UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIERÍA

"DOCUMENTACIÓN DE LA IMPLANTACIÓN DE UN SISTEMA WEB INTEGRAL DE APOYO A SERVICIOS FINANCIEROS"

MEMORIA

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN

PRESENTA

ERIC LÓPEZ FUENTES

ASESOR:

M. EN .A. SILVIA EDITH ALBARRÁN TRUJILLO

TOLUCA MÉXICO, JULIO 2013

DEDICATORIA

¡Gracias a Dios! por darme la oportunidad de vivir, cada día trataré de ser digno de la misión que me ha encomendado.

A mis papás que han sido y serán pilar importante les doy gracias por hacer de mi una persona de bien con principios que ustedes me han inculcado.

A mis hermanos José Antonio, Lupita, Jose Luis, Yeni y **Abraham de Jesús** les agradezco el apoyo que me han dado en momentos difíciles en donde hemos salido adelante.

¡Gracias! A mis abuelitos que han sido unos angeles que nos cuidan desde el cielo, no los olvidaré**.**

A mis profesores que me han apoyado en en mi formación académica, me hán compartido sus conocimientos y transmitido sus experiencias les estaré agradecidos por siempre.

Mi humilde agradecimiento especial a mi profesora y asesora **M. en A. Silvia Edith Albarrán Trujillo** por aceptar compartir esta experiencia y guiarme en este proyecto para realizar mi titulación.

ÍNDICE

INTRODUCCIÓN

Durante años los sistemas financieros han formado parte del cambio y crecimiento que ha tenido el país, aunque también en muchos casos han sido causantes de problemas ya que al no haberse actualizado y modernizado no tuvieron la capacidad para aplicar una administración de finanzas sanas, por lo que el país tuvo un atrazo tecnológico, por lo que muchos empresarios emprendedores dueños de los grupos financieros decidieron modernizar los sistemas bancarios con el propósito de brindar una mejor administración de capitales e inversiones, ofrecer mejores servicios de calidad, más productos con características que beneficiaran a los clientes aperturando y liberando créditos que detonaran el crecimiento económico y laboral generando un mayor consumo de bienes y servicios, por lo tanto, el banco al que haré mención en esta memoria es parte del Grupo Financiero (GF) con un 20% de capital extranjero y 80% nacional el cual decidió modernizar sus sistemas.

El objetivo general de mi memoria es describir las experiencias adquiridas durante el desarrollo del sistema web Integral de las áreas de Banco, Seguros, Afore como apoyo a la prestación de servicios financieros.

Uno de los principales servicios que ha proporcionado el banco han sido los créditos al sector empresarial solo que para regular estos servicios se han aplicado reglamentos de la Secretaría de Hacienda y Crédito Público (SHCP) y el Banco de México (BANXICO) que han hecho que los sistemas mantuvieran una constante actualización y expansión (Montes de Oca, Latapí y Boni, 2010).

Como el GF quería ser competitivo tuvo que invertir en tecnología, infraestructura y capital humano, para esto la dirección general decidió llevar acabo un análisis para aplicar una reingeniería a los sistemas apoyándose en metodologías y tecnologías modernas, esto lo redactaré a detalle incluyendo también los principales problemas que existieron en la organización como los altos costos de mantenimiento al Sistema Instanteneo Mundial de Pago Unico (IWB) que operaba antes de la transformación tecnológica que vivió el banco, había poca oferta en productos para clientes corporativos y personales que limitaba el crecimiento de la cartera de clientes, el atraso tecnológico era grande, no había crecimiento de capital humano ni generación de talento interno a la institución, los servicios de atención al cliente eran lentos, ante esto la Dirección de Sistemas decidió que se desarrollara un nuevo sistema para resolver este problema, además debía reducir los riesgos de seguridad contando con nuevos procedimientos, especificaciones y estándares que se implementaban para asegurar que las operaciones realizadas fueran eficientes, rápidas y que garantizaran la confidencialidad en los datos de los clientes.

Enfocándome al objetivo principal de esta memoria explicaré los conocimientos aprendidos durante el desarrollo del sistema web, para que se tenga una idea clara de cómo lo realicé, en qué participé y por medio de esta se pueda tener una visión clara de lo que significó desarrollar el sistema integral web, de lo que implicaron mis responsabilidades que existieron en cada fase del proyecto así como mi participación en la organización, planeación, distribución de asignaciones en equipos de trabajo, en las fases de aplicación de tecnología, en el desarrollo del código, en las pruebas, en la evaluación del sistema así como en las tecnoloigías, arquitecturas y patrones de diseño que implementé.

En el año 2007 me integré al equipo de trabajo y cinco meses después de haber ingresado a la institución, se liberó el sistema a producción, me tocó participar en esta liberación que en un principio contó con módulos de operación simples e incompletos donde participé completando su desarrollo aplicando la metodología planteada por los directivos de sistemas, también desarrollé nuevos procesos y módulos completos los cuales haré énfasis en este documento.

Esta memoria está formada de cuatro capítulos:

En el capítulo uno, abordaré la constitución de la empresa, evolución, antecedentes y descripción de los problemas y limitaciones existentes del sistema IWB que operaba en el banco así como las alternativas de solución.

En el capítulo dos explicaré la propuesta de solución, la metodología y su aplicación, las plataformas tecnológicas, los patrones de diseño, estándares de codificación y paradigmas de desarrollo que apliqué.

En el capítulo tres, explicaré mi participación en el desarrollo, el orden en que lo realicé, los diagramas UML que generé, me enfocaré en tres módulos que formaron parte del sistema integral los cuales fueron la ventanilla divisas, monitor efectivo ventanilla y operaciones intermediación.

En el capítulo cuatro mencionaré mi participación en la evaluación al sistema así como las desiciones estratégicas llevadas acabo para obtener resultados satisfactorios.

La parte de anexos está formada por cuatro apéndices:

Apéndice A. RECOMENDACIONES, mencionaré aspectos para contar con un buen modelo de clases y de entidad-relación, detallaré la nomenclatura de nombrado estándar del código del proyecto así como la ergonomía que apliqué en el diseño para que el sistema tuviera una vista amigable y fácil de entender para el usuario.

Apéndice B. RESULTADO DE COMPROBANTES Y REPORTES, presentaré algunas imágenes, la estructura, diseño de los documentos y comprobantes generados para el usuario.

Apéndice C. FORMATOS DEL MATERIAL DE LA METODOLOGÍA, integraré los formatos que se llenaron como parte de la metodología con el fin de ver el contenido requerido por el usuario.

Apéndice D. COMPILACIÓN Y CONFIGURACIÓN DE NEGOCIO Y BATCH, presentaré en imágenes el contenido de los archivos que ejecuté para llevar acabo la compilación del proyecto.

CAPÍTULO 1

ANTECEDENTES Y DESCRIPCIÓN DEL PROBLEMA

En este capítulo redactaré los inicios del grupo financiero, las gerencias que participaron en el proyecto entre ellas la gerencia en donde tuve participación, las actividades, conformación, integrantes y tareas que se realizaban, el crecimiento e infraestructura de la empresa, los problemas que existieron con el sistema anterior IWB, además explicaré las alternativas de solución, las limitaciones que impedían el crecimiento de la empresa, estos aspectos los menciono enseguida.

1.1 CONSTITUCIÓN E INICIO DEL GRUPO FINANCIERO

El 20 de mayo de 1985 Inversora bursátil hoy en día Casa de Bolsa y Seguros de México, se transformaron en subsidiarias, posteriormente se unieron formando el Grupo Financiero (GF), el 6 de Septiembre de 1993 se obtuvo la autorización por parte de la SHCP para realizar las operaciones de institución de banca múltiple, en 1996 se integró Afore, en 2007 se incorporó la Impulsora del Desarrollo y el Empleo de América Latina (IDEAL), en 2008 una caja española adquiere 20% de las acciones del GF, en la figura 1.1 presento este crecimiento.

El GF ha incorporado nichos de negocio adicionales tal como la administración de cuentas de afore a través de la subsidiaria Afore a partir de 1996 y la operación de una sociedad de inversión de capitales, posteriormente en el año 2000 comenzó a ofrecer productos bancarios a través del formato de menudeo buscando fortalecer e incrementar aún más su posición en el mercado mexicano, esto llevó a la creación de productos que tuvieran un enfoque más amplio, acentuándose en los utilizados en conjunto para que generaran valor al cliente (Montes de Oca, Latapí y Boni, 2010).

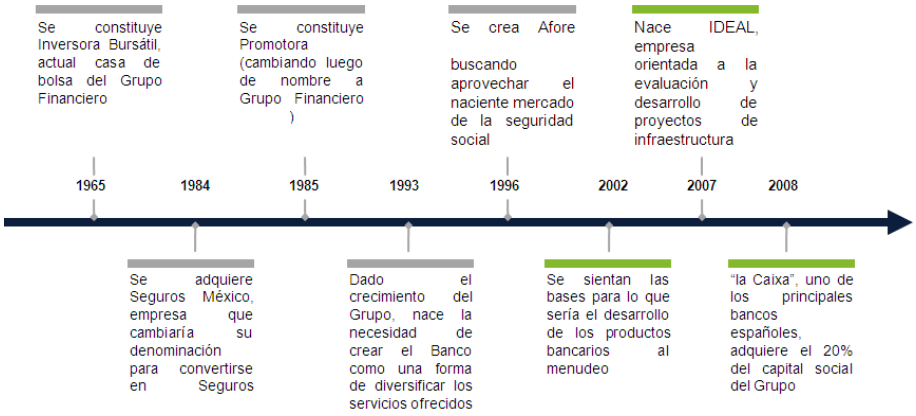


Figura 1.1: Progreso estructural del Grupo Financiero (Montes de Oca, Latapí y Boni, 2010).

