientes tengan a tiempo los equipos junto con el servicio que contrataron.

4) Creación de Reportes basados en el nuevo módulo de renovación de contratos a construir, estos reportes, son propuestos directamente por los usuarios y conciernen a sus necesidades operativas, por ejemplo, un reporte que muestre el nombre de los clientes a una próxima Renovación de Contratos durante los meses siguientes o un reporte para los supervisores del área de marketing para conocer cuántos clientes fueron atendidos durante cierto mes así como el operador que realizo la mayor cantidad de contactos.

2.1.2.1 Supervisión del Desarrollo de Módulo de Renovación de Contratos.

Para la supervisión en la construcción del nuevo módulo a desarrollar, fue necesario analizar la arquitectura existente, el código fuente, la estructura de las tablas de base de datos con las que se contaba, procedimientos/funciones u otros objetos de base de datos disponibles, reglas de programación, herramientas utilizadas para performance y pruebas de stress, reglas de programación para el desarrollo, acceso a ambientes de desarrollo, accesos a usuarios, acceso a repositorios de código fuente, reglas internas, etc.
Para esto, el equipo de desarrollo interno de la compañía, brindo su apoyo entregando manuales técnicos y de usuario de la aplicación a la cual se le agregaría el módulo a desarrollar, la última versión del código fuente, localización del código fuente dentro del repositorio de datos, diagramas UML de clases/secuencia del sistema, casos de uso iniciales de otros módulos, diagramas de base de datos, ejemplos de cómo crear una UI asociada a un menú nuevo/existente dentro de su sistema, generación de un proceso de negocio dentro de un EJB(Enterprise Java Bean), última versión del script de compilación del archivo de instalación (build.xml de ANT) para generar el archivo ear (enterprise application resource) a instalar en el servidor de aplicaciones, proceso administrativo para agregar/modificar/borrar objetos en la base de datos, proceso administrativo para pedir una copia reciente de la base de datos de producción para realizar pruebas, parámetros utilizados en el servidor de aplicaciones, accesos para base de datos, servidor de aplicaciones y de la aplicación en el ambiente de desarrollo, herramientas para calidad del software obligatorias a utilizar previo a una promoción con el departamento de QA, software del ambiente de desarrollo utilizado para la codificación (MyEclipse), acceso al servidor CVS para subir y obtener últimas versiones del código fuente, etc.
De acuerdo a lo anterior y para evitar contratiempos, conforme se transmitía información y conocimiento del sistema sobre el cual se debería integrar nuestro nuevo módulo, al mismo tiempo también se le transmitió al equipo de desarrollo en offshore a mi cargo de la siguiente manera, si por ejemplo, se explicaba cómo realizar un entregable, al final del día se enviaba la documentación recibida explicando el proceso, junto con algún documento con observaciones personales que sirviera para clarificar y reducir dudas al equipo, al siguiente día se organizaba una sesión matutina para aclarar dudas e inquietudes en cuanto a lo explicado o recibir retro-alimentación de que toda la información fue comprendida.
Dentro de los tópicos que fueron más importantes para el cliente y en donde se hizo un mayor énfasis, fue a la calidad del software a entregar al departamento de QA, para esto, se promovió el uso de varios analizadores de código, los cuales verifican las buenas prácticas de programación, solo por dar un ejemplo, si se declara una variable que nunca se utiliza dentro del código desarrollado, éstas herramientas envían una advertencia mostrando que una variable ha sido declarada pero nunca se utiliza, otro ejemplo, es declarar un método estático dentro de un objeto y llamarlo a través de la instancia del mismo, en lugar de invocarlo directamente:

Correcto: `MiObjeto.MiMetodoEstatico();`

Incorrecto:
```
MiObjeto miObj=new MiObjeto();
miObj.MiMetodoEstatico; //Permitido por el compilador,
                        //pero mala práctica de programación
```

Cuando una pieza de software es enviada a QA, ellos se encargan de analizar el código fuente utilizando distintos analizadores de código, de acuerdo con las políticas internas de la compañía, solo se permite hasta un límite de advertencias, en caso de rebasar dicho límite, los cambios que se están entregando son automáticamente rechazados y devueltos para su corrección con una lista de problemas encontrados.

2.1.2.2 Carga Masiva de Contratos para Unificación de la Información.

Mientras el equipo de offshore trabajaba en el desarrollo del módulo de renovación de contratos, paralelamente se comenzó el proceso de unificación de información proveniente de los distintos sistemas en los cuales se registran contratos, para que sea insertada dentro de una nueva tabla que mantendría toda la información en un solo lugar. Para esta unificación, se tuvo que realizar entrevistas con los responsables de cada una de estas aplicaciones, para conocer a detalle las tablas en donde registraban la información relevante a los contratos de clientes, conocer los nombres de campos, accesos a las bases de datos, instancias, resultando de manera positiva ya que se pudo tener acceso a los diagramas entidad relación de estas bases de datos y además conocer que todos los sistemas utilizaban el mismo motor de base de datos (Oracle) en diferentes instancias.

De acuerdo a lo anterior, la solución de software que se propuso, fue que toda la unificación de información se ejecutaría desde la base de datos, sin que ningún sistema externo interviniera, por lo que se generó un nuevo objeto paquete en la base de datos (package), el cual, tendría 2 modos de ejecución de acuerdo a sus parámetros de entrada, siendo que para el caso en que no recibe ningún parámetro, entonces se realiza el proceso de tomar toda la información existente desde las tablas de todos los sistemas involucrados, para insertarlas a la tabla creada para este propósito de unificación, conocido como migración histórica de datos, siendo que éste modo de ejecución, tendría que ser ejecutado tan solo una vez. En el segundo modo de ejecución del paquete, se recibe un parámetro de fecha de entrada (INPUT), dicho parámetro indica a partir de qué fecha se limita la extracción de la información referente a los contratos desde los distintos sistemas, teniendo que este modo de ejecución tendría que ser ejecutado mediante un objeto “cron” dentro de la base de datos, diariamente, con la fecha actual menos un día (sysdate -1), con esto, los usuarios, tendrán la información vigente de todos los contratos con tan solo 1 día de retraso (el día actual), esto último no fue ningún inconveniente a los usuarios quienes estuvieron de acuerdo que se procediera de esa manera. Técnicamente, la realización de este paquete en base de datos involucro el uso de DBLinks para poder realizar la conexión a las distintas instancias de todos los sistemas, dentro del paquete se realizaron varios procedimientos en los que cada uno tiene como función obtener los datos de cada sistema, consultar los contratos vigentes y pasarlos a la tabla principal del sistema de renovación de contratos a nuestro cargo.

2.1.2.3 Desarrollo del Módulo de Refurbish.

Una vez que se completó la integración de todos los sistemas en una sola tabla, se comenzó el desarrollo del módulo de Refurbish, para esto se construyó una interfaz gráfica de acuerdo a los estándares obligatorios (“Struts Framework”) en la cual se agregó un botón de tipo “check” con la etiqueta: “Cliente Retorna Equipos en buen estado”, que por default se encuentra en estado deshabilitado (FALSE), como se explicó anteriormente, una vez que un cliente renueva contrato, opcionalmente puede retornar sus viejos equipos a cambio de algún beneficio o descuento, sin embargo, esto es opcional debido a que los equipos ya pertenecen al cliente, es por eso que cuando el operador asigna a verdadero la opción de retorno de equipos, es cuando éstos son entregados físicamente y se determina que están en buen estado. Si los equipos son retornados, entonces se muestra otra UI en donde se inserta el número de serie y modelo de los equipos que está retornándose, para que con esta combinación de valores, se den de alta al sistema e ingresen al inventario. Esta re-integración al inventario no tuvo complicación alguna debido a que el equipo de trabajo perteneciente al área de sistemas del almacén, proporciono procedimientos almacenados de base de datos, que se encargan de insertar un ITEM (equipo) al inventario utilizando su número de serie y modelo, por lo que únicamente se realizó la invocación de estos procedimientos, haciendo la integración al inventario de los equipos relativamente sencilla.

2.1.2.4 Desarrollo del Módulo de Reportes.

Para la realización del módulo de reportes, se utilizó una herramienta llamada Web Reports, dicha herramienta, tiene la principal característica de poder conectarse a los principales motores de base de datos comerciales (DB2, MySQL, Oracle, MS SQL Server), extraer información mediante consultas SQL y mostrar los resultados a través de un front-end compatible con cualquier browser, debido a que los reportes construidos se generan dentro de un archivo con extensión “.HTML”, dichos archivos se generan dentro del servidor en donde se instala la herramienta, por lo que una vez que alguien se autentica en la herramienta, se puede acceder al repositorio de reportes para proceder con su ejecución. La entrega de reportes se revisa en dos áreas distintas, una de ellas son los usuarios que solicitaron los reportes y es en donde se discute la parte gráfica del reporte, sus resultados y el formato.
Una segunda área valida los costos de la consulta asociada al reporte, dichos costos son medidos desde un analizador de consultas de base de datos y debe tener un costo bajo, si el costo es superior al permitido, entonces, el reporte se rechaza para su optimización y si aún con la optimización el reporte mantiene un costo alto, como fue el caso de uno de los reportes asignados dentro del proyecto, entonces dicho reporte se envía a un comité con el departamento de base de datos, en dicho comité, se debe justificar las causas por las cuales el reporte tiene un costo alto a pesar de las optimizaciones que se han hecho, el comité analiza dichas razones internamente en conjunto con los usuarios que solicitaron tal reporte, para que en caso de aceptar, se pueda dar de alta el reporte al servidor de producción para su posterior utilización.

2.1.2.5 Proceso de Envío al departamento de QA (Quality and Assurance).

Parte de las actividades, fue el proceso de promoción de entregables al cliente, este proceso compete al área de desarrollo únicamente involucrando quizás a otros equipos, debido a que éstos, trabajan paralelamente en la codificación algún otro módulo, por lo que se tiene que revisar con ellos para verificar sus últimos cambios en el código fuente y generar una versión confiable que incluya los últimos cambios de todos los equipos, esto a pesar de que el código se encuentra versionado dentro de un repositorio. Para el proceso de entrega del paquete de instalación, que se despliega (Deployment) dentro de servidor de aplicaciones de pruebas (para posteriormente instalar a un ambiente de producción), se utiliza un script desarrollado con la herramienta de software ANT, la cual, sirve para ejecutar tareas de manera secuencial a través de un archivo con extensión .XML generalmente nombrado build.xml, en donde se colocan los pasos necesarios para:

- Obtener la última versión del código fuente desde el repositorio de datos.
- Compilar el código fuente para generar los archivos de clase necesarios.
- Construir el archivo de instalación (archivo final con extensión ".ear" Enterprise Application Resource).
- Agregar (si es requerido) nuevas librerías y/o archivos auxiliares dentro del proceso de construcción/compilación necesarias para el funcionamiento de la aplicación.

El archivo .ear se envía a un servidor especifico en donde es tomado por el equipo de QA, todo lo anterior siguiendo un protocolo de correos electrónicos enterando a todos los equipos involucrados del nuevo entregable, junto con 2 archivos obligatorios: el caso de uso de lo que se entrega y la matriz de pruebas a ejecutar, este último es un archivo que describe paso por paso cada nueva funcionalidad añadida/modificada/eliminada dentro de la pieza de software que se está entregando, sirviendo además como evidencia de que lo que se está entregando ya ha sido probado por el equipo de desarrollo antes de su liberación. El equipo de QA, instala el nuevo software y sigue el documento de la matriz de prueba esperando obtener los resultados allí expuestos, si durante este proceso se detecta un defecto en el software, el proceso se detiene y se contacta con el equipo de desarrollo para aclarar cualquier duda sobre algún resultado dudoso o en un peor escenario, para notificar que se encontró un problema y el software se tendrá que regresar para su pronta corrección.
Durante las pruebas, el equipo de QA, sigue la matriz de prueba paso por paso, haciendo variaciones a sus pruebas utilizando distintos parámetros que de antemano se sabe que causaran errores, esto con la finalidad de revisar que el sistema es robusto y puede manejar cualquier escenario adverso, si todas las pruebas son correctas y todas las situaciones en las cuales surgen errores son correctamente controladas, entonces, el software se promueve al departamento de UAT para que se realicen pruebas de estrés con usuarios y si todo es correcto, finalmente se congelan las versiones y se promueve todo al ambiente de Producción.

La entrega de código fuente y ejecutable tiene el procedimiento arriba descrito, pero existe otra entrega simultanea que involucra únicamente cambios a la base de datos, la cual puede incluir tanto instrucciones de manipulación de datos (DML) como de definición de datos (DDL), para dicha entrega, se debe realizar un script entendible por el manejador de base de datos, con extensión .sql, para ser enviado y validado por el equipo de DBA's quienes ejecutan dicho script primeramente sobre un ambiente similar a producción, para asegurase que todo lo que se está entregando, se ejecuta sin errores y finalmente se ejecuta un segundo script de regresión el cual, regresa la base de datos a su estado original, por ejemplo:

Si el script a entregar contiene la siguiente sentencia:

```
INSERT INTO CUSTOMERS (ID,CUSTOMER_NAME,DESCRIPTION) Values (20,'Bill', 'CEO');
```

Se debe tener dentro del script de regresión una sentencia que elimine todo lo que haya sido afectado por el script de instalación:

```
DELETE FROM CUSTOMERS WHERE ID=20;
```

2.2 Proyecto Asignado BOFA MERRILL LYNCH

Giro**.** Casa de Bolsa
Posición Desempeñada: Líder de Técnico Sr.
Project Manager: Variable
Personas a mi cargo: Offshore India
Equipos de IT (Praxis, GoNet, Stefannini).

Organigrama del Proyecto:

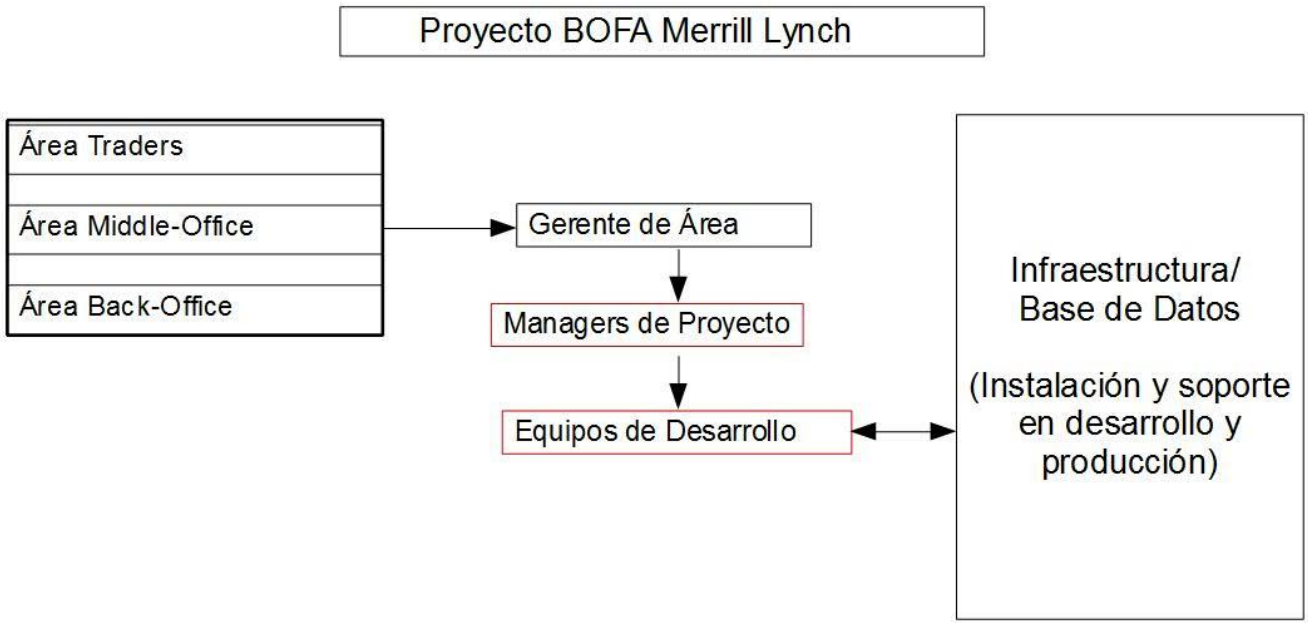


Fig. 2.2 Organigrama Proyecto Merryll Lynch

2.2.1 Introducción General del Proyecto.

La compañía Merrill Lynch (actualmente BOFA Merrill Lynch), es una compañía dedicada a la compra y venta de capitales en distintos mercados internacionales incluidos México, trabajando para distintos clientes y operando bajo las normas y regulaciones establecidas de cada país, en donde para el caso de México es la Bolsa Mexicana de Valores de aquí en adelante abreviada como BMV.

El proyecto consistía en dar soporte/construcción a una aplicación Multicapa construida con tecnología JAVA, dicha aplicación nombrada YAX (Palabra de origen Azteca que significa EL PRIMERO) la cual, sustituiría a otra existente llamada MIVIRO la cual, estaba construida dentro del Mainframe de AS-400 con una base de datos DB2, ambas aplicaciones tienen como principal función la generación de Ordenes para el proceso de compra y venta de acciones dentro de la BMV.
La aplicación como ya se mencionó, consiste en una arquitectura multicapa construida bajo los estándares J2EE (Java Enterprise Edition) con una base de datos Microsoft SQL Server, los procesos de negocios se construyeron dentro de la especificación EJB (Enterprise Java Beans), se cuenta con APIs para envío de correos JAVA MAIL, recepción y envío de colas de mensajes JMS (Java Message Services), capa de presentación WEB, un servidor de aplicaciones IBM Websphere todo instalado en un sistema operativo Linux RedHat Server.

La aplicación es bastante robusta e incluye varias tecnologías tanto para la parte del front-end como la parte del back-end además de utilerías externas como el envío y recepción de colas de mensajes a través de un servidor de mensajería TIBCO. La aplicación está dividida en **Subsistemas**, es decir, al ser tan grande y compleja la operación, se decidió que las clases estarían divididas entre varios subsistemas, los cuales mantienen una relación entre ellos para compartir información, por ejemplo:

División de la aplicación en sub-sistemas
ReferenceData
Orders
Securities
USBook
Reports
Accounts

Tabla. 2.2 Ejemplos de Subsistemas

Otra parte importante del sistema, es la base de datos, en la cual, muchos procedimientos almacenados y funciones auxiliares se crearon para aprovechar los recursos ofrecidos por el motor de base de datos brindando un mejor performance a la aplicación, sin necesidad de codificar nada dentro de la misma, permitiendo también, la corrección de errores sin necesidad de una re-compilación del código o reinicio de la aplicación, únicamente ejecutando la actualización de alguna función o procedimiento mediante un script de base de datos.

2.2.1.1 Introducción a la Operación de la Compañía y Conceptos Generales.

Inicialmente el contrato con Merill Lynch, consistía en atender cualquier problema con la aplicación (YAX), dando soporte primario a producción, esto es, atender fallas dando una solución pronta a los usuarios. Paralelamente, la aplicación MIVIRO seguiría funcionando en caso de alguna contingencia dando respaldo a la aplicación. Con el paso del tiempo, las funciones iniciales del proyecto, cambiaron de dar soporte de primer nivel a tomar el control total de la aplicación, dar mantenimiento, construcción e integración de nuevos requerimientos dentro de la misma.

Glosario de Términos utilizados.
Antes de comenzar este capítulo, no se tiene como finalidad explicar ningún proceso de negocio en un sentido altamente técnico, sin embargo, para esta parte de mi experiencia, se deben definir algunos de los términos que comúnmente serán utilizados a lo largo de este apartado, debido a que se mencionan constantemente y el no tener un concepto de los mismos pueden causar confusión.

- **Orden (Order)**.- Una Orden se define como toda aquella operación que involucra la compra o venta de acciones cuya características principales son: un folio único, cliente al cual se le acredita la operación, sentido de la operación, títulos o acciones (securities), precio (precio fijo definido o precio de mercado en caso que no se asigne alguno) que se desea comprar o vender, cantidad de títulos que se desean comprar/vender, cantidad asignada (el número de títulos que se han conseguido comprar o vender), cantidad por asignar (el número de títulos que faltan para completar la orden), tipo de orden (Regular, Global, Al cierre, Picos, Lotes, Paquetes, etc.). Todas las Órdenes se generan dentro de una casa de bolsa para ser enviadas al organismo regulador, quien se encarga de recibirlas y una vez recibidas, comienza el proceso de ejecución de las mismas durante el día.

- **Sentido de una Orden**.- Se refiere a si la Orden es de compra o venta (Buy-Sell), forzosamente las ordenes deben tener un sentido cuando se crean.

- **Títulos (Securities)**.- Representa los valores que las compañías ofertan mediante una identidad única dentro del mercado de valores, los títulos se componen de un emisor (emisor) y una serie (serie), por ejemplo, la compañía de TELMEX, dentro del mercado accionario oferta sus títulos como:

  TELMEX A

  O bien la compañía YAHOO como:

  YHOO B

en donde la emisora es TELMEX/YHOO y las series son A y B respectivamente.

- **Ejecución o Hecho (Execution)**.- Se define como la cantidad comprada/vendida de una Orden dentro del mercado accionario, dicha cantidad puede ser menor o igual a la cantidad total solicitada en la Orden teniendo que por cada cantidad ejecutada, la casa de bolsa emisora de la orden recibe una notificación con dicha cantidad que ha sido comprada o vendida. La notificación contiene un folio único de ejecución, cantidad ejecutada, sentido de la ejecución, folio de la orden (opcional)*, precio de la ejecución.

- **Asignación (Allocation)**.- Dentro de cada casa de bolsa, una asignación representa la relación entre una Orden y sus Ejecuciones, ya que contiene tanto el folio de la orden como el de la ejecución. Por lo tanto, las asignaciones sirven como un mecanismo de rastreo, en donde se puede revisar el comportamiento que ha tenido cierta orden en cuanto a la cantidad de títulos que han sido comprados/vendidos durante su operación.

Los conceptos anteriores se pueden explicar mediante un ejemplo sencillo, que permite entender de una mejor manera, la principal función de los que se hace dentro de las casas de bolsa a nivel negocio, a continuación se enumera el flujo básico para un mejor entendimiento:

*En algunas ocasiones, dentro de la BMV, se envían ejecuciones que no tienen una Orden en particular, es decir no se generó ninguna orden previa, sin embargo, este tipo de ejecuciones son conocidas como oportunidades de mercado debido a que contienen precios atractivos para los inversionistas, en el caso de que la casa emisora contenga Ordenes que hagan match con la ejecución recibida, entonces, la cantidad de dicha ejecución se asigna proporcionalmente a esas Órdenes, en caso de que no exista ninguna Orden, entonces se crea una orden especial conocida como Paquete, el cual se puede desglosar en diferentes subórdenes con distintas cuentas entre sí, dependiendo del interés de los clientes para aprovechar dichas oportunidades de mercado. Los paquetes se tratan de manera similar a una orden y reciben ejecuciones durante el día hasta que la cantidad es completada en su totalidad.

1) La casa de bolsa genera las siguientes Órdenes, las cuales son enviadas correctamente a la BMV:

Folio Orden	1
Sentido	Compra
Cantidad	10,000
Emisor	YHOO
Serie	*
Precio	12.5
Cantidad Asignada	0
Cantidad por Asignar	10,000

Folio Orden	2
Sentido	Venta
Cantidad	5000
Emisor	CEMEX
Serie	B
Precio	37
Cantidad Asignada	0
Cantidad por Asignar	5000

Fig. 2.3 Órdenes Enviadas a la BMV

2) La BMV, recibe correctamente las órdenes solicitadas y posteriormente envía las siguientes ejecuciones a la casa de bolsa:

Folio	1
Sentido	Compra
Cantidad	5000
Emisor	YHOO
Serie	*
Precio	10
Folio Orden	1

Folio	2
Sentido	Compra
Cantidad	2500
Emisor	YHOO
Serie	*
Precio	9
Folio Orden	1

Folio	3
Sentido	Venta
Cantidad	4500
Emisor	CEMEX
Serie	B
Precio	39.8
Folio Orden	2

Fig. 2.4 Ejecuciones Recibidas desde la BMV

3) Una vez recibidas las ejecuciones, se realiza el proceso de asignación, quedando de la siguiente manera.-

ORDER 1

Folio Orden	1
Sentido	Compra
Cantidad	5 000
Emisor	CEMEX
Serie	B
Precio	12.5
Cantidad Asignada	**7500**
Cantidad por Asignar	2500

Detalle de la Asignaciones

Folio Orden	1
Folio Ejecución	1
Cantidad	5000

Folio Orden	2
Folio Ejecución	3
Cantidad	4500

ORDER 2

Folio Orden	2
Sentido	Venta
Cantidad	5 000
Emisor	CEMEX
Serie	B
Precio	37
Cantidad Asignada	4500
Cantidad por Asignar	500

Detalle de la Asignación

Folio Orden	1
Folio Ejecución	2
Cantidad	2500

Fig. 2.5 Proceso de Asignación

En cuanto la orden se complete con más ejecuciones y la cantidad por asignar se convierta a 0 entonces, la orden queda completada y se dice que es una orden cerrada. Obviamente existen muchas reglas alrededor de este proceso, que tienen que ver con el tipo de orden, la hora en que se recibe la ejecución, el precio de la ejecución dado que no se puede vender a un menor precio del que se oferta, ni comprar a mayor precio del que se busca. Como se observa en las tablas anteriores, en la primer orden de Compra, el precio de compra al que se envió fue de 12.5 para tratar de comprar 10,000 títulos de YAHOO, las ejecuciones recibidas tuvieron precios inferiores, por lo que se pudo lograr las asignaciones correspondientes, en el caso de la venta (segunda orden), se envió un precio de 37 y la ejecución recibida fue por un precio de 39.80 lo cual es una señal positiva debido a que se está vendiendo a un precio mayor del que se envió con la orden el cual fue de 37.00.

2.2.1.2 Departamentos Internos dentro de la Compañía.

Los siguientes departamentos descritos, son aquellos en los cuales se han desarrollado los módulos principales de la aplicación, para sus operaciones diarias:

- **Trading**.- Departamento encargado de realizar la compra y venta de acciones mediante la creación de Órdenes para las distintas carteras de clientes. Deben evitar los riesgos durante las operaciones del mercado y generar la mayor ganancia mediante el manejo responsable en las decisiones de inversión.

- **Middle-Office**.- Este departamento apoya directamente al departamento de Trading durante la operación diaria, entre sus funciones principales están la entrega de reportes sobre el comportamiento de las operaciones durante el día, configuraciones a catálogos del sistema como modificación borrado o inserción de nuevas cuentas, emisoras, comisiones, ejecución de procesos de negocio, etc.

- **Back-Office**.- Básicamente este departamento comienza sus labores durante el cierre de operaciones, ejecutando reportes internos sobre el comportamiento de las operaciones durante el día, como también reportes a los distintos clientes que así lo requieran, realizan la ejecución de respaldos de las operaciones mejor conocido como procesos “Batch”.

2.2.2 Resumen de las Actividades Realizadas.

Durante los primeros meses del proyecto, se comenzó en actividades de soporte y configuraciones al sistema, mientras se realizaba el análisis de la arquitectura del mismo, para comenzar con la creación de nuevos procesos de negocio y a la vez poder capacitar a cualquier persona del equipo u otros equipos para realizar cualquier actividad, evitando la dependencia a un solo grupo de trabajo. Parte de este análisis, consistió en entender completamente la arquitectura del sistema, para que cada nuevo desarrollo la respetara, dejando entonces, un sistema consistente y de fácil mantenimiento.