Estructura del Grupo Financiero

El Grupo Financiero es una sociedad controladora de acciones de empresas que prestan servicios financieros (Montes de Oca, Latapí y Boni, 2010). En la figura 1.2 presento la jerarquía y enmarco el área en la cual participé, tuve comunicación e intercambié conocimientos.

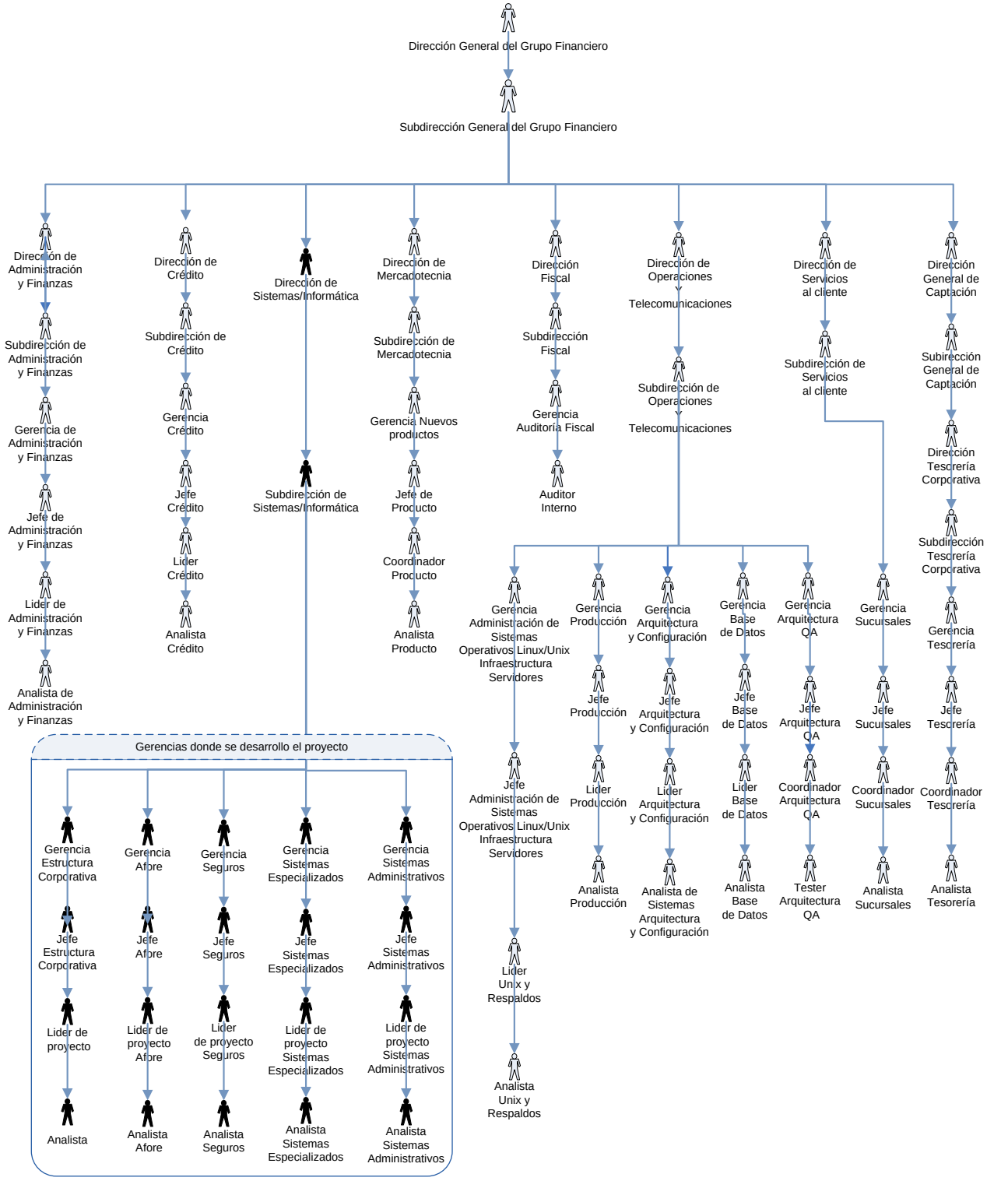


Figura 1.2: Organigrama del Grupo Financiero (Documento interno GF, Procesos: 2010)

1.2 GERENCIAS PARTICIPANTES EN EL DESARROLLO DEL SISTEMA INTEGRAL

Las gerencias que participaron el desarrollo del sistema pertenecían a la Dirección de Sistemas, cada gerencia desarrollaba su propios proyectos, la gerencia de Sistemas Administrativos se encargaba del desarrollo y mantenimiento del sistema de banco, la gerencia de Sistemas especializados, se encargaba del desarrollo y mantenimiento de sistemas para clientes preferenciales, la gerencia de Afore se encargaba del desarrollo y mantenimiento de sistemas para la administración de Afores, la gerencia de seguros se encargaba del desarrollo y mantenimiento de sistemas que administraban los seguros de personas o empresas, la gerencia de estructura corporativa desarrollaba sistemas para administrar y mantener integra la información del GF, estas gerencias las detallo a continuación.

Gerencia de Sistemas Administrativos

Se encargaba del análisis, diseño, desarrollo, mantenimiento, solución de incidencias, escalamiento de los nuevos desarrollos del banco así como el desarrollo de procesos entre el banco y los otros sistemas como Afore seguros y Crédito, laboraba un gerente, cuatro jefes seis líderes y cuarenta y nueve analistas. Las actividades que se realizaban eran:

a) Análisis de factibilidad, diseño y desarrollo de nuevos proyectos, solución de incidencias en ambiente desarrollo, usuario y producción, configuraciones a nivel desarrollo de los servidores locales de cada desarrollador.

b) Análisis y optimización de consultas a la base de datos, pruebas de stress, mantenimiento a nuevos desarrollos de Ventanilla de Banco, Portal, Cajeros ATM (Cajeros automáticos mediante transacciones asíncronas), Call Center, Ventanilla Universal, Ventanilla Express, Ventanilla Divisas y Operaciones Intermediación.

Gerencia de Sistemas Especializados

Tenía a su cargo el Sistema de Operaciones Internacionales (SOI) el cual realizaba todas las operaciones con productos que involucraron divisas internacionales, el otro sistema era Créditos donde se daban de alta o se pactaban créditos con los clientes, se generaban amortizaciones de capital, alta de formas de pago, registro de firmas, consulta de créditos, vencimiento de créditos, renovación de créditos, capitalización de créditos, convenios con tarjetas de crédito emitidas por empresas del GF y tarjetas emitidas por American Express®. Otras actividades que realizaban fueron la solución de incidencias del Sistema SOI, reportes de corresponsales, reportes de cheques de viajero, de giros, de remesas, de emisores, de cuenta de cheques, de operaciones y supervisión de operaciones de Intermediación que recibía el sistema SOI. Esta gerencia contó con un gerente, dos jefes, un líder y seis analistas.

Gerencia de Afore

Se integraba por un gerente, dos jefes, tres líderes y nueve analistas, en total eran quince personas, se encargaban del desarrollo mantenimiento y expansión del sistema Afore, los desarrolladores compartían requerimientos con los desarrolladores de la gerencia de Sistemas administrativos para aclaración de dudas de los nuevos desarrollos, otras de sus actividades fueron:

a) Generar reportes de información, desarrollar procesos automatizados para cierre mensual.

b) Desarrollo de actualizaciones y mantenimiento del sistema de afore, cuando del lado de banco se realizaban aportaciones eran responsables de reflejarlas del lado de afore.

Gerencia de Seguros

Se integraba por un gerente, tres jefes, cuatro líderes y diecisiete analistas, compartían requerimientos con la gerencia de Sistemas administrativos, desarrollaba reportes para clientes, desarrollaba procesos de seguros autos, seguros vida, seguros educa y seguros médicos.

Gerencia de Estructura Corporativa

Conformada por un gerente, un jefe, un líder, diez analistas, tenía el control de la información del GF, ya que si el sistema de Banco, Seguros o Afore querían ver ciertos datos, debía realizar una solicitud donde justificara el uso de los datos ya que se debía cumplir con la seguridad correspondiente para que esta información fuera visible, también realizaba la depuración, integración, unificación la información de los clientes, optimizaba consultas a la base de datos, migraba la información de una base de datos a otra, daba de alta a usuarios y definía su rol. En el siguiente apartado detallaré el crecimiento de la organización.

1.3 CRECIMIENTO E INFRAESTRUCTURA

En Diciembre del 2008, se llegó a tener 96 oficinas comerciales, 591 cajeros automáticos con una base de más de 6.7 millones de clientes. Con la finalidad de continuar con dicho crecimiento, durante el 2008 el GF alcanzó un acuerdo con una caja española, para que éste último adquiriera el 20% del capital social del grupo (*Aguado, 2010).* En diciembre de 2009, se llegó a 198 oficinas comerciales, 689 cajeros automáticos y se contaba con una base de más de 7.2 millones de clientes (Aguado, 2010). En el año 2009, la Comisión Nacional Bancaria y de Valores (CNBV) autorizó la operación a través de corresponsales con el objetivo de incrementar la oferta de servicios financieros ofrecidos en masa al autorizar operaciones financieras fuera de sucursales bancarias, por ello, los clientes del banco podían realizar los pagos de créditos y retiros de efectivo a través de las sucursales de Telecomm Telégrafos y tiendas departamentales, se contaba con 1,577 y 148 establecimientos respectivamente (Montes de Oca, Latapí and Boni, 2010).