A continuación, se muestra el diagrama de la arquitectura del sistema:

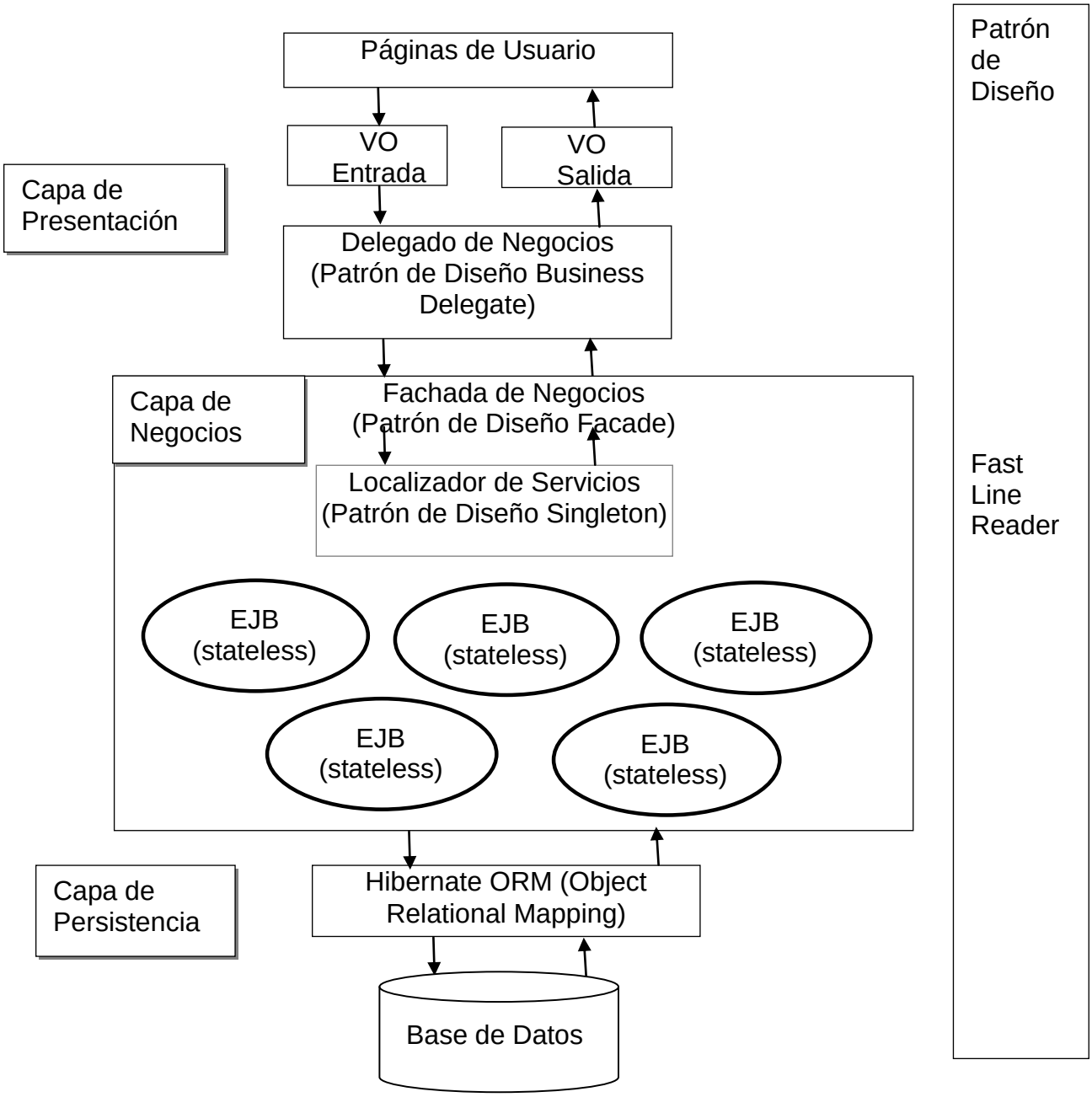


Fig. 2.6 Arquitectura Multicapa del Sistema

Como se observa en el diagrama anterior, la aplicación se compone de las siguientes capas (Multicapa) utilizando varios patrones de diseño dentro de su arquitectura, en seguida se analiza cada capa junto con una descripción detallada.

2.2.2.1 Capa de Presentación.

La primer capa del sistema, es la denominada de presentación, en donde se encuentran las páginas web en donde el usuario interactúa invocando los servicios del sistema, por ejemplo, operaciones de alta, cambios y borrado, consulta de información, llamado a procesos de negocio, ejecución de reportes, etc., ésta capa se diseñó en dos tecnologías, Macromedia Flash y “Tapestry Framework”, un marco de trabajo enfocado a crear aplicaciones web altamente acoplado con el lenguaje de programación JAVA.

2.2.2.1.1 Capa de presentación en Macromedia Flash.

Las páginas utilizadas con tecnología FLASH, fueron diseñadas para evitar el “refrescado” de las mismas producido generalmente durante una petición HTTP (REQUEST/RESPONSE) mostrando información en tiempo real, la principal interfaz gráfica del sistema construida bajo esta tecnología es el “Monitor de Ordenes” que sirve para generar las Ordenes durante el día y mostrarlas en forma de tabla junto con el detalle de las asignaciones que han sufrido durante el día, por lo que cada que se recibe una ejecución desde la BMV, entonces, la cantidad asignada y cantidad por asignar va cambiando de manera instantánea dentro del monitor y los usuarios pueden ver en tiempo real como se va comportando el mercado. Otras UIs basadas en FLASH fueron construidas dentro del sistema, por ejemplo el US BOOK (modulo para realizar reportes), la interfaz de Ordenes parciales (envía órdenes a la BMV de manera parcial durante lapsos de tiempo periódicos), asignación de paquetes etc.,

Cabe mencionar, que actualmente, el uso de **AJAX (**Asynchronous JavaScript And XML**)** resuelve el dilema del refrescado de la pantalla mediante instrucciones que envían la información de un formulario mediante un envío http(s) de tipo POST/GET al servidor, sin necesidad de ejecutar un refrescado de la página, además de que existen hoy en día muchos frameworks que implementan Ajax en sus controles entre los que se pueden mencionar Jquery, Dojo, JackBe, Prototype, etc. para construir páginas web con alto performance y poca codificación del lado del programador, durante el proyecto se cuestionó el **no** uso de esta tecnología, pero el equipo de construcción original, argumento que FLASH era más popular, teniendo más soporte que algún otro framework basado en AJAX, debido a su gran utilización en millones de sitios web, siendo esto lo que propicio su uso sobre otras tecnologías.

Durante el proyecto, parte de las tareas primarias, fue la de realizar mantenimiento a algunas interfaces construidas en tecnología FLASH, parte de este mantenimiento incluye:

- Modificación del look & feel de alguna UI.- Cambios directos en la interfaz, como el color/tamaño de etiquetas, mover de posición u ocultar controles, cambiar tipos de letra, colores de fondo, etc.

- Modificación o construcción de nueva funcionalidad.- En algunas ocasiones, estas UIs requerían funcionalidad adicional, por ejemplo, dentro del monitor de ordenes se mostraban diferentes filtros a elegir, cada filtro mostraba diferentes resultados al usuario, conforme nuevos tipos de Ordenes surgían o desaparecían, entonces más filtros fueron requeridos o eliminados, por lo que, se tenía que agregar dicha funcionalidad.

- Deshabilitar funcionalidad.- Con el paso del tiempo, algunos módulos realizaban funciones que ya no eran requeridas, por lo tanto parte de las tareas, era eliminar dicha funcionalidad sin afectar otras, por lo que se tenía que hacer un análisis profundo de que no se vería afectado ningún otro modulo debido al principio de re-utilización.

- Creación de nuevas UI basadas en FLASH.- No fueron muy solicitadas, sin embargo, se tuvo la necesidad de crear nuevos interfaces utilizando FLASH, para esto, se tiene que realizar la nueva UI utilizando el entorno de desarrollo Macromedia, compilarla para obtener el archivo con extensión .SWT, hacer pruebas de funcionalidad mediante peticiones HTTP al servidor de Aplicaciones y una vez verificado su correcto funcionamiento, agregarla al proyecto Web del sistema (archivo .WAR) mediante la compilación del todo el proyecto.

2.2.2.1.2 Capa de Presentación con el Framework Tapestry.

El resto de la capa de presentación (la mayor parte del sistema) está desarrollada utilizando el framework Tapestry, cabe mencionar que incluso todas las interfaces de usuario hechas en FLASH están montadas sobre una página Tapestry, este framework está altamente acoplado con el lenguaje java e internamente utiliza el patrón MVC (Model-View-Controller) para su desarrollo en donde:

- **Model**.-Se implementa mediante una plantilla XML, con tags definidos para que sean entendidos por este framework, cada tag define un elemento dentro de la vista, por ejemplo.-

```
<id=”button” type=”submit” hidden=”no” name=”myButton”>
```

representa un botón de tipo formulario, que enviará datos al servidor de aplicaciones mediante una petición http.

- **Controller**.- Se representa mediante una clase java, la cual, contiene toda la lógica asociada a los elementos de la página, por ejemplo, si se crea una caja de texto dentro del modelo, dentro de la clase se debe escribir el código asociado a dicha caja de texto, de no ser así, no hay forma de interactuar con el valor asociado, para ser más claro, si se crea una caja de texto dentro del modelo, cuyo nombre es “myText”:

```
<id=”text” type=”normal” hidden=”no” name=”myText” max=100>
```

Entonces, se deben crear sus propiedades (getters/setters) dentro de una clase java para interactuar con los valores asociados a la misma:

Fragmento Clase Java:

```
private String myText;
public String getMyText(){
     return myText;
}
public void setMyText(String newText){
    this.myText=newText;
}
```

- **View**.- Representada por un archivo con extensión .HTML que despacha la vista, aquí se cargan los elementos de la pantalla por su ID definidos dentro del modelo (archivo tipo XML) para mostrarse dentro de la interfaz gráfica.

Por ejemplo, si se define un botón dentro del modelo con un ID=”myButton”, entonces se debe mandar utilizar dicho nombre dentro de la vista:

```
<html>
    <body>
       <p>Mi boton abajo:</p>
       <class id=”myButton”>
    </body>
</html>
```

De acuerdo a lo anterior, para cada nueva pantalla requerida, se tienen que generar los 3 archivos manualmente, editar el código necesario en cada uno, subirlos al servidor dentro de los archivos Web y reiniciar el servidor para incluya los archivos creados o modificados.

El proceso de aprendizaje de este framework fue más que obligatorio debido a que es utilizado dentro de toda la capa de presentación de la aplicación, a la larga se hizo familiar su uso, tan familiar que incluso se detectaron y documentaron algunos bugs internos de este framework. Por mencionar uno de los más recurrentes y que en un principio costo varios días conocer su causa, fue que si se colocaba un botón de tipo “submit” antes de alguna caja de texto o combo de datos del cual se tenía que obtener su valor para enviarlo dentro de una petición http, entonces, internamente se tomaba el valor anterior (o nulo), enviando siempre un valor desfasado al servidor, por lo que el botón de “submit” siempre tenía que ir al final de todos las controles, de donde se quisiera obtener un valor para enviarse dentro de cada petición o colocar una caja de texto oculta, para sincronizar los valores evitando el desfasamiento.

2.2.2.1.3 Seguridad de Autorización/Autenticación a la Capa de Presentación.

Autenticación
Cada página creada, necesitaba autenticarse dentro del sistema, por lo que una vez creada, se daba de alta el nombre de la UI dentro de una tabla de base de datos de seguridad junto con su URL asociada, esto se realiza mediante un script de base de datos, una vez dada de alta, también se agrega a un archivo XML global que sirve en tiempo de ejecución para crear el menú principal del sistema y que al detectar la nueva UI, revisa que exista en la base de datos para mostrarla.

Autorización.
La autorización a una página se controla mediante el uso de “ROLES” de usuario, todo controlado dentro de la base de datos, mediante el uso de tablas donde se dan de alta las UIs, mencionadas anteriormente y los roles que pueden acceder a la misma, por lo que una vez que un usuario trata de abrir alguna UI, se compara su rol contra el asociado a la UI para verificar si tiene permiso para visualizarla.

2.2.2.1.4 Patrones de Diseño Aplicados a la Capa de Presentación.

- **Patrón de Diseño Value Objects** de Entrada/Salida.

Value Object de Entrada.

Un Value Object de entrada (aquí en adelante identificado como VO), se define como un objeto que contiene información que se envía por las distintas capas de un sistema (desde la Capa de Presentación hasta la Capa de Base de datos o Capa de Negocios), este tipo de objetos deben ser inmutables durante su transporte, es decir, que los valores que contienen no deberían cambiar durante el proceso y en caso que esto sucediese, las buenas prácticas de programación indican que el objeto debe cambiarse por otro nuevo, en lugar de mutar los valores que contiene, por ejemplo, para ejecutar una operación CRUD (Create/Update/Delete) en el catálogo de Cuentas, generalmente desde la UI diseñada para este propósito, el usuario da de alta lo siguientes valores: Nombre de la Cuenta, número asignado a la cuenta, comisión etc., utilizando cajas de textos u otros controles. Por lo tanto, dicha UI podría usar el siguiente VO de entrada asociado:

```
public class AccountValueObject{
    private Long ID;
    private String accountName;
    private String acountNumber;
    private BigDecimal accountCommission;
    private Boolean isActive;
   .
   .
   /*Get/Setters de las propiedades*/
}
```

Por lo tanto, cuando se envía la información al servidor para persistir una nueva cuenta, editarla o borrarla, se ajustan las propiedades del VO con los valores de las cajas de texto dentro de la UI que utiliza el usuario para dar de alta las cuentas en el sistema y finalmente, esta información es enviada al proceso de negocio (capa de negocios) involucrado.

Value Object de Salida.

Un VO de Salida, es un objeto similar al de entrada, pero con la diferencia que contiene información sobre los resultados que se obtuvieron durante la ejecución de algún proceso de negocios dentro de la aplicación, es decir, viaja de regreso desde las capas más internas del sistema hacia la capa más externa (generalmente la capa de presentación), la información que puede contener puede ser muy variada, desde un valor Booleano que indica éxito o problema resultante de la operación o hasta incluso una lista de valores que representan el resultado de una consulta a la base de datos, por ejemplo, tomando el caso anterior, si una nueva cuenta se agrega al sistema correctamente, entonces, el VO de salida y que “viaja” de regreso hasta el lugar donde fue invocado (la UI de usuario), podría contener la siguiente información:

```
public class ProcessResultValueObject(){
    private Set processInfo; //Información con resultados del algún proceso
    private Set processError;//Información de Errores
    .
    .
    /*Verificar si existen errores */
    public Boolean isError()
    {
       return !processError.isEmpty();
    }
    public Boolean isProcessSucess()
    {
      return !processInfo.isEmpty() &&
            processInfo.contains("SUCESS");
    }

}
```

Como se observa, cuando se envía el VO de salida, se puede incluir dentro de él, en una colección (o cualquier otro tipo de objeto) la información resultante del proceso de negocio invocado, que como se mencionó anteriormente puede ser muy variada o también en su caso, los errores que se pudieron haber lanzado, por lo tanto, si un proceso de negocio ha sido invocado desde la capa de presentación (página web), el VO de salida que viaja de regreso puede ser analizado para obtener información que se muestra al usuario que mando llamar el proceso, por ejemplo, si se ha insertado una nueva Cuenta dentro del sistema correctamente, quizá pueda ser útil enviar en el VO de salida que ID único se generó en la base de datos para la nueva cuenta insertada y mostrar al usuario un mensaje descriptivo dentro de la UI como:

"The Account has been saved success with the ID# 486."

O en el caso de un error, entonces se puede mostrar al usuario la causa del mismo:

"The new Account cannot be created. Caused: The account name is already exists…please verify"

- **Patrón de Diseño Delegado de Negocios (Business Delegate)**

Cuando se envía información desde alguna UI para ejecutar algún proceso de negocio , existe una clase en particular encargada de conectar el Front-End (Capa de Presentación) hacia las siguientes capas del sistema, como se puede observar en el diagrama de arquitectura mostrado al inicio del capítulo, existe por lo tanto una clase "Delegado de Negocios" por cada subsistema, el cual se encarga de recibir cualquier petición sobre algún proceso de negocio a ejecutarse y por lo tanto, enviarlo hacia la siguiente capa del sistema, en este caso, la Capa de Negocios.
Suponiendo que se tiene el siguiente subsistema para dar de alta/modificación/baja las distintas Cuentas con las que el sistema actualmente trabaja u otras actividades administrativas como obtener un reporte de las cuentas activas, las inactivas o incluso algo más complejo como modificar la comisión de alguna cuenta y después de esto, enviarle un correo electrónico al titular de la misma informando del ajuste:

SUBSISTEMA ACCOUNTS

El patrón de diseño Business Delegate es utilizado como puente entre la capa de Presentación hacia la siguiente capa dentro de la arquitectura (Capa de Negocios), quizá un uso más frecuente de este patrón reside en el llamado directo de métodos de negocios, sin embargo, su implementación en el sistema fue más como una clase intermedia para conectar todas las llamadas a procesos de negocios por cada subsistema, las cuales después llamarían a los procesos de negocios mediante el uso de otras clases.

A continuación se muestra un ejemplo del patrón para el subsistema de Cuentas, como se observa, los métodos por lo general reciben un VO de Entrada como parámetro y regresan como valor de retorno un VO de Salida con la información del resultado del proceso de negocio que se mandó ejecutar, a continuación se ejemplifica un fragmento de un típico Delegado de Negocios:

```
public Class AccountBusinessDelegate{
         //VO de Salida                                    //VO de Entrada
 public  ProcessResultValueObject  saveOrUpdateAccount(AccountValueObject  inputVO)
{
  ProcessResultValueObject outputVO=new  ProcessResultValueObject();
  //.....Alguna llamada a la capa de negocios.
  outputVO=AccountFacadeBean.saveOrUpdateAccount(inputVO);
  return  outputVO;
}
 //Otros Métodos dentro del Delegado del Negocios
 public ProcessResultValueObject deleteAccount(AccountValueObject inputVO){...}
 public ProcessResultValueObject getActiveAccounts(){...}
 public ProcessResultValueObject changeCommision(AccountValueObject inputVO){...}
}
```

Analizando el fragmento de código anterior, tenemos que el nombre de la clase se compone con el nombre del subsistema Account junto con la palabra BusinessDelegate (Delegado de Negocios), que aunque no es mandatorio utilizar este nombre, siempre es una buena práctica para poder identificarlo dentro del código.

El siguiente punto de análisis es el método de ejemplo:

```
public ProcessResultValueObject saveOrUpdateAccount(AccountValueObject inputVO)
```

Como se observa, el método debe ser público (o protected) para que sea visible ya que servirá como un punto de entrada para que pueda ser llamado por otras clases, el valor de retorno es un objeto de tipo `ProcessResultValueObject` el cual, como se explicó anteriormente de acuerdo a la arquitectura, es un VO de salida que contendrá información acerca del proceso de negocio que se manda ejecutar, después se tiene el nombre del método, el cuál es meramente explicativo: saveOrUpdateAccount (Salva o Actualizar), por lo tanto, quiere decir que dicho método es encargado de crear o actualizar alguna cuenta en particular.
El parámetro de entrada al método, es un objeto de tipo `AccountValueObject` el cual, de acuerdo a la arquitectura, es un VO de entrada que contiene información sobre la cuenta que se desea dar de alta/modificar, dicho objeto es enviado a su vez al método de negocio correspondiente como un parámetro de entrada, que como ya se explicó anteriormente, este tipo de objetos son creados para viajar entre las diferentes capas del sistema.

Para finalizar, solo como observación, no todos los métodos de negocio a ejecutar, requieren un VO de entrada, como se observa en la clase ejemplo anterior, el método:

```
public ProcessResultValueObject getActiveAccounts(){...}
```

no recibe ningún parámetro de entrada, esto debido a que el llamado de este proceso no lo requiere. Seguramente al ejecutarse el proceso de negocio asociado, se obtienen todas las cuentas activas, por lo que al llegar a las capa internas del sistema, se realizará una consulta como:

```
"Select Account_Number FROM ACCOUNT Where ACTIVE=1"
```

Por lo que no es necesario incluir información de entrada para poder ejecutar el proceso, lo que es seguro, es que el proceso de negocio, enviará un VO de salida incluyendo el resultado de la consulta dentro de alguna colección y dicha colección será enviada como parte de su información, la cual cuando sea recibida en la capa de presentación será analizada y desplegada en la UI del usuario mediante algún control.

- **Patrón de Diseño FastLineReader.**

En el diagrama de la arquitectura del sistema arriba expuesto, existe un patrón de diseño que como se observa, viaja desde la capa de presentación hasta la capa de base de datos de manera directa, este tipo de patrón es conveniente cuando se desea agregar performance a una aplicación, debido a que se comunica directamente con la base de datos para obtener algún resultado, el cual en la mayoría de los casos, son solamente consultas que regresan resultados, además de que en la mayoría de las bases de datos, se puede optimizar aún más este tipo de consultas haciendo que los resultados tengan un cursor de solo lectura y de solo avance, por ejemplo, en el ejemplo anterior arriba descrito en donde se menciona un método para obtener todas las cuentas activas, se podría aplicar este patrón de diseño para optimizar la carga de trabajo a la aplicación simplemente consultando la base de datos, mediante una conexión directa y aplicando la misma consulta definida anteriormente:

```
"Select Account_Number FROM ACCOUNT Where ACTIVE=1"
```

Si se consulta directamente a la base de datos se obtiene un mejor performance, primero por lo simple de la operación, sin embargo se debe tener cuidado de que las operaciones a ejecutar en la base de datos, sean en su mayoría solo consultas para obtener información, es decir, se debe evitar realizar operaciones de negocio que alteren información, este patrón por lo tanto, funciona para operaciones simples que no conlleven algún tipo de transacciones complejas dentro de la base de datos.

2.2.2.2 Capa de Negocios.

Como su nombre lo indica, dentro de esta capa, se codifican todos los procesos de negocios de un sistema, es decir, aquí se debe colocar lógica compleja, algoritmos necesarios para algún proceso en particular, código que requiere tener un control en cuanto a seguridad, transaccionalidad, validaciones, etc.

Por ejemplo, para realizar una Asignación a una Orden en particular después de recibir una Ejecución por parte de la BMV, se requiere una lógica de negocios compleja que incluye interactuar con niveles de transacción, validaciones, seguridad, interacciones con otros subsistemas e incluso con sistemas externos como los sistemas de la BMV.

Una mala práctica de programación e incluso de arquitectura, es el dejar expuesto operaciones de alta complejidad a nivel de cliente, por ejemplo, si se quisiera programar un módulo que realiza operaciones de ALTA/BAJA/CAMBIO dentro de un sistema, para persistir cierta entidad de negocio y que esta lógica se codifique dentro de la capa de Presentación, a pesar de que esto es posible utilizando tecnologías como JSP (Java Server Pages), PHP (Hypertext Preprocessor), JSF (Java Server Faces), no es una buena práctica e incluso debería ser rechazado si en un análisis del código se encontrara lógica de negocios expuesta al nivel de capa de presentación (lado del cliente). Los procesos de negocios de un sistema deben ser separados por varias razones, una de ellas es por seguridad, debido a que si se expone el código de algún proceso de negocio importante o sensible para la operación diaria y se codifica en el lado del cliente, dicho código queda expuesto a cualquier persona, pudiendo ser modificado impactando negativamente y causando afectaciones que pondrían en riesgo la integridad tanto de datos como de los resultados que se esperan obtener normalmente.
Otra buena causa para separar procesos de negocios es el aspecto enfocado a arquitectura de software, dado que siempre beneficia tener un código ordenado que permita su fácil identificación para las personas involucradas en el proyecto o las nuevas que se integren dentro del tiempo de desarrollo, por lo que un código mezclado siempre es más difícil de controlar, modificar y aumenta la curva de aprendizaje para las personas que se integran a los equipos de desarrollo.

2.2.2.2.1 Patrones de Diseño aplicados a la Capa de Negocios.

- **Patrón de Diseño Facade (Fasade) o Fachada de Negocios.**

Este patrón de diseño sirve como punto de entrada para ejecutar un proceso de negocio, se tiene una fachada para cada subsistema, la que generalmente expone métodos públicos (o protected) que sean visibles por cualquier otra clase, estos métodos son el punto de entrada para ejecutar un proceso de negocio ocultando la implementación de los mismos, por ejemplo, si se requiere dar de alta una nueva Cuenta, entonces se puede crear una clase "fachada" dentro del subsistema de CUENTAS pudiendo definirse de la siguiente manera:

```
public class AccountsFacadeBean{
       public ProcessResultValueObject createAccount(AccountValueObject inputVO)
       {
         try{...
           return AccountBusinessLocator.getInstance().saveAccount(inputVO);
         ...}
       }
       public ProcessResultValueObject updateAccount(OrderValueObject inputVO) {...}
       public ProcessResultValueObject deleteAccount(OrderValueObject inputVO) {...}
}
```

Como se observa en el ejemplo de la clase anterior, se han creado algunos métodos públicos que reciben como parámetro de un VO de entrada y regresan un VO de salida ya explicado anteriormente, por lo tanto esta clase se puede llamar en cualquier momento para ejecutar un proceso de negocio, teniendo que la clase que la manda llamar comúnmente es el Delegado de Negocios explicado anteriormente, aunque podría ser incluso otra clase fachada de otro subsistema u otra de un sistema externo.
Analizando el fragmento de código de la clase anterior, tenemos algunos puntos importantes a destacar, uno de ellos es el nombre de la clase `AccountsFacadeBean` que como se observa, se compone del nombre del subsistema (En este caso Accounts), la palabra Facade, lo cual indica que es una fachada (no obligatorio, pero buena práctica hacerlo) y finalmente la palabra "bean", lo que indica que se refiere a una entidad de negocios correspondiente a un EJB (Enterprise Java Bean) del estándar J2EE (Java 2 Enterprise Edition), por lo que para la arquitectura de este sistema, las fachadas de los subsistemas fueron implementadas mediante EJBs, más adelante se hará una breve definición de la arquitectura de software mediante Java2EE EJB.

Otro elemento importante, está dentro del método de ejemplo denominado: `createAccount` que como se observa, se manda llamar a través de otra clase denominada "**`AccountBusinessLocator`**":

```
AccountBusinessLocator.getInstance().saveAccount(inputVO)
```

Lo anterior nos permite conocer por el nombre de esta clase y por la forma en que se instancia, que en cada método de nuestra clase fachada, se manda llamar a la clase auxiliar **`AccountBusinessLocator`** para localizar y ejecutar el proceso de negocio requerido para dar de alta una nueva Cuenta, en este caso "saveAccount", por lo tanto, tenemos una mezcla de 2 patrones distintos, por un lado el patrón de diseño Facade que funciona como punto de entrada a un Subsistema y por el otro una clase que implementa el patrón de diseño "Singleton" encargada de buscar y ejecutar los servicios* que se se ejecutan dentro de los métodos de la clase fachada, en el siguiente apartado se analiza este patrón de diseño.

- **Patrón de Diseño Singleton para Localizar de servicios.**

Este patrón, es uno de los patrones más utilizados dentro de la programación, su principal característica es que crea la instancia de una clase en memoria una sola vez durante toda la vida del sistema, es decir, se crea una instancia de cierta clase tan solo una vez. Lo anterior es útil para tener alguna utilidad general que se expone en un grupo de métodos y que siempre están disponibles y listos para su uso por cualquier otra clase dentro del sistema, al tener los métodos disponibles siempre, se obtienen muchos beneficios, entre los cuales se destaca el performance, debido a que si otra clase utiliza los mismo métodos, no se requiere otra instancia de la clase simplemente se utiliza la ya creada anteriormente. Para lograr su única instancia (singularidad), este tipo de clases utilizan un constructor "privado" que evita que la clase se pueda instanciar mediante una operación "new" en cualquier parte del código, en lugar de esto, simplemente exponen un método público y estático para su utilización, en dicho método, si la instancia de la clase no existe entonces se crea y en caso contrario se utiliza la instancia existente:

* El término "servicio", se refiere dentro de este contexto, a la ejecución de algún proceso de negocio y no propiamente a una arquitectura orientada a Servicios construida por ejemplo con el framework de Spring o una arquitectura SOA implementada con algún Framework para BPEL como Oracle Business Process Execution Language .

```
public class AccountBusinessLocator{
   private static AccountBusinessLocator myClass;
   private  AccountBusinessLocator(){} //Constructor Privado
   //Método Auxiliar para hacer uso de la clase
   public static AccountBusinessLocator getInstance()
   {
         if (myClass==null)
         {
           return new OrderBusinessLocator ();//Se instancia primera vez
         }else{
           return myClass //Si ya existe se regresa la instancia creada
         }
   }
  //Otros Métodos...
  public static final ProcessResultValueObject saveOrder(OrderValueObject inputVO){
       return ...
   }
public static final ProcessResultValueObject updateOrder(OrderValueObject inputVO){
      return ...
   }
}
```

El patrón Singleton por lo tanto, sirve como una clase auxiliar que localiza los servicios que son utilizados dentro de los métodos de la clase fachada descrita anteriormente. Como se observa en el fragmento de código arriba descrito, los métodos se declaran Estáticos y Finales (el estudio de los modificadores de acceso en JAVA esta fuera del alcance de este informe), esto con la finalidad de que puedan ser llamados directamente con el nombre de la clase de la forma:

```
MyClase.MiMetodo();
```

por lo que en el ejemplo de la clase fachada, se puede observar el llamado a su localizador de servicios de la siguiente forma:

```
AccountBusinessLocator.getInstance().saveAccount(inputVO);
```

Como mención aparte, a pesar de que un localizador de servicios de un subsistema podría ser importado en otro y así obtener los servicios de ese subsistema, esto no está permitido dentro de la arquitectura y no es aceptado, por lo tanto, si por ejemplo el subsistema de Ordenes, desea obtener la lista de las Cuentas activas, entonces, debe acudir al subsistema de cuentas a través de su clase fachada y ejecutar el proceso de negocio encargado de obtener las cuentas requeridas y no instanciar la clase Singleton asociada.

2.2.2.2.2 Enterprise Java Beans (EJB's).

La lógica de negocios quedo totalmente dentro de la especificación Java EJB (Enterprise Java Beans), de manera introductoria, un EJB es una porción de código que se ejecuta dentro de una transacción dentro de un servidor de aplicaciones (del lado del servidor) con la capacidad de ejecutarlo (por ejemplo Jboss Server, Bea WebLogic Server, WebSphere Server, GlassFish Server, etc.), la ejecución de código dentro de un servidor mediante el uso de EJB tiene algunas analogías con el típico nivel de transaccionalidad que se encuentra en una base de datos, en donde comúnmente si se desea ingresar un grupo de registros, puede entonces abrirse una transacción:

```
sql> Begin Transaction
                ....Operaciones a la base de datos....
```

después se realizan las operaciones que se desean, si todo va bien y los registros se insertaron correctamente entonces se cierra la transacción abierta previamente mediante una instrucción:

```
sql>Commit Transaction
```

pero si se produce un error, durante el proceso, entonces se ejecuta (explicita o implícitamente) una instrucción:

```
sql>RollBack Trasaction
```

y todos los cambios hechos regresan al estado inicial a como se encontraban antes de abrir la transacción, esto para evitar inconsistencia de datos y adoptar el modelo ACID de una transacción, esto es Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español, de manera similar, un EJB funciona, pero desde un panorama más general, es decir, no necesariamente se encapsula una operación que modifica la base de datos, sino que puede haber más operaciones que involucran entre otras cosas, ejecución de algoritmos, “n” modificaciones en base de datos, interacción con otros sistemas de software, etc., por lo que, un EJB se encuentra a un nivel más general abarcando todo un grupo de operaciones en una única unidad de trabajo, en caso de algún error, todas las variables, modificaciones a la base de datos, etc. regresan a su estado inicial.