Hardware utilizado

El GF a través de la subsidiaria de seguros contaba con tres servidores Hewlett Packard HP RP4440, tres servidores Hewlett Packard HP Rx4640, un Servidor Hewlett Packard HP Rx8620, dos Servidores Hewlett Packard HP Rx6660 y un Servidor Hewlett Packard HP Súper Dome, en ellos se tenían las aplicaciones de producción del sistema IWB, Seguros, Afore, el sistema SOI al nivel de Usuario y Producción; para desarrollo, el sistema SOI se encontraba en dos PC Pentium®, se contaba con un sistema de portal independiente del sistema IWB, este sistema de portal también se alojaba en los servidores, a nivel producción y usuario, para a nivel desarrollo portal estaba en una PC Pentium®; para los empleados y desarrolladores se contaba con 3500 computadoras de escritorio, cada sucursal contaba con un servidor

Intel® Pentium® (Martínez, 2008). Los problemas existentes en la organización eran grandes que limitaban el crecimiento tal como lo detallo a continuación.

1.4 PROBLEMÁTICA EN EL GRUPO FINANCIERO

El sistema IWB que operaba, sólo ofertaba productos como cuentas empresariales, créditos corporativos y servicios financieros como depósitos y cargos a cuentas de clientes morales, se podían realizar transferencias con las cuentas tanto en ventanilla como portal, cada sucursal contaba con una copia del sistema IWB tanto para oficinas de banco como de seguros, el identificador único de una transacción realizada en el sistema IWB era diferente al sistema SOI, al de Portal, al de Seguros y al de Afore, estos sistemas eran independientes.

Las sucursales operaban con el sistema IWB y si el cliente quería realizar operaciones relacionadas con algún seguro debían ir a una sucursal que tuviera el sistema de seguros, los sistemas de afore y SOI se encontraban en las sucursales de banco que contaban con el sistema IWB, pero si los clientes querían realizar operaciones con divisas internacionales, los cajeros debían salir del sistema IWB y posteriormente debían ingresar al sistema SOI, esto hacía muy lento el servicio. Algunos límites que se empezaron a encontrar fueron el número de registros para la transacción, cuando se llegaba al límite de almacenamiento que era de tipo numérico de 9 dígitos, se debían depurar, existían problemas con la concurrencia ya que cuando dos o más transacciones se realizaban al mismo tiempo, se guardaban en la base de datos con la misma hora, minuto, segundo y los reportes de las sucursales al final del día las cantidades entre lo recibido y entregado por el cajero al cierre del día no quedaban en cero, a esto se le llamaba descuadre, por lo que se tenía que conciliar la información de las operaciones realizadas.

Operaciones permitidas y no permitidas

El sistema sólo permitía realizar depósitos y retiros de cuentas, pagos de créditos corporativos, en cuanto a sistema de portal, se podía realizar traspasos bancarios, pagos de créditos corporativos y personales, en cajeros ATM se podían realizar retiros a cuenta y consulta de saldos, además no permitía realizar pagos de servicios como teléfono, Agua, luz, tarjetas de crédito en dólares, tarjetas de crédito en moneda nacional, pago de seguros, aportaciones Afore, pago de tarjetas de crédito departamental, pago de impuestos locales, pago de impuestos federales, cable y telefonía celular.

Seguridad

Para acceder al sistema los cajeros avisaban a los jefes de sucursal ya que los procedimientos y formatos de acceso y salida al sistema se realizaban de forma manual, no se definían ni asignaban perfiles y roles, no se sabía con precisión si un usuario (cajero o jefe) de una sucursal se transladaba a otra y no se tenía la información de que ya había renunciado a la empresa, no se contaba con privilegios, no había bitácora de acceso al sistema en consultas, actualizaciones y eliminaciones por parte del usuario, no se sabía si los cajeros intercambiaban los equipos, la información clasificada como cantidades de saldos siempre estaba a la vista de cajero.

Servicio al cliente

El tiempo de servicio era tardado ya que los cajeros se tenían que firmar en diferentes sistemas y perdían tiempo. El cliente esperaba demasiado tiempo en la sucursal, el cliente no estaba satisfecho con el servicio ya que cuando solicitaba aclaraciones de sus movimientos a cuenta, no se detallaban los depósitos o retiros como el cliente lo quería.

Reportes y comprobantes

Los comprobantes de los depósitos, abonos, pagos de créditos corporativos, se llenaban manualmente por los cajeros, estos comprobantes eran formatos previamente impresos, no los imprimía el sistema, las órdenes de pago también las llenaba el cajero de forma manual, tenían folio pero era parte de la plantilla, no había reportes de movimientos realizados por los cajeros, no se podían vender cheques de caja por medio del sistema, no había forma de imprimirlos, los comprobantes de las operaciones no se registraban con identificador de transacciones, la dotación y recepción de efectivo de la bóveda y las cajas no se realizaba en el sistema, no se certificaban cheques, las cuentas de clientes corporativos no contaban con chequeras, tarjetas de débito ni tarjetas prepago para peaje en autopistas (IAVE). Para resolver los problemas, los directivos plantearon soluciones que explico en el siguiente apartado.

1.5 ALTERNATIVAS DE SOLUCIÓN

Ante las limitaciones tanto en operación, tecnología, servicio, productos e infraestructura, existieron diferentes alternativas de solución analizadas por la junta directiva del GF. Con base en el análisis de viabilidad, factibilidad y operatividad realizado por la dirección de sistemas la cual presento en la tabla 1.1 donde describo las limitaciones y desventajas de operación que tenía el sistema IWB frente a la competencia, la Dirección general del GF consideró que no se continuara operando con ese sistema.

Tabla 1.1: Funcionamiento del sistema IWB (Procesos GF, 2006)

Tema	Descripción
Productos	Se ofrecían pocos productos como cargo a cuenta, depósito a cuenta y pago de crédito.
Clientes	El tipo de clientes que se tenían eran solo empresas.
Arquitectura del sistema	El sistema estaba creado con la arquitectura cliente-servidor en cada una de las sucursales.
Usuarios	Los usuarios que existían eran sólo cajeros y jefes de sucursal.
Operatividad	La atención al cliente era lenta ya que al realizar depósitos la operación se realizaba en el sistema IWB y cuando los clientes también querían realizar una operación con divisas, el cajero tenía que salir del sistema IWB e ingresar al sistema SOI y si el cliente quería también realizar un pago de seguros, el usuario se tenía que cambiar al sistema de Seguros, todo esto hacía que el tiempo de servicio se incrementara.
Comprobantes y entregables	Los comprobantes y formatos que se entregaban a los clientes eran llenados manualmente por los cajeros.
Seguridad	El sistema no contaba con una clasificación de roles, permisos y categorías de usuarios, el acceso al sistema lo supervisaban los jefes de sucursal.
Operaciones por transacción	No existía clasificación de operaciones de entrada y salida, esto es que solo se realizaba una operación por transacción, el cajero es el que se encargaba de complementar la operación de entrada con otra transacción que tuviese una operación de salida.
Atención al cliente	No había forma de darle un servicio de aclaraciones al cliente con un detalle suficiente de las operaciones realizadas con sus cuentas.

Las alternativas posibles fueron postuladas por la Dirección de Sistemas, la Dirección de Organización y Métodos posteriormente presentadas a la Dirección General del GF quien tenía el control de todos los proyectos de desarrollo así como la toma de decisión sobre los mismos, estas alternativas fueron:

1. **Reingeniería.**

   La Dirección de Sistemas decidió aplicar esta alternativa ya que permitió generar capital humano además de talento interno al GF con el suficiente conocimiento tanto de negocio como tecnológico en diferentes herramientas, metodologías, paradigmas, plataformas con desarrollo de código abierto asegurándose de que el sistema fuese extensible, expandible, que su mantenimiento no fuese costoso y que cumpliera con los requerimientos del usuario.

2. **Actualización.**

   Debido a que la actualización del sistema IWB requería un soporte amplio y constante por parte del proveedor a tal grado de seguir dependiendo del mismo y como consecuencia generando altos costos, esto fue factor para que la Dirección de Sistemas no optara por esta alternativa de actualizar el sistema IWB.

3. **Compra de Software.**

   En base a la experiencia y resultados obtenidos a través de los años en los que había estado operando el sistema IWB, a la Dirección de Sistemas no le pareció realizar la adquisición de otro software para implementarlo por que limitaría el crecimiento de la organización.

Como la desición tomada por la Dirección General del GF fue la reingeniería, esta debía contar con una metodología adecuada y acorde a las exigencias, tamaño, visión y proyección de la organización, debía contar con una arquitectura suficiente, partadigmas, estándares, patrones y marcos de trabajo reconocidos a nivel internacional, todos estos aspectos los puntualizo en el siguiente capítulo dos.

CAPÍTULO 2

PROPUESTA DE SOLUCIÓN

En este capítulo mencionaré el contenido de la propuesta de solución elegida por la Dirección de Sistemas, Dirección de Organización de Métodos del GF, explicaré la metodología detallando mis decisiones y aportaciones en las fases donde tuve participación al momento de su aplicación, en las fases donde tuve participación y aportación mencionaré que yo lo realicé, además de los paradigmas, la arquitectura, estándares, patrones de diseño y marcos de trabajo que implementé.

2.1 METODOLOGÍA

La Dirección de Sistemas, Dirección de Organización y Métodos propusieron la metodología basada en la estandarización de procesos de ingeniería del software llamada Proceso Unificado Rational (RUP) la cual fue iterativa e incremental permitiendo asignar tareas y responsabilidades dentro de la organización del equipo de desarrollo y poder generar un sistema adecuado de calidad de una forma rápida y con menor costo (Documento interno, Jorge Delgado, 2005). Las metas perseguidas con el uso de la metodología fueron:

a) Satisfacer al usuario realizando el proyecto con los requerimientos dentro del tiempo estimado.

b) Minimizar riesgos.

c) Especificar indicadores de puntos clave en el desarrollo del proyecto.

d) Estandarizar el trabajo en cada proyecto.

e) Estandarizar la documentación.

f) Proveer la comunicación entre el personal de sistemas.

g) Generar un apoyo para seleccionar el sistema correcto a instalar.

La metodología permitió a la organización una estructura homogénea, cualidad que se incrementó mientras mayor número de personas participaron en el proyecto, además aseguró los pasos a seguir y los resultados a obtener logrando que la dependencia en la gente se redujera debido a la estandarización y documentación, otro aspecto importante de la metodología fue que facilitó la realización del proyecto dentro del tiempo estimado, el desarrollo del sistema contó con las siguientes fases:

- ✓ Planeación y análisis.

    a) Se analizaron las necesidades de información de la organización.

    b) Se desarrolló una estrategia de tecnología de la información.

    c) Se preparó un reporte de definición e identificación de proyectos.

- ✓ Diseño.
    a) Se diseñó el sistema desde un punto de vista funcional y técnico.
    b) Se proporcionó suficiente detalle para dar una estimación exacta del costo y esfuerzo en la instalación.
- ✓ Desarrollo e implementación.
    a) Se completó el diseño detallado del sistema.
    b) Se transladó el diseño del sistema a un ambiente total de operación del mismo.
    c) Se desarrollaron los procedimientos y el entrenamiento al usuario.
- ✓ Soporte y mantenimiento de sistemas.
    a) Analizar, definir la prioridad e implementar los requerimientos de cambio.
    b) Monitorear y evaluar periódicamente el sistema.
    c) Planear un nuevo sistema cuando sea requerido.

Enfoques

La metodología incluyó tres enfoques para las fases de diseño, desarrollo e implementación:

a) **Sistemas a la medida**. Las necesidades de negocios y de información fueron únicos.
b) **Sistemas de paquetes**. Las necesidades de la organización se cubrieron por paquetes es decir por hardware y software seleccionado.
c) **Desarrollo iterativo e incremental**. Los requerimientos se modelaron en iteraciones de forma repetitiva y creciente.

Material de metodología

La Dirección de Organización y Métodos controló el material de la metodología que se integró de:

a) **Formatos**. Fueron los esqueletos (documentos vacios) donde se añadieron datos requeridos por la metodología.
b) **Documentos**. Fueron los formatos llenados con la información necesaria.

2.2 APLICACIÓN DE LA METODOLOGÍA

La metodología RUP utilizada para el desarrollo del SI fue definida, estructurada, creada y controlada por el área de procesos, fue revisada por el comité de sistemas, todas estas áreas pertenecían al GF, esta metodología se integró de las siguientes fases:

- ✓ **Planeación y análisis**

Esta fase inicial fue considerada la base principal ya que al haber realizado una buena planeación y análisis, no se tuvieron tantos problemas para la construcción e implementación mediante la aplicación del siguiente procedimiento:

1. **Objetivos**. Se definieron para identificar las necesidades de información del negocio estableciendo las bases para la aprobación e inicio del proyecto.

2. **Las actividades fueron tres**:

   a) **Definición del proyecto**. Se incluyó el trabajo requerido para obtener la aprobación del proyecto, la planeación del esfuerzo necesario incluyendo la definición del alcance, el desarrollo del plan de trabajo, la organización y entrenamiento del equipo de trabajo.

   b) **Diseño conceptual**. Se identificaron las principales entradas y salidas, las interfaces y datos del sistema así como las funciones y el ambiente físico necesario para soportar la arquitectura.

   c) **Aprobación de la dirección**. El equipo del proyecto presentó a la dirección o comité la definición y resultados planeados en donde la presentación incluyó el diseño conceptual propuesto, el enfoque, alcance, tiempo, costo y plan de trabajo.

3. **Insumos**. Consistieron en la planeación estratégica del negocio y de los planes de información.

4. **Productos**. Se presentaron reportes para definir el proyecto el cual incluyó la descripción, beneficios del SI propuesto, plan de trabajo, costo de implementación, impacto del sistema, requerimientos de hardware, software y personal humano para desarrollar el SI.

5. **Factores críticos**. Fueron visión, misión y objetivos bien definidos, compromiso de los participantes, oportunidades estratégicas, planeación de la capacidad, infraestructura de comunicaciones, requerimientos de seguridad y control interno.

6. **Plan de trabajo**. Se integró de tres actividades:

   a) **Organización del proyecto**. Se definieron objetivos, se detalló el plan de trabajo, los documentos generados fueron PT-PLANEACION (Plan de trabajo en MS Project).

   b) **Planeación y análisis**. Se revisaron las funciones, documentación, reportes, pantallas, formatos, objetivos y estrategias de información, se definieron requerimientos de seguridad e información, se analizó la arquitectura que se tenía, los planes de hardware, software y comunicaciones, se identificaron las arquitecturas potenciales en las que se podía desarrollar el sistema, se definieron funciones, se identificaron interfaces, entradas y salidas de datos, se realizó estimación y se revisaron con los usuarios y la dirección, los enfoques, costos, tiempos y beneficios.

c) **Revisión y aprobación**. Se realizó la presentación del proyecto, recepción de observaciones y su aprobación.

- ✓ **Diseño del sistema a la medida como parte del desarrollo**

Esta fase permitió tener un esquema general, contar con un panorama real del tamaño del proyecto tomando en cuenta los costos y beneficios basados en la recopilación de requerimientos e inquietudes de los usuarios, el procedimiento fue el siguiente:

1. **Objetivos**. Se especificaron para definir el sistema y sus costos de instalación además de dar al equipo de trabajo el conocimiento necesario para implementar el sistema.

2. **Entregables**. Se integró del diseño funcional basado en los requerimientos del usuario, se realizó un reporte de especificaciones y un prototipo funcional, se completó el diseño técnico para estimar la cantidad de trabajo, se creó un plan de instalación, se realizó un reporte gerencial describiendo costos y beneficios.

3. **Plan de trabajo**. Se conformó de ocho actividades:

   a) **Organización**. Se defininieron objetivos y se detalló el plan de trabajo, los documentos generados fueron PT-MEDIDA, BI-PT (Administración del Plan de trabajo).

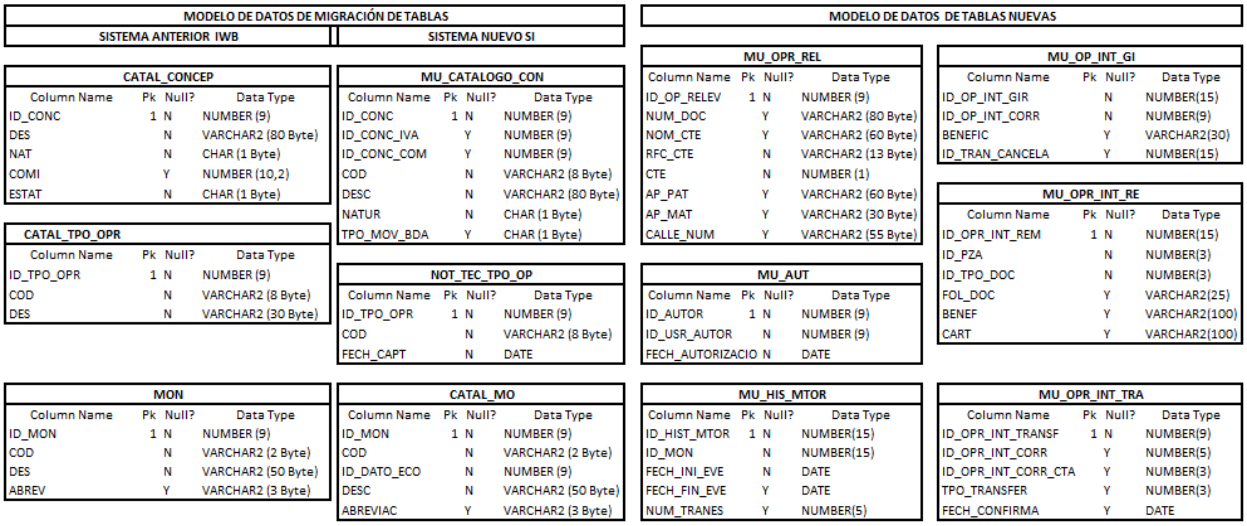
   b) **Requerimientos de usuario**. Se revisó la documentación del sistema, se realizaron entrevistas, se identificaron requerimientos funcionales que ayudaron a definir funciones del sistema y su descomposición, revisé con los usuarios las nuevas funciones del proyecto para tener mayor cantidad de productos a ofrecer al cliente final.

   c) **Diseño usuario**. Se definieron entradas y salidas para definir, diseñar reportes, pantallas y formatos requeridos, se definieron funciones de proceso que integraron el flujo de la información por función, además de que se identificaron interfaces.

   d) **Diseño técnico**. Diseñe parte de la base de datos, generé el documento BI-ET (Especificación técnica) donde sugerí revisar la integridad referencial de las tablas de la base de datos, nombres de campos y tablas para el nuevo sistema, posteriormente el área de base de datos realizó la migración de tablas ejecutando sentencias SQL contenidas en archivos script con extensión SQL.