Existen 3 tipos básicos de EJB dentro de la especificación J2EE, con estado (statefull) el cual es capaz de mantener una conversación con el cliente que lo ocupa de manera similar a como una sesión HTTP Web mantiene conversación con un cliente que accede mediante un navegador hasta que esta expire o sea explícitamente cerrada. El segundo tipo de EJB es sin estado (stateless) lo que se conoce comúnmente como un EJB “tonto”, es decir, se corre la porción de código invocada y no existe ninguna relación más allá de lo que suceda, es decir la llamada al método termina cualquiera que sea el resultado obtenido (éxito o error), finalmente se tiene el EJB Singleton, el cual es idéntico al Stateless, con la diferencia que solo una y solo una instancia del mismo durante la vida del sistema (Singleton Pattern).
Durante el proyecto únicamente se utilizó el tipo EJB sin estado para toda la capa de negocios.

Transaccionalidad. Se define una transacción generalmente a una unidad de trabajo indivisible que puede finalizar en 2 posibles estados: Commit (Cometer) o Rollback (Retroceder), si dentro de su ejecución ocurre alguna situación que afecte la integridad de la misma, entonces se debe anular y volver a un estado inicial antes de que el proceso comenzó, esto con la finalidad de evitar inconsistencia de datos, más si por el contrario, la unidad de trabajo termino correctamente, entonces todos los cambios realizados dentro de la misma deben persistir y ser consistentes.
En una transacción de negocios, no necesariamente implica un cambio a la base de datos, incluso puede que no se afecte a la base de datos, por ejemplo supongamos que un proceso de negocios durante su ejecución afecta una variable global estática que se utiliza durante el funcionamiento del sistema, dicha variable es un contador numérico, definido en una clase del sistema inicializado a 0:

```
public static Integer counter=new Integer(0);
```

Cierto proceso de negocio incrementa en uno dicho contador:

```
public void someBusinessMethod()
{
   Counter.counter +=1; //El valor de la variable se incrementa
   .
   .
   .
}
```

Durante cada ejecución, del método “someBusinessMethod” el contador se incrementa en uno, por lo que si este método es invocado 10 veces durante el día, en esta variable se lleva la contabilidad de sus ejecuciones, sin embargo, si durante la ejecución número 11 ocurre un error y la transacción hace un rollback, la variable regresará a su estado previo 10, lo mismo ocurrirá con cualquier cambio hecho en base de datos u otras variables.
Los EJB tienen los siguientes atributos de transacción que se pueden colocar a nivel de clase o para un método en particular, entendiendo que el atributo a un método tendrá prioridad sobre el declarado a nivel de clase, en cuyo caso, si un método no contiene ninguno, entonces se usa el de la clase:

Required
RequiredNew
Mandatory
Supports
NotSupported
Never

Tabla. 2.3 Atributos de Transacción Especificación EJB J2EE

No se explicará a detalle cada atributo ya que esta fuera del contexto de este trabajo, sin embargo, dentro del proyecto, generalmente todas las operaciones que implican una operación en base de datos se configura el atributo REQUIRED (atributo por Default en caso de que no se elija ninguno) que le indica al contenedor de EJBs que si se invoca un método y no hay una transacción activa entonces abra una, si ya existe alguna, entonces que se utilice la existente.
Para otros procesos de negocio menos complejos como obtener el resultado de consultas se utiliza el atributo SUPPORTS debido a que da un mejor performance, este tipo de atributo, indica que si existe una transacción abierta se utilice, sino existe no se utiliza ninguna y el contenedor continúa con la ejecución del método.

2.2.2.3 Capa de Persistencia.

La última capa del sistema, es la que tiene que ver con la interacción con la base de datos, donde las operaciones de persistencia de entidades de negocios y consulta de información ocurren, sea por ejemplo una entidad de negocio requerida para insertar las Ordenes que se generan diariamente dentro del sistema, dicha entidad puede ser persistida en base de datos mediante una tabla denominada Orders.

Para esta capa se utilizó el ORM (Object Relational Mapping) de Hibernate, el cual es un software de terceros (third party) de código abierto que sirve para realizar un mapeo entre tablas (y sus relaciones con otras tablas) de una base de datos contra objetos (clases) dentro de un sistema, este mapeo se puede entender mejor con el siguiente ejemplo:

Por ejemplo, si se tiene una tabla denominada “Orders” cuya estructura es:

```
CREATE TABLE ORDERS (ID bigint PRIMARY KEY CLUSTERED,
                     ORDER_NUMBER int,
                     QUANTITY bigint,
                     ACTIVE tinyint
                       .
                       .
                       .
);
```

Esta tabla se puede representar (mapeo) mediante una clase (Entity Hibernate Class) como:

```
public class Orders{
   private Long ID;
   private Long orderNumber;
   private BigDecimal quantity;
   private Boolean isActive;
   ...//getters/setters
}
```

El tipo de variables utilizada depende del motor de base de datos, por lo que se recomienda consultar la página del proveedor de base de datos para conocer que tipos son soportados y como se deben utilizar, en el ejemplo anterior, se puede ver que en la clase se ha declarado una propiedad “isActive” cuyo tipo es booleano:

```
      private Boolean isActive;
```

En la tabla de base de datos este tipo de datos se declara como:

```
      ACTIVE tinyint //Entero corto
```

Por lo que se entiende que al ajustar un valor de tipo true/false en el objeto, dentro de la tabla es guardado con un 1/0 respectivamente, debido a que el valor tinyint dentro de SQL Server representa un valor numérico entero de poca capacidad, sin embargo, en otros motores de base de datos, existe el valor Bolean dentro de sus tipos de datos, lo cual resultaría mas adecuado que un entero corto del ejemplo anterior.

De acuerdo al ejemplo anterior, la ventaja de realizar mapeos directos de tablas de base de datos contra clases de código, es precisamente no tener que lidiar directamente con la base de datos, cambiando las propiedades de los objetos que han sido mapeados junto con una sesión válida asociada, se pueden hacer todo tipo de operaciones como Consultas, Altas, Cambios, Borrado, teniendo que para cada tabla, existe un objeto asociado (esto de acuerdo a las necesidades de cada sistema, ya que hay tablas auxiliares por ejemplo auditoria (logs) que quizá no sea necesario su mapeo).

El uso de Hibernate como ORM junto con librerías auxiliares como XDOCLET y ANT también facilitan la creación del modelo entidad relación a partir de las clases mapeadas, ya que permiten generar en automático a partir de un objeto y sus propiedades, un script de base de datos para crear las tablas y sus relaciones dentro del motor de base de datos designado.
El siguiente ejemplo muestran dos objetos mapeados de la tabla de Orders y Account así como la relación entre ambas tablas, a saber, cada Orden creada debe contener una Cuenta asociada, por lo cual existe una relación (One to One) entre las tablas de Orders y Accounts:

Objeto ORDER Mapeado como:

```
public class Orders{
   private Long ID;
   pribate Long orderNumber;
   private BigDecimal quantity;
   private Boolean isActive;
   private Account account; //One to One
   public void setAccount(Account account){
          this.account = account;
   }
   public Account getAccount(){
       return account;
   }
   ...otros getters/setters
}
```

Como se observa, dentro de la Clase Orders se tiene una propiedad de tipo “Account” la cual pertenece a la clase Account (Mapeada a la tabla Accounts) y como se observa, mantienen una relación referencial, este tipo de relación también debe ser indicada dentro de la configuración de mapeos, el cual, para la versión Hibernate 2.0 se configura dentro de dos archivos principales, uno con extensión hbm.xml nombrado comúnmente con el nombre de la entidad que se desea mapear por ejemplo Orders.hbm.xml y Account.hbm.xml y un archivo de configuración general XML comúnmente nombrado hibernate.cfg.xml, aunque cabe mencionar que estos archivos en nuevas versiones de Hibernate han sido deprecados y en su lugar se utilizan anotaciones a nivel de clase para realizar la relaciones de entidades y en general el mapeo de las clases.

2.2.2.3.1 Delegación de Trabajo dentro de la Base de Datos.

Un aspecto importante dentro del sistema, fue el uso constante de objetos de base de datos como Funciones y Procedimientos Almacenados para realizar operaciones del sistema pero ejecutarlas en la base de datos. Lo anterior, resulta útil en el sentido de que se delega parte de funcionalidad a la base de datos, dando un mejor performance a la aplicación debido a que se reduce la cantidad de código y por ende ejecución del mismo dentro del servidor, a manera de ejemplo, el patrón de diseño FastLineReader (arriba mencionado) que se utiliza comúnmente para ejecutar y obtener los resultados de una consulta a la base de datos, utiliza un Procedimiento Almacenado asociado que contiene las instrucciones SQL necesarias, así como parámetros de entrada que permiten realizar filtros a la consulta, ordenación de los resultados y selección de los campos que se desean obtener, si se hubiese tenido que codificar dicha lógica dentro de la aplicación por cada consulta resultaría en una tarea laboriosa.

Pero no solo procedimientos almacenados son utilizados, también hay distintas “Funciones” de base datos para hacer ciertas operaciones aritméticas, útiles para algunas consultas de tablas que contienen información numérica, con esto, mucha lógica de programación quedo fuera del código del sistema y delegada a la base de datos.

Una de las principales ventajas de delegar trabajo a la base de datos, es el hecho de que en el caso de instalación de una nueva funcionalidad o modificación a una existente que se encuentra programada dentro de un objeto de base de datos, no es necesario detener la aplicación, re-compilar clases, iniciar la aplicación y revisar que el cambio hecho funciona correctamente, quedando claro que todo este proceso, puede demorar mucho tiempo y si además es ejecutado dentro de un ambiente de producción, entonces la situación se agrava debido a que genera la molestia de los usuarios y sobre todo, impacta directamente a la operación pudiendo causar pérdidas que pueden generar multas e incluso la suspensión de actividades. Los cambios hechos a algún objeto de base de datos son más sencillos de instalar, modificar o incluso remover, pero la principal ventaja, es que los cambios en ellos se ejecutan como comúnmente se conoce en caliente (hotfix), lo que quiere decir que no es necesario detener la aplicación para ver reflejado el cambio o incluso detener el servidor de base de datos, generalmente solo se envía el archivo con el “script” de instalación (CREATE), cambio (ALTER) o borrado (DROP) del objeto en cuestión e inmediatamente la aplicación ve reflejado dicho cambio.

2.2.3 Resumen de Módulos Desarrollados Durante el Proyecto.

- **Curso de Inducción al Sistema.**

Durante mi instancia en el proyecto, parte de mis labores fue la de entender toda la arquitectura en la que el sistema se construyó además de asegurar de que todos los equipos involucrados la respetaran, por lo tanto, cuando había nuevos ingresos en los equipos de desarrollo se impartía un curso explicando la arquitectura del sistema junto con ejemplos prácticos de como crear una interfaz de usuario (UI), realizar un servicio tipo CRUD, crear un EJB y asignar niveles de transaccionalidad, crear un procedimiento almacenado en la base de datos y mandarlo ejecutar desde el código fuente, etc. para que se familiarizaran y reducir así la curva de aprendizaje integrando a la gente al trabajo diario.

- **Nueva recepción de Ejecuciones desde la BMV.**

Uno de los mayores cambios que envió la BMV fue la conexión con las distintas casas de bolsa y su sistema de recepción de Órdenes. Previamente, se enviaba una cadena de caracteres basada en posiciones, en cada número de posición se encontraba un valor entendible por el sistema de la BMV, por ejemplo la cadena:

```
200      ¬B¬YHOO      ¬* ¬43001   ¬12.00    ¬CL
```

Indica el envío de una Orden con folio igual a 43001 de tipo al cierre (CL) y en donde se desean comprar (B) 200 títulos a un precio de 12.00MXN de la emisora/serie YAHOO *.

Esta forma de envío se re-diseño ahora utilizando la llamada a un Web Service que la BMV exponía para las casas de bolsa, por lo que se necesitó generar un cliente para la URL que nos fue entregada y para su integración e invocación dentro del sistema se utilizó la utilería de Java denominada "wsimport" con la cual se puede, a partir de una URL (Web Service corriendo bajo protocolo HTTP WSDL) generar en automático una clase cliente que se pueda utilizar dentro del sistema para hacer uso del mismo.
Por default esta utilería genera la clase incluyendo la URL del servicio web como un valor fijo, por lo que fue necesario realizar algunos cambios entre los cuales se dejaba la URL en cuestión dentro de un archivo de propiedades (.properties) esto con la finalidad de que si por alguna razón, la BMV cambiaba la dirección de su servicio web (la URL), no se tenga necesidad de volver a importar a una nueva clase y se pueda seguir utilizando la clase existente cambiando solamente la URL en el archivo de propiedades.

- **Nuevas Interfaces de Usuario para mostrar cambios de precios en el mercado utilizando el framework de AJAX.**

Siendo el framework de FLASH un poco tedioso tanto en su implementación como codificación, como se explicó anteriormente en el apartado de capa de presentación, se tuvo la necesidad de realizar algunos monitores que pudieran mostrar información en tiempo real pero con la aprobación del usuario de evitar el uso de la tecnología FLASH para dicha tarea. La propuesta, fue utilizar un framework de AJAX denominado Jquery, el cual incorpora algunos controles (JqGrid) que sirve para mostrar información dentro de una grilla en tiempo real, permitiendo realizar operaciones asíncronas de manera fácil de codificar. Se realizaron los monitores integrando Jquery/JQGrid conectándolos mediante un envío HTTP POST cada 5 segundos (pudiéndose parametrizar) para ejecutar una consulta de información que al tener cambios en base de datos, se veía reflejada inmediatamente dentro de la interfaz de usuario, similar al monitor de ordenes construido en FLASH.