      Los principales problemas que encontré en la migración de la base de datos fueron la compatibilidad de los tipos, la longitud de los campos tanto numéricos como alfanuméricos, para las tablas nuevas creadas no se presentaron problemas, para las que cambiaron de nombre tuve que ajustarlos, para todos estos cambios realicé scrips SQL de actualización y creación de objetos de base de datos, estos scripts los ejecutó el área de base de datos a primeras horas del día sábado que es el horario donde el sistema tenía el mínimo de operación y actividad permitiendo la migración, para la segunda parte del diseño estructuré las nuevas tablas con sus respectivos nombres de

campos y tipos de datos de cada una de ellas, en la figura 2.1 presento el modelo de datos de tablas migración y tablas nuevas del sistema SI.



MODELO DE DATOS DE MIGRACIÓN DE TABLAS

SISTEMA ANTERIOR IWB

CATAL_CONCEP

Column Name	Pk	Null?	Data Type
ID_CONC	1	N	NUMBER (9)
DES		N	VARCHAR2 (80 Byte)
NAT		N	CHAR (1 Byte)
COMI		Y	NUMBER (10,2)
ESTAT		N	CHAR (1 Byte)

CATAL_TPO_OPR

Column Name	Pk	Null?	Data Type
ID_TPO_OPR	1	N	NUMBER (9)
COD		N	VARCHAR2 (8 Byte)
DES		N	VARCHAR2 (30 Byte)

MON

Column Name	Pk	Null?	Data Type
ID_MON	1	N	NUMBER (9)
COD		N	VARCHAR2 (2 Byte)
DES		N	VARCHAR2 (50 Byte)
ABREV		Y	VARCHAR2 (3 Byte)

SISTEMA NUEVO SI

MU_CATALOGO_CON

Column Name	Pk	Null?	Data Type
ID_CONC	1	N	NUMBER (9)
ID_CONC_IVA		Y	NUMBER (9)
ID_CONC_COM		Y	NUMBER (9)
COD		N	VARCHAR2 (8 Byte)
DESC		N	VARCHAR2 (80 Byte)
NATUR		N	CHAR (1 Byte)
TPO_MOV_BDA		Y	CHAR (1 Byte)

NOT_TEC_TPO_OP

Column Name	Pk	Null?	Data Type
ID_TPO_OPR	1	N	NUMBER (9)
COD		N	VARCHAR2 (8 Byte)
FECH_CAPT		N	DATE

CATAL_MO

Column Name	Pk	Null?	Data Type
ID_MON	1	N	NUMBER (9)
COD		N	VARCHAR2 (2 Byte)
ID_DATO_ECO		N	NUMBER (9)
DESC		N	VARCHAR2 (50 Byte)
ABREVIAC		Y	VARCHAR2 (3 Byte)

MODELO DE DATOS DE TABLAS NUEVAS

MU_OPR_REL

Column Name	Pk	Null?	Data Type
ID_OP_RELEV	1	N	NUMBER (9)
NUM_DOC		Y	VARCHAR2 (80 Byte)
NOM_CTE		Y	VARCHAR2 (60 Byte)
RFC_CTE		N	VARCHAR2 (13 Byte)
CTE		N	NUMBER (1)
AP_PAT		Y	VARCHAR2 (60 Byte)
AP_MAT		Y	VARCHAR2 (30 Byte)
CALLE_NUM		Y	VARCHAR2 (55 Byte)

MU_AUT

Column Name	Pk	Null?	Data Type
ID_AUTOR	1	N	NUMBER (9)
ID_USR_AUTOR		N	NUMBER (9)
FECH_AUTORIZACIO		N	DATE

MU_HIS_MTOR

Column Name	Pk	Null?	Data Type
ID_HIST_MTOR	1	N	NUMBER(15)
ID_MON		N	NUMBER(15)
FECH_INI_EVE		N	DATE
FECH_FIN_EVE		Y	DATE
NUM_TRANES		Y	NUMBER(5)

MU_OP_INT_GI

Column Name	Pk	Null?	Data Type
ID_OP_INT_GIR		N	NUMBER(15)
ID_OP_INT_CORR		N	NUMBER(9)
BENEFIC		Y	VARCHAR2(30)
ID_TRAN_CANCELA		Y	NUMBER(15)

MU_OPR_INT_RE

Column Name	Pk	Null?	Data Type
ID_OPR_INT_REM	1	N	NUMBER(15)
ID_PZA		N	NUMBER(3)
ID_TPO_DOC		N	NUMBER(3)
FOL_DOC		Y	VARCHAR2(25)
BENEF		Y	VARCHAR2(100)
CART		Y	VARCHAR2(100)

MU_OPR_INT_TRA

Column Name	Pk	Null?	Data Type
ID_OPR_INT_TRANSF	1	N	NUMBER(9)
ID_OPR_INT_CORR		Y	NUMBER(5)
ID_OPR_INT_CORR_CTA		Y	NUMBER(3)
TPO_TRANSFER		Y	NUMBER(3)
FECH_CONFIRMA		Y	DATE

Figura 2.1: Modelo de datos migración de tablas (Sistemas Administrativos GF, Eric López Fuentes, 2007).

e) **Diseño de la conversión (liberación).** Diseñé procedimientos de prueba para identificar las fuentes de conversión, determiné programas de prueba, los documentos generados fueron BI-PC (Especificación de pruebas y conversión).

f) **Plan de instalación**. Sugerí la creación de un manual de instalación con una dimensión del esfuerzo requerido estableciendo los requerimientos del personal, los documentos generados fueron PT-MEDIDA, BI-PT, BI-EE (Estimación del esfuerzo).

g) **Análisis de costo beneficio**. Se realizó el análisis de los costos, beneficios y estimación de tiempos del desarrollo, los documentos generados fueron BI-ACB (Análisis costo-beneficio).

h) **Revisión y aprobación**. Se llevó a cabo la aprobación del diseño por parte del usuario, QA y soporte técnico, también se realizaron resúmenes del diseño, especificaciones funcionales, no funcionales y técnicas, se realizaron presentaciones, se recibieron las observaciones y se realizó la aprobación, los documentos generados fueron BI-ET.

✓ **Instalación del sistema a la medida como parte del desarrollo**

Participé en el entrenamiento a los usuarios sobre la operatividad del nuevo sistema ya que ellos fueron quienes lo utilizaron, para esta fase se aplicó el siguiente procedimiento:

1. **Objetivos**. Se establecieron para la implemenación del sistema (diseño detallado, programación, pruebas y conversión).

2. **Entregable**. Se completó el diseño técnico, se dividió cada programa del sistema en módulos, se revisaron las especificaciones de cada programa, se confirmó la estimación de los costos, se generaron los programas para el sistema, se realizaron los procedimientos de entrenamiento a usuarios en el nuevo sistema y se probó el sistema.

3. **Plan de trabajo**. Se conformó de ocho actividades:

   a) **Organización**. Se completó el programa de trabajo, se detalló plan de la instalación, capacité al nuevo equipo de trabajo los cuales apoyaron en el desarrollo del código del sistema ya que no tenían muchos conocimientos del software que se instaló así como de las herramientas y configuraciones aplicadas, los documentos generados fueron PT-MEDIDA, BI-PT.

   b) **Diseño detallado.** Se completó el diseño técnico y se incorporaron puntos nuevos de diseño, los documentos realizados fueron BI-ET.

   c) **Programación**. Se generó el soporte del desarrollo estableciendo la supervisión y control de las actividades de programación java en Eclipse, organicé a grupos de trabajo para que se desarrollara el nuevo proyecto en forma distribuida de manera que cada desarrollador supiera cúal era la asignación que le correspondía, también realicé scrips de pruebas para evaluar el desempeño del sistema, se realizaron pruebas unitarias para verificar resultados reales con los esperados y que el usuario viera estas pruebas, los documentos generados fueron PT-MEDIDA, BI-PT.

   d) **Preparación de la conversión**. Junto con el área de arquitectura completé el plan de conversión por medio de una planeación estableciendo condiciones de prueba, junto con el área de sucursales realicé un modelo basado en matrices de escenarios típicos y atípicos, se generaron los documentos PT-MEDIDA y BI-PT.

   e) **Pruebas del sistema**. Realicé pruebas de integración llevadas acabo con el levantamiento de ambientes tanto de portal como ventanilla en sus respectivos puertos, el usuario realizó las pruebas aplicando el modelo de matrices que mencioné ateriormente, se verificaron y evaluaron los resultados del modelo de pruebas, se obtuvo la aprobación final del sistema ya que los usuarios no observaron bloqueos en los flujos al operar el nuevo sistema, los documentos generados fueron BI-PC y BI-DA (Documento de aceptación).

   f) **Procedimientos y entrenamientos de usuarios**. Diseñé y revisé procedimientos para crear manuales de entrenamiento y apoyo al personal para adaptarse rápidamente al nuevo sistema, baséndome en requerimientos de entrenamiento, capacitación mediante asignaciones a cada integrante, los documentos generados fueron BI-PC.

   g) **Conversión**. Transferí el estado operacional delegando responsabilidades al grupo de soporte, también me aseguré que el sistema estuviera listo para operarse, revisé la consistencia e integridad de la información, el área de soporte generó respaldos,

removió el sistema obsoleto y deployó el nuevo, otras acciones realizadas fueron el monitoreo a producción el cual incluyo el soporte en fase incial además del monitoreo de procesos manuales y automáticos, así como registrar y controlar variaciones en procesos para afinarlos, se generó el documento BI-PC.

h) **Revisión posterior**. Se llevaron a cabo revisiones continuamente, se identificaron mejoras potenciales y la evaluación final del proyecto.

- ✓ **Selección de paquetes**

La recopilación de las necesidades de hardware y software llevó a tomar desiciones importantes en la elección y selección de proveedores que cubrieran dichas necesidades utilizando criterios para la evaluación y así poder llegar a las negociaciones siguiendo el siguiente procedimiento:

1. **Objetivos**. Se definió la forma en que tendría que operar el sistema para su correcto funcionamiento, además de cómo debía implementarse sin dejar de mencionar que debía existir el compromiso de la dirección de usar paquetes en común y estar consientes de que podía haber cambios en el proyecto como resultado de las pruebas.

2. **Entregables**. Se integró del diseño funcional basado en los requerimientos del usuario, se desarrolló un reporte de especificaciones así como un prototipo funcional, se completó el diseño técnico a detalle para estimar la cantidad de trabajo, se creó un plan de instalación y un reporte gerencial describiendo costos y beneficios.

3. **Plan de trabajo**. Se conformó de ocho actividades:

   a) **Organización**. Se organizó el proyecto mediante la revisión de objetivos y personal involucrado así como la actualización del plan de trabajo, los documentos generados fueron PT-PAQUETE, BI-PT.

   b) **Requerimientos de usuario**. Se analizaron los requerimientos, se realizaron entrevistas, se documentaron las funciones que en ese momento existían, se identificaron y seleccionaron proveedores y se resumió la información.

   c) **Criterios de selección**. Se revisó el sistema que estaba en ese momento, tanto en documentación como inventarios, pantallas y formatos, se identificaron los requerimientos funcionales, se definieron los flujos de información, se identificaron entidades de datos a utilizar, se identificaron requerimientos no funcionales a través de la definición de las necesidades de integridad, seguridad, servicios, así como los requerimientos de Hardware y Software.

   d) **Evaluación y selección de paquetes**. Se evaluaron las características para determinar los servicios por proveedor, se obtuvo retroalimentación con los clientes, se analizó documentación del paquete, se negociaron los términos de contratos para obtener la aprobación de la dirección.

e) **Confirmación y diseño (esto para desarrollo adicional)**. Se definió el flujo de la aplicación, se determinaron requerimientos de las modificaciones, se definieron pantallas, reportes y formatos requeridos, se diseñaron tanto interfaces como las modificaciones mediante la identificación y prioridad.

f) **Planeación de la conversión**. Se definió el proceso de conversión mediante programas nuevos, prueba, limpieza de archivos y procesos de conversión manual, se estimaron requerimientos de Hardware, Software y personal, se generó el documento BI-PC.

g) **Análisis costo beneficio**. Se realizó la estimación de tiempo, costo de adquisición y desarrollo, los documentos generados fueron BI-EE y BI-ACB.

h) **Revisión y aprobación**. Se resumió tanto el diseño como las especificaciones funcionales y no funcionales mediante una presentación donde se recibieron observaciones para posteriormente ser aceptado.

- ✓ **Instalación de paquetes**

Concluidas las negociaciones con los proveedores, se procedió a preparar el lugar y equipo físico en donde instalé y configuré tanto hardware como software aplicando el siguiente procedimiento:

1. **Objetivos**. Se decidió instalar el paquete con la mínima interrupción de las actividades del negocio, se desarrollaron procedimientos para entrenar usuarios y asegurar la aceptación del sistema.

2. **Entregables**. Se complementó el diseño técnico además de dividir el sistema en módulos, se aprobó el sistema asegurándose de que cumpliera con los requerimientos del usuario.

3. **Plan de trabajo**. Se conformó de tres actividades:

   a) **Organización**. Se capacitó al equipo de trabajo, se detalló el plan de instalación, se crearon los documentos PT-PAQUETE y BI-PT.

   b) **Diseño detallado**. Se completó el diseño técnico incorporando nuevos puntos, el documento generado fue BI-ET.

   c) **Instalación de Hardware y Software**. Preparé el área física, definiendo la ubicación de cada PC, se negociaron requerimientos y fechas de entrega de los proveedores, se determinó plan de contingencia, se instaló, probó y monitoreó el wardware y software. Realicé la instalación y configuración del software requerido para realizar las actividades necesarias, también apoyé al equipo de trabajo para solucionar problemas de configuración. También verifiqué que el hardware estuviese funcional y debidamente conectado a red, que el equipo hardware estuviese completo con PC pentiun con Windows XP y paquete se servicios 2. El software que instalé y configuré lo detallo a continuación:

Instalación y configuración de Weblogic

Instalé el servidor de aplicaciones Weblogic 9.2 directamente en la unidad C:\Bea92, luego configuré los dominios de NEGO y BATCH de banco, las *templates* (plantillas) las agregué en la ruta de *templates*, estas plantillas fueron archivos *.jar proporcionados por el área de Arquitectura. En la tabla 2.1 detallo la configuración que realicé al pool de conexiones para conectarse a la base de datos.

Tabla 2.1: Configuración de *pools* de conexiones (Sistemas Administrativos GF, Eric López, 2007)

	Pool de Datos	**Pool de Seguridad**
Name	SI_JDBCConnectionPool	SI_Security_JBDCConnectionPool
Vendor	Oracle	Oracle
Driver	BEA's Oracle Driver (Type 4) versión 8.1.7, 9.0.1,9.2.0	BEA's Oracle *Driver* (Type 4) versión 8.1.7, 9.0.1,9.2.0
DBMS	SID	SID
DBMS *host*	146.2.20.1	146.2.20.1
DBMS *port*	1521	1521
User	DESARR	SEGUR
Password	xxxxxxxxxx	xxxxxxxxxx

Seleccioné la versión de java a utilizar para el desarrollo, compilación y deploy del proyecto, esta fue JAVA "***Sun SDK 1.5 @ C:\bea92\jdk150_22***".

Configuré los archivos con extensión *.Jar que estaban en la carpeta C:\bea92\weblogic92\server\lib***mbeantypes***, posteriormente sustituí la carpeta original mbeantypes con la que instalé Weblogic con el mismo nombre, esta nueva carpeta tenía otros jars actualizados con nuevas clases *.class. La reconfiguración de cada pool de conexiones la realicé desde la consola de Weblogic, al ingresar apareció el menú principal en la parte izquierda de la consola, seleccioné el botón *Lock* & *Edit*. En el dominio SI-NEGO seleccioné *Data Sources -> Connection Pools*, después seleccioné el *pool* que se modificaría, ingresé a la opción *Configuration -> Connection Pool,* configuré las IP, el usuario y *password*, después, en el menú de la izquierda seleccioné la opción *Save -> Activate Changes*.

Configuré de forma local la transaccionalidad haciendo que el servidor fuera el que realizara las ejecuciones de commit y rollback, las características ***Supports Global Transactions*** y ***Emulate Two Phase Commit*** las habilité.

Instalación y configuración de Apache JMeter

En las variables de ambiente agregué en el ***CLASSPATH*** que fue la ruta de java que utilicé en Weblogic, la cual fue C:\bea92\jdk142_05\bin, el sistema operativo para el desarrollo fue Windows XP .

La versión de java que utilicé para compilar y levantar el ambiente de WebLogic Server fue la misma que JMeter necesitaba para ejecutarse ya que JMeter buscó en la variable de ambiente *CLASSPATH* la ruta de java ejecutable. La herramienta quedó instalada en **C:\jakarta-jmeter-2.3.4** y se ejecutó con **jmeter.bat.**

Instalación y configuración de Eclipse

De forma local instalé Eclipse en la ruta ***C:\Eclipse*** y para configurar la memoria, desde el ícono del escritorio, dí click derecho y en propiedades coloqué como target: ***C:\eclipse\eclipse.exe -vmargs -Xmx512M*** para establecer 512 Mb que evitaron que se bloqueara ya que se tuvieron problemas cuando en Eclipse se estaba realizando la acción de *building* se desbordaba la memoria.

Otra forma con la cual especifiqué la memoria designada para Eclipse fue editar el archivo ***C:\eclipse\eclipse.conf*** y coloqué lo siguiente***:***

OpenFile

--auncher.XXMaxPermSize

--coloqué

512M

Instalación y configuración del *plugin CheckStyle*

Instalé este plugin de CheckStyle para que el proyecto cumpliera con los estándares de codificación especificados por *Sun Microsystems*, el procedimiento fue:

1. Copié el archivo proporcionado por el área de arquitectura ***com.atlassw.tools.eclipse.checkstyle_3.4.1.0.zip*** en la carpeta donde instalé Eclipse "C:\Eclipse*plugins*".
2. Dentro de Eclipse, en la opción *Window**Preferences*\ seleccioné la opción "*CheckStyle*"
3. Seleccioné el botón "***Import CheckSytle Config"***
4. Busqué el archivo "***SI-checkstyle.xml***" y lo seleccioné
5. Elegí botón "***OK***"
6. Seleccioné el Proyecto (dentro de eclipse – *Projects explorer*)
7. Seleccioné la opción "***Properties***"
8. Seleccioné la opción "***CheckSyle***"
9. Seleccioné el botón "***Add***"
10. En el campo con la etiqueta "***File Set Name***" escribí "***SI-checkstyle.xml"***
11. En el combo con la etiqueta "***Check Configuration***" seleccioné el archivo "***SI-checkstyle.xml"***
12. Oprimí botón ***Add* y *OK.***

Los errores que resolví aplicando checkstyle fueron la utilización errónea de primitivas con objetos numéricos, atributos ocultos, falta de *gettes* y *setters*, import no declarados, clases importadas que aún no se declarabas, falta de referencias a interfases, archivos *.jar no importados ni declarados en el archivo *.classpath*.