- **Mejoramiento al Algoritmo de Asignación**

Una importante mejora fue realizada al algoritmo de Asignaciones, el cual es el proceso más complejo e importante del sistema, debido a que en gran medida contribuye a realizar de manera exitosa las operaciones que generan la ganancia dentro de la compañía. Por ejemplo, si un cliente desea comprar 1000 títulos de una emisora a cierto precio, el algoritmo de asignación es parte fundamental para que esto suceda ya que registra las asignaciones hechas a una Orden a través de una ejecución recibida, por cada orden completada correctamente, entonces, la casa de bolsa recibe una comisión de acuerdo al cliente (Cuenta) al que se representa.
Sin embargo, este proceso tenía un problema, en muchas ocasiones, la BMV tiene equivocaciones y puede enviar una Ejecución que no era destinada a cierta casa de bolsa, en cuyo caso, se envía una nueva Ejecución, pero de cancelación, por lo que todas las asignaciones ocurridas se deben de borrar (borrado físico) en ese instante y volver a generar sin utilizar obviamente la Ejecución cancelada generando 2 eventos secuenciales con el siguiente orden:

a) Se produce una des-asignación de todas las ordenes cuya emisora y sentido hagan match con la ejecución cancelada, en ese momento, todas las ordenes vuelven a su estado inicial, donde la cantidad asignada es igual a 0 y la cantidad por asignar es igual a la cantidad solicitada.

b) Una vez que las ordenes vuelven al estado inicial, comienza a correr el proceso de asignación tomando todas las Ejecuciones recibidas durante el día, descartando únicamente la Ejecución cancelada.

Descripción del Problema.- Con el comportamiento arriba descrito, se tiene una problema en el performance debido a que se vuelve a correr todo el algoritmo de asignación para todas las ordenes y ejecuciones desde cero, es decir, tomando todas las Ordenes y Ejecuciones desde el inicio del día dando como resultado un impacto negativo al sistema que incluso en días con alto volumen de operaciones, se llegó a tener que reiniciar el servidor de aplicaciones causando perdida de información y discrepancias en la integridad de los datos, por lo que incluso se tuvo que realizar modificaciones directas a la base de datos para remediar el problema.

Solución.- Analizando el algoritmo de asignaciones junto con sus reglas, no era necesario hacer una des-asignación total de todas las ejecuciones recibidas, debido a que si una ejecución se recibe a cierta hora del día entonces, existe una regla de asignación que dice que solo se asignará en Ordenes posteriores a la recepción de dicha ejecución, sin tomar en cuenta las Ordenes cuya hora de creación sean menores, por ejemplo, sean las siguientes Ordenes:

Orden1
```
Folio 1 Security -YAHOO-* Cantidad 500 Hora Envío a la BMV 11:29 AM
```
Orden2
```
Folio 2 Security -YAHOO-* Cantidad 500 Hora Envío a la BMV 11:31 AM
```

Se recibe la siguiente EJECUCION desde la BMV.-

Ejecución
```
Folio 1  Security YHOO-* Cantidad 500 Hora 11:30 AM
```

De acuerdo al escenario anterior, la ejecución recibida con hora de 11:30, solo afecta a la orden cuyo folio es 2 debido a que tiene una hora de envío a las 11:31 el cual es superior a la hora asignada en la ejecución, la orden con folio 1 no se verá afectada y ni siquiera será tomada en cuenta por el algoritmo de asignación.
Si por ejemplo se recibe la cancelación de la ejecución, en un inicio, el sistema cancelaba y des-asignaba todas las ordenes y generaba nuevamente el proceso de asignación, incluyendo todas las Ordenes con hora anterior a la hora que tiene la ejecución cancelada, sin embargo, dichas Ordenes permanecerán igual por lo que no es necesario des-asignarlas y volverlas a asignar, ya que al final quedaran exactamente igual, las únicas ordenes que se verán afectadas serán aquellas que tengan hora posterior a la de la ejecución en cuestión, por lo tanto, se realizaron los ajustes pertinentes al algoritmo para que únicamente tomara en cuenta las ordenes que se verían afectadas por la cancelación de la ejecución y eliminar la carga de trabajo innecesaria para las que no son afectadas y con esto, mejorar

significativamente el performance de la aplicación disminuyendo la tardanza del proceso que se tenía anteriormente, sobre todo en días en donde la operación rebasaba los cientos de Ordenes.

- **Módulo de RINO (Revisión Integral de la Norma Operativa)**

RINO, fue una serie de cambios para todas las casas de bolsa que operaban bajo la regulación de la BMV, dichos cambios son mejoras a los protocolos de envío y recepción de ordenes mediante la implementación del protocolo FIX*, cambio a algunas reglas a las Ordenes y el sistema de precios entre otras normas.

Este proyecto estuvo copado a una fecha fija, no modificable, es decir, no se podía pedir una ampliación a la fecha de entrega debido a que la BMV envío un documento con todas las nuevas regulaciones, la cuales se tenían que cubrir de manera obligatoria sin omitir ninguna de ellas para la fecha marcada como día de entrega (due date), en caso de no cubrirse con el 100% de los cambios en dicha fecha, la casa de bolsa que no cumpliera se haría acreedora de una fuerte multa y se cancelaría su operación no permitiendo tener ningún tipo de actividad hasta que el comité de la BMV diera su visto bueno.

Debido a lo anterior, hubo una fuerte presión a los equipos de desarrollo, por lo que fue necesario trabajar a marchas forzadas desde el primer día antes de que la fecha comprometida fuera alcanzada. Afortunadamente se logró terminar con todos los cambios establecidos con 2 semanas de anticipación. Parte de las actividades más importantes dentro de la normatividad, fue el hecho de integrar el protocolo FIX*, el cual es, un protocolo de comunicación que sirve para poder generar operaciones de compra y venta de capitales mediante el envío de información desde un servidor automático (robot), el cual, generalmente se implementa mediante un servidor de cola de mensajes, el cual envía cadenas de texto (construidas de acuerdo al protocolo) para que una vez recibidas por cualquier casa de bolsa, se conviertan en Ordenes que se enviaran a la BMV, con esto, desde cualquier parte del mundo mediante alguna casa de bolsa, se puede participar en las operaciones de mercado mediante el envío de órdenes, las cuales, una vez generadas, son tratadas como cualquier otra que se haya generado dentro de cualquier sistema, con la diferencia que fueron recibidas desde otro lugar, para el caso de Merrill Lynch, los mensajes provenían desde un servidor de mensajes establecido en New York, desde donde un grupo de Traders enviaban Ordenes para comprar o vender acciones en la BMV.

***FIX** Financial Information eXchange es un protocolo para comunicación electrónica dedicado para transacciones de compra y venta de capitales (Securities) dentro del mercado accionario.

Capítulo 3
Compañía JP Morgan Chase Vastera

Dirección.- Av. Prolongación Paseo de la Reforma 1236 Piso 8 México, (Santa Fe)
Giro**.** Casa de Bolsa/Logística
Posición. Applications Development Sr.
Project Manager. Robert Fenster
Organigrama del Proyecto:

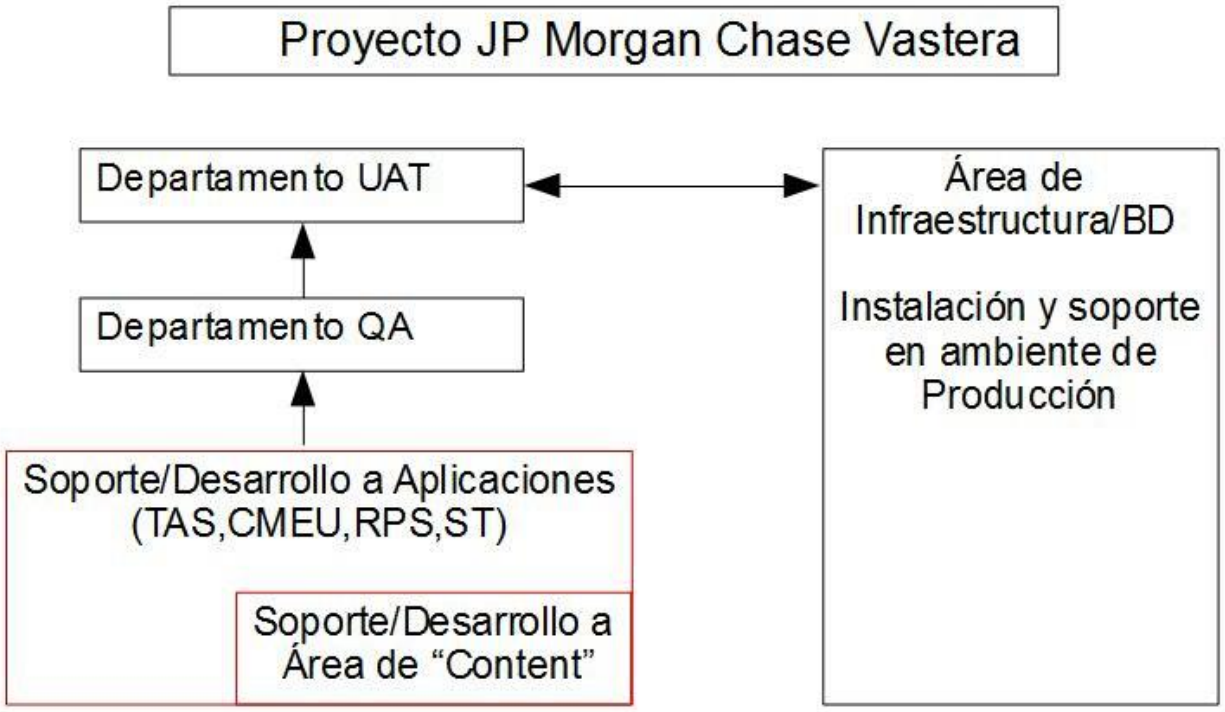


Fig. 3.1 Organigrama Proyecto JP Morgan

3.1 Introducción General a las Aplicaciones de la Compañía.

La empresa JP Morgan tiene varios negocios, entre los que destacan servicios Bancarios para empresas, casa de bolsa con presencia internacional incluyendo México, servicios de logística con soluciones de sistemas y soporte, etc. Dentro de esta empresa tuve la posición de Applications Development para la subdivisión de Logistics la cual, es una división encargada para el proceso de clasificación de productos a nivel global, así como el servicio de logística en cuanto a importación/exportación dentro de la comunidad europea dando soporte, desarrollo y mantenimiento a distintas aplicaciones ofrecidas como productos dentro de la compañía.

3.1.1 Arquitectura Genérica de las Aplicaciones.

Antes de revisar las distintas aplicaciones, se da una breve introducción a la arquitectura de las mismas, la cual está basada en 2 capas, un front-end para la interface de usuario y el back-end donde ocurren todas las llamadas a procesos de negocio, esta aplicaciones son de tipo Web por lo que se instalan en un servidor de aplicaciones y los usuarios las ejecutan desde un navegador.
La parte del Front-End se desarrolló en un entorno basado en AJAX, mediante una Aplicación Web la cual lleva por nombre “Tradesphere”, en la cual se integran decenas de módulos JavaScript, que una vez instalada dentro de un servidor web (Tomcat, IBM Websphere, WebLogic) sirve como base para montar otras aplicaciones, para evitar confusión, esta aplicación web, se entiende que es una aplicación para soportar otras aplicaciones, abstrayendo ciertas funciones comunes que cualquier aplicación web contiene, por ejemplo un módulo de Login para validar el inicio de sesión de un usuario, manejo de sesiones, controles de usuario (tablas de datos, calendarios, etc.), hojas de estilos, paginas HTML genéricas (Error, ReDirect, etc.) funciones para validaciones de campos, manejo de locales e idiomas en etiquetas, funciones para construcción de interfaces de usuario.

El back-end se compone de clases construidas con el lenguaje de programación Java dentro de un archivo JAR (cp.jar) el cual se conoce como CP (Collaborative Platform) y contiene los módulos necesarios para desarrollar la capa de negocios de una manera sencilla, por ejemplo, para la codificación de algún nuevo servicio de negocio, se importa la librería de la plataforma colaborativa y se pueden utilizar los diferentes servicios que expone entre los cuales se destacan: el obtener una conexión con la base de datos, limpiar la conexión cuando el servicio termina, manejar niveles de transaccionalidad, manejar información de sesiones de usuario, funciones genéricas para fechas y horas, manejo de cadenas, funciones de redondeos y otros cálculos, manejo de errores etc.,
Por lo tanto, con solo unas cuantas líneas de código, utilizando la Plataforma Colaborativa de ahora en adelante abreviada como CP, cuando se desarrolla algún nuevo módulo, el desarrollador, no tiene que preocuparse, por ejemplo, del manejo de la sesión con la base de datos, el nivel de transaccionalidad o el manejo de errores, ya que todo queda delegado a este framework utilizando únicamente los servicios que expone.

Otro punto importante de la arquitectura, es la manera en la cual está construida, dado que tanto el modulo web denominado “Tradesphere.war” junto con las clases de la CP (cp.jar) no son una instalación típica web java, en donde comúnmente el modulo web se empaqueta junto con todos los recursos dentro de un archivo con extensión .war y se instala dentro de un servidor de aplicaciones. En este caso, la forma de instalar las aplicaciones es totalmente distinta, utilizando 5 java virtual machines (JVMs) corriendo dentro del servidor de aplicaciones, que en su conjunto constituyen todo el marco de trabajo genérico:

- **Tradsphere Server**.- Contiene todo el framework web, como ya se explicó, aquí se montan todas las paginas HTML, archivos JavaScript, hojas de estilo, imágenes, clases, etc. Por lo tanto, aquí reside todo el front-end de la aplicación, utilizando folders bajo la carpeta Web-INF, se instalan todas las pagina pertenecientes a la aplicación, dichas paginas pueden ser de cualquier tecnología desde simples páginas estáticas con extensión .html hasta otras más dinámicas con extensión .jsp, también se incluyen archivos javascript (.js), archivos de imágenes, hojas de estilo.

- **Report Server**.- Esta JVM es parte de Tradesphere pero se extrajo del framework a un módulo independiente únicamente por motivos de performance, sirve para generar y despachar todos los reportes, los cuales básicamente se generan en hojas de cálculo (*.XLS) y tipo PDF, aunque también se incluye un módulo para conexión y envío de FAX.

- **Engine Singleton**.- JVM encargada de correr los procesos de negocios, es decir, todas las clases de la aplicación Tradesphere, así como la de la aplicación que se integre en ella, ejecuta las llamadas al back-end, por ejemplo, en caso de tener un error dentro de una clase de negocio, se debe revisar los logs de esta JVM para ver la excepción arrojada.

- **Extension Engine Singleton**.- JVM dedicada para todas las conexiones con sistemas externos, por ejemplo, recepción y envío de correos, ejecución de Webservices, conexiones a colas de mensajes, conexión a otros sistemas dentro del mismo servidor u otros externos a las aplicaciones mediante los protocolos HTTP/HTTPS/FTP/SFTP

- **Admin Server**.- Coordina todas las JVMs arriba mencionadas mediante un puerto privado, por ejemplo, si un usuario envía información desde una UI (ubicada en el Tradesphere Server), esta JVM recibe la petición y la envía al Engine Singleton Server para procesar la llamada al Back-End y entonces obtener una respuesta, dicha respuesta vuelve al Admin Server y se envía finalmente al Tradesphere Server que utilizará respuesta ya sea para mostrar éxito o fallo en la llamada del servicio. Esta es la JVM mas importante, si no se inicializa no se puede ejecutar ninguna aplicación

Finalmente otra parte de la arquitectura, es que el contenido estático de las aplicaciones, se manejan dentro de un servidor web http, el cual, sirve para despachar paginas HTML, imágenes y archivos JavaScript, etc., ayudando a mantener un mejor performance, debido a que se le delega este trabajo, creando un balance de cargas reflejado en una mayor rapidez para las solicitudes de páginas que realizan los usuarios.

De aquí en adelante, todas las aplicaciones que se mencionan están construidas bajo la misma plataforma colaborativa arriba mencionada (CP) y por ende, comparten la misma arquitectura teniendo como motor de base de datos común la base de datos Oracle.

3.1.2 Resumen de las Actividades Realizadas por Aplicación.

- **CMEU (Custom Management For Europe Union).**

Descripción.- Esta aplicación es la más robusta y compleja de la compañía, ya que se encarga de llevar a cabo el registro de cualquier producto que sale desde alguna planta con destino a algún país, guarda el folio único, la descripción de los productos, el estatus de la orden generada y todos los datos de la misma como el país destino, el método de envío (tierra, mar o aire), cálculos de impuestos, etc. Este sistema recibe notificaciones del sistema TAS mas adelante descrito y de acuerdo a dichas notificaciones, los usuarios deben realizar distintas acciones según el tipo de notificación.

Actividades.- Dentro de esta aplicación, las principales actividades fueron el soporte a los usuarios de la misma, mantenimiento de módulos existentes y en pocas ocasiones, desarrollo de módulos nuevos.

- **TAS (Transit and Statistics)**

Descripción.- Aplicación instalada en las aduanas de los países integrantes de la comunidad europea, la cual sirve como apoyo a la aplicación CMEU para obtener la descripción de la carga (ítems) que están circulando verificando que este completa, si es el caso, entonces se hace el cálculo de impuestos y actualizan campos dependiendo las regulaciones aduanales del país en donde se encuentre, esta información se envía de regreso al sistema CMEU mediante colas de mensajes para guardar un registro del estado de la carga así como del lugar en donde se encuentra. Si en alguna aduana se detecta que faltan piezas u ocurrió un percance, esta información también se registra y se envía de regreso para que los usuarios tomen las decisiones pertinentes.

Actividades.- Dentro de esta aplicación, las principales actividades fueron el soporte a los usuarios de la misma y el mantenimiento de módulos existentes debido a que constantemente, las aduanas de los países europeos cambian sus reglas (cada año) exigiendo nuevos datos para las cargas que allí circulan, además de constantes cambios en la forma de calcular el impuesto que se cobra.

- **RPS (Restricted Party Screen)**

Descripción.- Esta aplicación es un sistema de seguridad, principalmente utilizada en el área bancaria de la compañía y sirve para cotejar las listas negras tanto de personas como direcciones con las cuales el gobierno de EUA no puede realizar ningún tipo de actividad de tipo bancaria o comercial. La aplicación consta de varios módulos, sin embargo, el más utilizado es aquel en donde se los usuarios colocan un nombre o dirección a manera de texto abierto, por ejemplo, si se escribe el nombre: “Joaquín Guzmán Loera”, la aplicación lo escanea y busca en las listas negras ese nombre pudiendo arrojar:
"El Chapo Guzmán", "Shorty Guzmán", “El chapo”, etc.
Pero no solo se escanean nombres, también se escanean direcciones, mismas que se cotejan con varias bases de datos para encontrar patrones de coincidencia y arrojarlos a la como direcciones ilegales.

Actividades.- A esta aplicación se le daba soporte de infraestructura, es decir, se depuraban archivos tipo log, se depuraba el cache de las paginas, se reiniciaba el servidor de aplicaciones cuando era necesario o cuando había bajo performance, se mantenía el código fuente y se apoyaba en la participación en cambios al mismo.

- **ST (Solicitation Tracker)**

Descripción.- Esta aplicación, está enfocada en mantener las solicitudes de envío de productos de ciertas compañías en una base de datos confiable durante su almacenaje, básicamente, cuando un cliente envía una solicitud denominada “REQUEST” en la cual coloca distintos productos identificados mediante un ID único, además de descripciones, cantidades, etc., el sistema envía éste REQUEST para que entonces, cuando los productos se reciben para su almacenaje de acuerdo con la solicitud enviada, se genere un registro de respuesta denominado “RESPONSE”, con los productos recibidos, dicha respuesta es procesada por el sistema generando entonces la documentación correspondiente para enviar un reporte por correo o FAX al cliente, para enterarlo de que sus productos fueron recibidos correctamente y quedaran en resguardo por los días que fue solicitado su almacenaje.

Actividades.- Este sistema fue mejorado construyendo nuevas mejoras al usuario, una de ellas fue dentro de los productos a dar de alta, ya que se agregó la opción de poder agregar una imagen del estado de los mismos, ya que en algunas ocasiones, los clientes manifestaban que los productos estaban en buen estado pero que en el almacenaje se dañaban, por lo que la imagen sirve como evidencia del estado en que se recibió la mercancía antes de su almacenaje.
Otro cambio importante, fue en la forma de operar el sistema, ya que en algunas ocasiones, se cargaban grandes volúmenes de productos, arriba de 2000, por lo que fue necesario una forma de introducirlos utilizando eventos de teclado eliminando toda necesidad del “mouse”, con esto, los capturistas cargan más rápidamente altos volúmenes de datos, otro problema fue que comúnmente, durante estas cargas, los usuarios salían a comer después de haber realizado la carga de cierta cantidad de productos, pero olvidaban guardar la información, cuando regresaban su sesión con el servidor había caducado y por ende todo su trabajo estaba perdido, teniendo que volver a cargar toda la información, por lo tanto, se agregó una funcionalidad de auto refrescado de sesión mientras el usuario trabajaba, así como una funcionalidad de auto-guardado en el caso de que la sesión expire y se detecte un trabajo sin salvar.

3.1.2.1 Actividades para el Departamento de “Content”.

El departamento de Content pertenece a una subdivisión denominada “Classiffier” quienes se enfocan a la clasificación de productos dentro de la comunidad europea, se construyeron las siguientes aplicaciones a solicitud de éste departamento: “HSLoader” y “FileSet Receiver”, ambas construidas en JAVA utilizando el marco de trabajo “Spring” para la capa de servicios back-end, Jface para la capa de presentación y Oracle como motor de base de datos, estas aplicaciones ayudan a realizar actividades al departamento que tardaban horas cuando se hacían manualmente, a continuación un breve resumen de cada una de ellas.

HSLoader. Esta aplicación se conecta a páginas de Internet para obtener números denominados HS (Harmonized Commodity Description and Coding System) los cuales son números de clasificación internacional que sirven para estandarizar todo aquello que se puede exportar, por ejemplo, 1000023 puede representar un Caballo de Carreras, pero 1000024 representa un caballo muerto para consumo humano, etc.
Cada cuarto de año se debe revisar la clasificación de productos para mantenerlos actualizados y vigentes, siendo que a algunos de ellos solo les cambian la descripción, otros son eliminados y otros tantos son agregados a la clasificación. Previamente se construyó un software capaz de realizar las actualizaciones de los HS numbers, pero resulto bastante lento, tardaba más de 5 horas para ejecutar una actualización y se pudo constatar que tenía errores debido a que por alguna causa no insertaba muchos números de clasificación. La versión 2 se construyó completamente de 0 a petición del cliente, es decir no se utilizó la aplicación previa e integre una interfaz amigable de tipo standalone utilizando Jface que es un framework de desarrollo basado en SWT (utilizando la librería de código abierto swt.jar) el cual utiliza la interfaz nativa del sistema operativo pero sin depender del mismo, con esto, cambiando una librería, la aplicación se puede exportar a cualquier otro sistema operativo como Linux (gwt.jar).
La nueva aplicación demoraba 5 minutos para cargar más de 100 mil números HS y se evitó la dependencia de un equipo de capturistas, para los procesos de actualización debido a que la aplicación se programaba periódicamente para revisar en las páginas oficiales por nuevas actualizaciones, si las encontraba, entonces procedía de manera automática con dicha actualización y enviaba un correo informando de los registros afectados.

FileSet Receiver. Esta aplicación surgió de la necesidad para cargar números HS de emergencia, es decir, a veces dentro de la comunidad europea, se requería actualizar ciertos números HS, sin embargo, al no ser estos cambios tan grandes, se enviaban a través de un archivo en Excel, los usuarios debían cotejar dicho archivo y agregar/modificar los nuevos HS numbers manualmente, por lo tanto, desarrolle una utilería con java/spring y jface similar al HS Loader, la cual cargaba el archivo de Excel y mostraba a los usuarios dentro de una tabla, los números HS nuevos, los existentes que tendrían modificación y los que se eliminarían, por lo tanto se pueden elegir los números que se desean afectar (o todos). Una vez elegidos, el programa hace todas las operaciones necesarias manteniendo la vigencia de los números dentro de la base de datos, ahorrando bastante tiempo en cotejar el archivo recibido y hacer la carga manual como se realizaba anteriormente.

3.1.2.2 Migración de Aplicaciones a una Infraestructura de Código Abierto.

Cuando ingrese a la compañía, las aplicaciones CMEU, RPS, TAS, ST estaban instaladas y repartidas en los siguientes servidores de aplicaciones: WebLogic, IBM Websphere y en los siguientes servidores http web: IBM HTTP Server, iPlanet Web Server corriendo bajo un sistema operativo Windows Server. Lo anterior, sobre todo en la parte de los servidores de aplicaciones comenzó a generar un fuerte costo a la empresa, en primer lugar, porque el tema de renovación de licencias se volvió alto y difícil de sostener, siempre ligado a que en cierto momento el soporte técnico a los servidores se cancelaba si no se renovaba la licencia, por lo que entonces se tomó la decisión de que todas las aplicaciones se migrarían totalmente a una nueva infraestructura de código abierto para reducir los altos costos. Parte de mi labor fue proponer ante un comité la nueva infraestructura que mitigaría costos y una vez tomada una decisión, mi labor se enfocaría en comenzar a migrar las aplicaciones.

La nueva plataforma para las aplicaciones quedo de la siguiente manera:

Sistema operativo:	Red Hat Linux en un servidor Virtual
Servidor de aplicaciones:	Apache Tomcat
Servidor Http Web:	Http Apache Server

Tabla. 3.1 Migración de Aplicaciones a Open Source.

Se comenzó con la migración de la aplicación Solicitation Tracker (ST) para después continuar con las demás, esto como estrategia para conocer qué tipo de errores se presentarían debido al cambio de Sistema Operativo, Servidor de Aplicaciones y servidor http Web, afortunadamente a pesar de que ocurrieron ciertos problemas, sobre todo en las clases de la plataforma colaborativa (cp.jar) debido a que ciertas clases tenían dependencias específicas para funcionar sobre los servidores WebLogic y Websphere, por lo que se tuvo que agregar clases específicas para el servidor de aplicaciones Apache Tomcat.

Se encontró además, otro tipo de problemas relacionados con las páginas web construidas con la tecnología JSP que no soportaban cierto tipo de operaciones, como el manejo de la sesión en la especificación del nuevo servidor de aplicaciones, sin embargo, se pudo llevar a buen término el proceso de migración, corrigiéndose todos los problemas además de no tener afectaciones para el caso que se quisiera regresar nuevamente a los servidores anteriores. Con el tiempo, todas las aplicaciones fueron migradas hasta tenerlas en una infraestructura común, lo que permitió tener un mejor control de las mismas, así como en conseguir un ahorro económico para la compañía bastante significativo.

3.1.2.3 Proyecto IBM Datastage.

Parte de las soluciones compradas dentro de la compañía dentro del área bancaria, fue el software IBM Datastage. Me fue asignado un proyecto de re-diseño a un software existente construido tiempo atrás que recibía archivos de texto plano colocados en una carpeta de entrada desde un servidor FTP, para procesarlo hacia una base de datos u otros archivos de salida.

Problemática. El software anterior, comenzó a ser difícil de mantener ante cualquier cambio, debido a que cada vez se requerían mas archivos de recepción y los ya existentes, cambiaban con mucha frecuencia, es decir, si un nuevo campo era agregado a los archivos de recepción, se tenía que hacer una fuerte modificación en el código, re-compilación, empaquetado, pruebas e instalación, con lo cual, muchas veces se afectaban otro tipo de archivos recibidos generando errores y retrasos en los procesos, por lo que se tenía que volver a revisar el código para ver lo que fue afectado y corregirlo a la brevedad.

Descripción de la Actividad. Para solucionar esta problemática, se implementó el uso de la herramienta IBM Datastage, la cual, entre otras funciones, sirve para procesar información desde distintas fuentes de entrada como pueden ser un archivo de texto plano o separado por comas, el resultado de una consulta en base de datos, un mensaje recibido desde una cola de mensajería, un petición http, etc., y al recibirse la información, se puede realizar distintas operaciones de transformación o simplemente recibirla sin hacer ningún tipo de tratamiento para enviarla a alguna fuente de salida, que puede ser, de igual manera un

archivo que se guarda físicamente en disco, a una base de datos, a un servidor FTP, a un servicio web, a colas de mensajería etc.

Teniendo que los archivos recibidos tiene distintos formatos y propósitos, el proyecto se dividió en varias clasificaciones, generando la transformación de los datos de los archivos recibidos para enviarlos a la base de datos como lo hacía anteriormente el programa que se utilizaba, pero ahora con la diferencia que se crearon distintos proyectos dentro de la herramienta Datastage, por lo que al dividir en proyectos, se obtuvo un mejor control sobre la recepción de archivos permitiendo que si un tipo de archivo tuviese un cambio o modificación dentro de su estructura, simplemente se abría el proyecto para ese tipo de archivo, se hacen los cambios solicitados dentro de su espacio de trabajo dentro del servidor y no se afecta la recepción de otros proyectos, manteniéndolos intactos permitiendo su funcionamiento de manera paralela.