Instalación y configuración del *plugin Jaloppy*

Para instalar la configuración predeterminada del *Jalopy* en el proyecto seguí el siguiente procedimiento:

1. Copié el archivo proporcionado por el área de Arquitectura ***de.hunsicker.jalopy.plugin.eclipse_0.2.7.zip*** en la carpeta donde instalé Eclipse "C:\Eclipse *plugins*"
2. En la opción ***Window*** seleccioné la opción "***Jalopy Preferences***"
3. Seleccioné la opción "***General***"
4. Seleccioné la opción "***Import*.**"
5. Busqué y seleccioné el archivo "***SI-Jalopy.xml***"
6. Presioné botón "***OK***"
7. Presioné botón "***Apply***"
8. Seleccioné botón "***OK***"

Para darle formato a una clase Java desde Eclipse, abrí el archivo *.java y en el editor de código seleccioné la opción "***Format with Jalopy***" o utilicé simultaneamente "**Ctrl+Alt+F10**". Estos lineamientos se establecieron en el documento " **Lineamientos Programación*".*** Los errores más comunes que resolví aplicando jalopy fueron la falta de comentarios.

Instalación y configuración del *Plugin Subversión* SVN y repositorio CVS

Lo utilicé para integrar la funcionalidad subversión (SVN) y Sistema de Versión de Código (CVS) mediante los cuales pude subir, bajar, sincronizar con los repositorios de branch (proyecto) de trabajo, los pasos que seguí para la configuración fueron:

1. Copié la carpeta ***SVN_SubVersion*** que tenía el *plugin* a la ruta directa de C:\

2. Dentro de Eclipse, seleccioné *Menu -> Help -> Install new software -> New Local Site*, luego busqué y elegí la carpeta donde se encuentraba el jar correspondiente.

3. Reinicié Eclipse para que se tomara la configuración. Para agregar el repositorio, seleccioné -> *Window* -> *Open Perspective* -> *Other* -> SVN *Repository Exploring* -> click *OK,* agregué el nuevo repositorio SVN ingresando la IP y en nombre del repositorio:

   http://146.21.6.6/RepositorioRequerido

4. Para agregar el **repositorio CVS** no utilicé *plugin* externo.

Variables de Ambiente

Agregué y configuré las siguientes variables de ambiente:

- ✓ JAVA HOME = "C:\bea92\jdk150_22"
- ✓ ANT HOME = "C:\Server_Util\apache-ant-1.6.2"
- ✓ ECLIPSE_HOME= "C:\Eclipse_Helios_Java5"
- ✓ WEBLOGIC_HOME=" C:\bea92\weblogic92"

Actualización y configuración de *Jasper Reports* 1.0 a 3.0

El procedimiento que realicé para la configuración de las librerías que se necesitaban para la creación y compilación de reportes lo explico enseguida:

1. Coloqué en la carpeta ***lib/jasper*** el jar ***esiglo_reports.jar*** que contenía la sobrecarga del constructor *ReportGenerator* y en el constructor agregué un nuevo parámetro el cual indicaba si el reporte tendría que compilarse en tiempo de ejecución o no, esto fue *ReportGenerator(ImprimirComp, true).*
2. Eliminé los siguientes elementos del proyecto:

   a) ***poi-2.0-final-0126.jar***
   b) ***iText-1.3.1.jar***
   c) *jasperreports-1.0.1.jar*

3. Coloqué en la carpeta ***lib/jasper*** el archivo ***jar*** **poi-3.0.1-FINAL-2007*.jar*** que correspondió a la nueva versión.
4. Coloqué en la carpeta ***lib/jasper*** el jar ***jasperreports-3.0.1.jar*** que correspondió a la nueva versión.
5. Coloqué en la carpeta ***lib/jasper/iText*** el jar ***iText-2.1.3.jar*** que correspondió a la versión más reciente.
6. Agregué en las propiedades del proyecto las nuevas librerías ***jasperreports-3.0.1.jar***, ***poi-3.0.1-FINAL-2007.jar*** y ***iText-2.1.3.jar***
7. Agregué a la clase ***ConfigurationManager*** el método ***getInputStreamFromFile*** el cual recuperaba la ruta donde se encontraba el reporte compilado ****.jasper***, la ruta del directorio raíz ***reports.home*** y las concatené.
8. En la clase de negocio donde generé el reporte, modifiqué lo siguiente:
   a) En la llamada al constructor envié como segundo parámetro una bandera en "*true*" que indicaba que el reporte ya se encontraba compilado:

      ```
      new ReportGenerator(ImprimirComp, true)
      ```

   b) La llamada al método getInputStreamFromContext la sustituí por el método getInputStreamFromFile como lo muestro enseguida:

      ```
      ConfigurationManager.getInputStreamFromFile(ConfigurationContextConstant.REPORT_
      CTX,"multif.imprimirReporte.xml");
      ```

   c) Eliminé el llamado directo de *jasper* omitiendo la siguiente línea:

      ```
      System.setProperty("jasper.reports.compile.class.path","./silibs/jasperreports-1.0.1.jar");
      ```

9. En los archivos siguientes de configuración:

   \config\application-configuration.xml
   \config\wls\application-configuration.xml
   \config\batch\application-configuration.xml
   \config\batch\JobXml\application-configuration.xml

   Llevé acabo las siguientes acciones:

   a) Modifiqué en cada reporte la extensión del archivo *.jrxml" por *.jasper" tal como lo presento en las siguientes líneas:

      <property name="multif. imprimirReporte.xml">
      <value>multif/banco/ imprimirReporte.jasper</value>
      </property>

   b) Modifiqué la propiedad que hacía referencia al jar de *Jasper Reports*:

      <property name="jasper.jar">
      <value>./silibs/jasperreports-3.0.1.jar</value>
      </property>

   c) Agregué la propiedad de la variable ***reports.home*** apuntando al directorio raíz:

      <property name="reports.home">
      <value>/reports</value>
      </property>

10. Modifiqué el archivo *build*/jasperc.xml colocando la nueva versión a utilizarse de *Jasper* ***Reports***:

    <filelist dir="${lib.dir}/jasper" files="jasperreports-3.0.1.jar" />

Configuración y configuración de *ORACLE* 9i *Client*

El proceso de instalación y configuración que apliqué fue la siguiente:

a) Generé un servicio de la base de datos llamado *ORACLE*.

b) Seleccioné el NOMBRE/IP y puerto 1521 del *host* de la base de datos.

c) Para la configuración del servicio de red, coloqué un nombre cualquiera, ya que en ese momento no era importante.

d) No guardé el nombre del servicio y terminé la instalación.

Instalación y configuración de *Toad* para *Oracle*

En la instalación de *Toad* donde se configura el *Job Scheduler* , las opciones de *Install* UNIX/*Job Scheduler Scripts*,*Back up SQL Templates*, *Back up Script Manager Scripts* y *Back* up UNIX *Job Scheduler Script* las dejé desactivadas.

Para conectar a la base de datos, solicité un usuario y clave así como el rol de *CONNECT* para poder hacer consultas. Para el desarrollo conté con un nombre de usuario y

password exclusivos, a los usuarios les asignaron diferentes, los jefes y el gerente de sistemas tenían permisos especiales de lectura y consulta a las bases de datos de usuario, también había usuarios QA.

La configuración de las instancias la realicé en un archivo ubicado en la carpeta: C:\oracle\ora92\network\ADMIN**TNSNAMES.ORA**

Dentro de la herramienta TOAD pude interactuar con la base de datos, el archivo **TNSNAMES.ORA** contenía la sintaxis general de configuración que me permitió conectar a diferentes bases de datos.

d) **Configuración de paquete y programación**. Le dí soporte a la programación y configuración mediante la codificación de parámetros del paquete, coordinando la programación de las modificaciones, realicé inventario de mensajes que apliqué en pruebas unitarias de forma cíclica, verifiqué los resultados y los revisó el usuario, los documentos generados fueron PT-PAQUETE, BI-PT y BI-ET.

e) **Preparación de la conversión**. Completé el plan de conversión desarrollando procedimientos, se determinaron requerimientos por cada archivo, realicé la planeación de las pruebas del sistema, determiné las condiciones y llevé a cabo el modelo de prueba, se generaron los documentos PT-PAQUETE, BI-PT y BI-PC.

f) **Pruebas del sistema**. Realizé pruebas de integración, verifiqué resultados y realicé cambios necesarios así como las pruebas de usuario monitoreando el desempeño obteniendo así la aprobación del sistema. Estas pruebas las realicé en conjunto con el usuario, es precisamente en este punto donde el usuario observó el funcionamiento previo y podía realizar sus observaciones si es que tenía la intensión de solicitarlas, estos cambios se fueron aplicando conforme el usuario lo pedía, los documentos generados fueron BI-PC.

g) **Procedimientos y entrenamiento de usuarios**. Revise los procedimientos que ya existían, programé sesiones y evalué el entrenamiento a los usuarios, también modifiqué los manuales de operación y los entregué al usuario.

h) **Conversión**. Transferí responsabilidades al grupo de soporte, me aseguré que los recursos estuvieran listos revisando la integridad de la información manteniendo rrespaldo, se monitorearon procesos manuales y automáticos, se generó el documento BI-PC.

i) **Revisión posterior**. Se realizaron modificaciones con alta prioridad, se revisaron e implementaron peticiones, hubo revisiones continuas, se preparó el reporte final, se obtuvo la aprobación final de la dirección de sistemas, se realizó la evaluación final del proyecto, también identifiqué mejoras al sistema.