3.1.3 Proceso de Promoción de Código.

Para finalizar este capítulo, menciono la forma en la cual se entrega cualquier desarrollo realizado dentro de la empresa, ya que es un procedimiento algo laborioso pero que sin duda, mantiene un buen control a todos los cambios que ocurren en los sistemas.
Todo empieza con la generación de un ticket dentro de una aplicación web propietaria (Serena Software), la cual genera un ID único y en la cual se debe hacer una descripción de los cambios a realizar, después se deben agregar las rutas de los archivos afectados junto con su versión antes y después de los cambios, dicha versión se obtiene en automático desde el software para control de versiones (PCVS es el software utilizado dentro de la compañía), después se genera la documentación, la cual, es básicamente, un archivo de texto con datos de los nombres de archivos, descripción detallada de los cambios, numero de ticket asignado, caso de uso para probar los cambios (si aplica), pasos para la instalación de los cambios, scripts de base de datos (si aplica).

Todo lo anterior se empaqueta dentro de un archivo tgz/zip según el sistema operativo donde se hagan las pruebas y se coloca en un servidor con un directorio compartido con el departamento de QA, a quienes mediante un correo electrónico se les da aviso de que un nuevo cambio ha sido promovido, por lo que ellos revisan la documentación siguiendo los pasos allí descritos tanto para la instalación como para las pruebas, si todo está bien, este departamento envía otro correo al departamento de UAT quienes realizan los mismos pasos, pero probando directamente con los usuarios, si ellos determinan correctos los cambios, finalmente el departamento de UAT envía un correo al equipo de producción con las mismas instrucciones para realizar la instalación final de los cambios realizados, una vez el cambio está en el ambiente de producción, entonces el ticket generado debe cerrarse, cambiando su estatus de Open a Closed y con esto termina el ciclo, en caso de que se detecte algún problema dentro del ciclo por ejemplo en el departamento de QA o UAT entonces, el ticket se pasa a estatus rechazado (Rejected) y se deben hacer las correcciones pertinentes para volver a iniciar el ciclo una vez más.

Capítulo 4
IBM Servicios Profesionales

Dirección. Calzada Legaría No. 853 Irrigación, Miguel Hidalgo, CIUDAD DE MEXICO, Distrito Federal, 11500
Posición. IT Specialist.
Giro**.** Consultoría en Sistemas
Project Manager: Sai Peruru Kumar
Personas a mi cargo: Ninguna
Organigrama del Proyecto.-

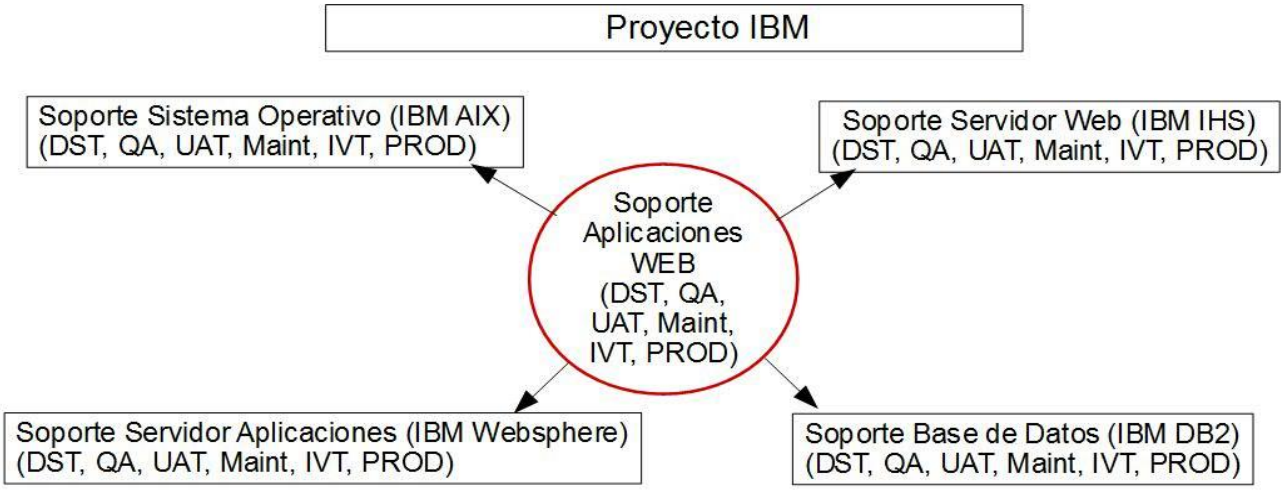


Fig. 4.1 Organigrama Proyecto IBM.

4.1 Introducción General del Proyecto.

Las actividades realizadas dentro de esta empresa, son brindar soporte a distintas aplicaciones web internas propiedad de la compañía, entre las que destacan ICFS Web, GCS, GCMS, las cuales se instalan en diferentes ambientes y tienen como principal utilidad la venta o arrendamiento de equipos de computo e impresoras en distintos países ya sea para dependencias gubernamentales o iniciativa privada.
El soporte a las aplicaciones se puede resumir a: Servidores de Aplicaciones, Servidores de Bases de Datos, Servidores HTTP Web y Sistema Operativo.
Siendo IBM una empresa que tiene varios productos de software en el mercado, es obvio que dichos productos son los mismos que se utilizan para sus aplicaciones, por lo que a continuación se mencionan:

Servidor de Aplicaciones	IBM Websphere Server
Servidor de Base de Datos	IBM DB2
Servidor http	IBM HTTP Server (AHE Server)
Sistema Operativo	IBM AIX OS

Tabla. 4.1 Arquitectura de las aplicaciones.

4.1.1 Descripción de Ambientes.

Antes de proceder con el resumen de actividades a continuación se hace un resumen de los ambientes existentes así como su utilidad.

Ambientes de Pruebas, denominados "DST Environment"
Este tipo de ambiente, es utilizado por los desarrolladores de las aplicaciones para hacer pruebas tanto de nuevos módulos como modificaciones a existentes, así como cuando se construyen nuevas aplicaciones. Toda la configuración que pueda existir la realiza el equipo de desarrolladores, por lo que son muy pocas las ocasiones donde se pide soporte a este tipo de ambientes a excepción de cuando se requiere ayuda para actualizaciones, problemas de performance o ejecutar/detener aplicaciones mediante línea de comandos.

Ambientes de QA/UAT/Maint, denominados "CDT Environments".
Dentro de los ambientes para QA/UAT también denominados ambientes CDT, se instalan las aplicaciones a un grupo de servidores identificándolos según el ambiente:

- **Servidores ALPHA (QA)**

Los servidores denominados ALPHA, representan el ambiente para pruebas de QA (Quality & Assurance) en donde se prueban los cambios a las aplicaciones una vez liberados por el equipo de desarrollo, son por primera vez instalados antes de promoverse a un ambiente de producción. Existen un grupo de usuarios que realiza las pruebas necesarias dando su visto bueno o rechazo dependiendo los resultados de las matrices o casos de prueba a evaluar, durante estas pruebas se debe tener la aprobación de todas la regiones geográficas en donde la aplicación está instalada (si aplica), dichas regiones actualmente son: América, Europa y Asia, por lo tanto, los horarios de las pruebas se pueden extender durante toda una noche comenzando en América y terminando en la mañana de ASIA, todos los equipos de soporte y desarrollo deben montar guardia y tener disponibilidad durante las pruebas para la resolución de cualquier eventualidad.

- **Servidores BETA (UAT)**

Este tipo de servidores se utilizan para instalar ambientes de UAT (User Acceptance Testing), por lo tanto aquí se instalan las aplicaciones para que los usuarios técnicos den su visto bueno y aprobación, como se mencionó anteriormente, la aprobación se debe dar en todas las regiones cuando aplique y se debe mantener guardia para cualquier contingencia durante las pruebas.

- **Servidores Maint**

Este tipo de servidores funcionan como contingencia para los servidores de BETA (UAT), debido a que en algunas ocasiones, cuando alguna nueva liberación es enviada y es detectado un error durante la revisión de los usuarios, entonces, se espera que se corrija a la brevedad por parte del equipo de desarrollo, sin embargo, dicha corrección no se instala directamente en el servidor de UAT, sino que se instala en el servidor de mantenimiento (Maint), con lo cual, los usuarios pueden verificar que efectivamente el error ha quedado resuelto y al hacer la misma prueba que lo arrojo, efectivamente no vuelve a presentarse.

Ambientes Productivos “PRODUCTION Environments”.

- **Servidores IVT**

También conocidos como Pre-Producción, es un ambiente donde se instalan las aplicaciones antes de instalarse en un ambiente de producción, por lo tanto, este ambiente es una copia idéntica a producción, se tienen las mismas versiones de sistema operativo, servidor de aplicaciones, servidor http web, base de datos y por tanto las mismas configuraciones, esto con la finalidad de hacer una prueba de instalación, para tener la certeza que al promover en producción no se tendrá ningún tipo de problema, si la instalación en este ambiente falla, se tienen altas probabilidades de que falle también en producción, por lo que se deben hacer las correcciones pertinentes para corregir cualquier anomalía, si se logra una instalación exitosa entonces se autoriza la promoción al servidor de producción.

- **Servidores de Producción**

Este es el ambiente donde se instalan finalmente las aplicaciones y como su nombre lo indica, es el ambiente productivo en donde los usuarios finales utilizan las aplicaciones para desarrollar su trabajo diario. Generalmente, este es el ambiente con más restricciones y seguridad, teniendo que solo un cierto equipo de personas pueden modificar alguna configuración dentro de él o realizar otras actividades como el reinicio de aplicaciones, cambios de memoria, cambios a la base de datos etc.

4.2. Resumen de las Actividades Realizadas.

A continuación se resumen las actividades realizadas en los ambientes arriba descritos

- **Soporte primario a usuarios.**

Este tipo de actividad, se realiza cuando la aplicación presenta alguna falla en cualquiera de los ambientes, entendiendo por falla las siguientes causas: los usuarios reportan bajo performance, un proceso de negocio no genera el resultado esperado, se presentan excepciones dentro de la infraestructura como falta de memoria, un bloqueo en la base de datos, etc.,

Dependiendo el ambiente, se deben de tomar diferentes acciones, para el caso de los ambientes DST/CDT, se tiene control total de los mismos, pudiendo reiniciar los ambientes o realizar cualquier configuración que permita resolver el problema, todo debe quedar registrado en un ticket para tener una evidencia de lo que sucedió y como fue resuelto junto con el equipo que lo atendió, dado que existen varios equipos de acuerdo a la infraestructura, por ejemplo, si se detecta un problema de memoria dentro de la base de datos, debido a que un tablespace alcanzo su límite, se debe coordinar y trabajar entonces con el equipo de base de datos (DBA's), pero si el problema radica en un permiso sobre un directorio dentro del sistema operativo, entonces el ticket se dirige al equipo de soporte para los sistemas operativos (AIX Team).

Para el caso de errores en ambientes de Producción, entonces, se debe levantar un ticket con la máxima severidad (High Priority), dándole seguimiento inmediato hasta su resolución, dejando cualquier otra cosa que se este haciendo como pendiente, dado que en este tipo de problemas, se debe dar prioridad a los usuarios finales que utilizan los sistemas.

Dentro de la compañía existen herramientas para levantar y enviar tickets, cada uno de ellos se dirigen al equipo(s) requerido como se explicó anteriormente, asignándole un nivel de severidad según sea el caso, cuando algún miembro de los equipos lo recibe y procede a resolverlo, se recibe una alerta para darle seguimiento hasta su resolución y visto bueno para poder cerrar dicho ticket.

- **Soporte a Instalaciones y Mantenimientos.**

Las aplicaciones, durante el tiempo sufren cambios por parte del equipo de desarrollo o se generan nuevas aplicaciones, por lo que parte de las actividades son llevar a cabo el seguimiento del mantenimiento o instalación de nuevas aplicaciones, teniendo que ya sea una nueva aplicación o un nueva versión a una aplicación existente, en ambos casos, se levanta un ticket especial denominado CR (Change Request) el cual, describe de manera detallada todos los pasos necesarios para instalar/actualizar la aplicación dentro del ambiente destino, se deben involucrar a los distintos equipos de soporten dependiendo el tipo de mantenimiento junto con la agenda de fechas y horarios en los cuales se llevara a cabo. Previamente se realizan juntas de discusión entre todos los equipos involucrados para que den el visto bueno acerca de los cambios a realizarse, verificar disponibilidad durante las fechas acordadas y discutir cualquier inquietud que surja.

- **Soporte a Infraestructura.**

Cuando existen cambios en la infraestructura que impacta de manera directa o indirecta a las aplicaciones, por ejemplo, se puede recibir por orden administrativa, que todas las aplicaciones que se conecten a un motor de base de datos DB2, deban tener el ultimo fix pack de este motor de base de datos, por lo tanto, si bien la aplicación no se modifica en cuanto a código, se debe dar seguimiento a la actualización requerida con el fin de verificar que después de dicha actualización, la aplicación no fue afectada y continua trabajando correctamente. Para este tipo de soporte, también se generan documentos CR en los cuales se debe tener siempre en mente la fecha en la cual se llevaran a cabo las actualizaciones, para la verificación oportuna de las aplicaciones una vez realizadas. Otros ejemplos de este tipo de soporte, son cambios en la versión del servidor HTTP, actualizaciones al servidor de aplicaciones, parches al sistema operativo, etc.

CONCLUSIONES

Durante mi experiencia profesional, he tenido la satisfacción de contribuir en el desarrollo profesional de software para diversas compañías siempre a la espera de nuevos retos, parte importante de afrontarlos es mantener siempre un buen nivel de habilidades mediante la continua actualización.
Tengo el orgullo de haber pertenecido a las Facultad de Estudios Superiores Campus Aragón y por ende a la Universidad Nacional Autónoma de México donde he sido parte y es a quien le debo lo que hoy en día lo que soy como profesionista. El mundo de software es muy cambiante, siempre van apareciendo nuevas tecnologías y metodologías para hacer frente a los problemas actuales, el ingeniero en computación debe ser una persona capaz de poder hacer frente a cualquier problema para darle la mejor solución.

Sin duda, los capítulos anteriores reflejan un amplio recorrido de experiencia laboral la cual en algunas ocasiones han costado mucho tiempo, trabajo y dedicación, sin embargo, todo este costo es parte de ser un profesional y agradezco el haber podido participar en cada uno de los proyectos aportando mis conocimientos y experiencia además de seguir siempre en constante aprendizaje para poder seguir adelante a nuevos proyectos e implementación de nuevas soluciones cuando sean requeridas, las cuales, seguramente serán de gran utilidad para tanto usuarios como compañías que así las demandan, reflejándose en la reducción de sus tiempos de espera y la automatización de muchas de sus labores diarias.

BIBLIOGRAFÍA

- http://docs.oracle.com/javaee/6/api/javax/ejb/TransactionAttributeType.html

- Stephen Stelting, Olav Maassen
  Patrones de Diseño Aplicados a JAVA
  Pearson Pretince Hall
  ISBN 84-205-3839-6

- Antonio Goncalves
  Beginning Java2EE7
  Apress
  ISBN-13 (pbk): 978-1-4302-4626-8
  ISBN-13 (electronic): 978-1-4302-4627-5