- ✓ **Desarrollo interactivo de sistemas**

Una de las fases en las que tuve mayor participación fue esta, en donde llevé a cabo el desarrollo del sistema aplicando tanto los conocimientos adquiridos como nuevos, realicé modelado, diagramas, documentación de requerimientos y especificaciones, la integración del equipo de trabajo, la aplicación de especificaciones para el desarrollo del sistema, para esto se aplicó el siguiente procedimiento:

1. **Objetivos**. Se establecieron oportunidades y problemas de negocio permitiendo implementar soluciones al sistema.

2. **Aplicabilidad**. El enfoque para aplicación que desarrollé tuvo alta participación del usuario, restricción de recursos, ambientes volátiles y estimaciones.

3. ***Rational Unified Process***. Utilicé el proceso de reingeniería llamado Proceso Unificado Racional (RUP), permitiéndome asignar tareas y responsabilidades en la organización del equipo de trabajo.

4. **Entregables**. Consistió en el sistema operacional totalmente probado, la documentación del usuario y documentación para soportar la operación contínua y mantenimiento del sistema.

5. **Diagramas UML**. Realicé diagramas de casos de uso que describieron los roles del usuario, diagramas de clases que modelaron la relación entre ellas, diagramas de objetos que mostraron la configuración del sistema, diagramas de iteracción que mostraron el comportamiento mediante escenarios, diagramas de componentes que representaron paquetes físicos y diagramas de paquetes que agruparon los tipos de objetos.

6. **Plan de trabajo**. Se conformó de cuatro actividades:

   a) Iniciación. En esta fase se realizó el modelado del negocio para diagnosticar el estatus de la organización, se definieron objetivos del modelado del negocio, describí el negocio y sus procesos, definí los procesos que se debían automatizar, se realizaron requerimientos para analizar los problemas y a su vez comprender las necesidades de los usuarios, se definió el sistema y su alcance, analicé los requerimientos aprobados en el documento de visión, se administró el proyecto llevando a cabo la definición de uno nuevo evaluando alcances y riesgos, se definieron requerimientos para el ambiente de desarrollo, se definieron las plantillas (diseños) para los diferentes artefactos, los documentos generados fueron RUP-MODELO DE NEGOCIO (Documento descriptivo del Negocio), RUP-VISION (Documento de Visión y stakeholder´s o patrocinador del proyecto), PT-ITERATIVO, BI-PT y RUP-ARQUITECTURA (Documento de Arquitectura tecnológica).

   b) Elaboración. Determiné la prioridad de los casos de uso, detallé y estructuré el modelo de casos de uso, se desarrollaron prototipos de las interfaces de usuario, se realizó el análisis de la arquitectura candidata, identifiqué subsistemas e interfaces, identifiqué

clases de diseño, elaboré matrices de pruebas donde la implementación consistió en el desarrollo del prototipo funcional así como el diseño de las pruebas planificadas y diseñadas de casos típicos y atípicos, configuré repositorios a nivel desarrollo, realicé el control de versiones y configuraciones, evalué el alcance y los riesgos llevando a cabo el monitoreo del plan de trabajo, se afinaron las plantillas, documenté guías de uso y manuales técnicos, preparé el ambiente de desarrollo para las pruebas unitarias y funcionales, los documentos generados fueron D-CASO USO (Diagrama de caso de uso), RUP-ARQUITECTURA, D-CLASES (Diagrama de Clases persistentes), BI-DOC ARTE (Documento de diseño de Interfaz gráfica), D-SECUENCIA (Diagrama de secuencia), D-ESTADOS (Diagrama de transición de estados), D-ER (Diagrama de entidad-relación), RUP-ARQUITECTURA, BI-ODS (Documento de Orden de Servicio), RUP-M PRUEBAS-U (Matriz Pruebas Unitarias), RUP-M PRUEBAS F (Matriz Pruebas Funcionales), PT-ITERATIVO, BI-PT.

c) Construcción. Se analizaron y evaluaron los cambios en requerimientos, se diseñaron los componentes y *Frameworks* (Marcos de trabajo basados en un modelos previamente creados), participé en el diseño de la base de datos, codifiqué funciones y algoritmos, diseñé, definí y realicé las pruebas y su monitoreo, evalué alcances y riesgos, apoyé al equipo de desarrollo para el uso de herramientas tanto en instalación como en configuración, se crearon los documentos D-CASO USO, RUP-ARQUITECTURA, BI-PC, RUP-M PRUEBAS-U, RUP-M PRUEBAS F,BI-CVS (Esquema de trabajo con CVS), PT-ITERATIVO y BI-PT.

d) Transición. Consistió en la implantación (*deployment*) donde transferí responsabilidades al área de soporte quienes generaron y mantuvieron respaldos de la aplicación, se registraron y controlaron incidencias, se afinaron procesos, en la configuración se administraron cambios para poder implementar peticiones de alta prioridad y se afinó la configuración, se evaluaron los riesgos, se identificaron mejoras para la siguiente iteración, se preparó el ambiente de producción tanto Hardware como Software, se generaron los documentos BI-PC, RUP-ARQUITECTURA, BI-CVS, PT-ITERATIVO, BI-PT, RUP-ARQUITECTURA.

7. **Diagrama de actividades**.

En la figura 2.3 presento las actividades relacionadas de forma secuencial e incremental de cada fase las cuales fueron iterativas.

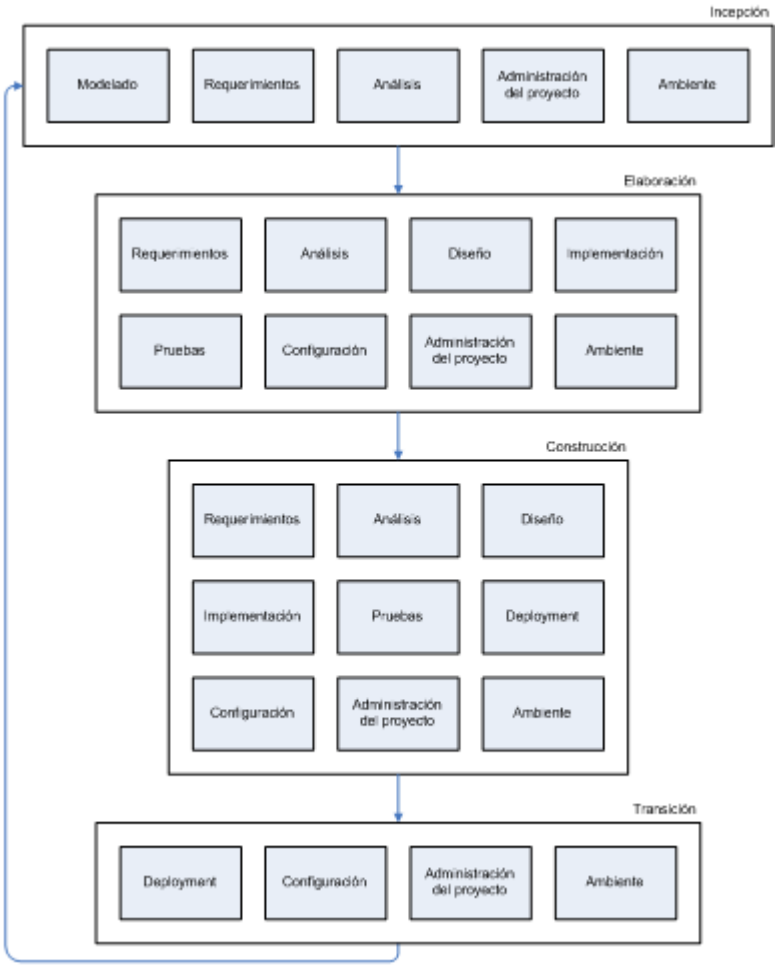


Figura 2.3: Diagrama de actividades (Documento interno del GF, Jorge Delgado, 2005).

- ✓ **Mantenimiento y soporte al sistema de producción**

1. **Objetivos**. Se aseguró que los cambios al sistema en producción fueran implementados efectivamente con las mínimas interrupciones para los usuarios, asegurando soporte a la organización, monitoreo y evaluación de estatus para que el sistema pudiera adaptarse a los cambios requeridos por el usuarios.

2. **Insumos**. Se integró por solicitudes de cambio y prioridades establecidas por el usuario.

3. **Entregables**. Los productos de esta fase fueron manuales de procedimiento, especificaciones de programación, cambios a programas de entrenamiento a usuarios, reportes de estatus de solicitudes de cambio atendidas, evaluaciones de desempeño, tiempos de ejecución y evaluación total del sistema en producción.

4. **Plan de trabajo**. Se conformó de siete fases:

   a) **Requerimientos del usuario**. Se revisaron y valoraron requerimientos de cambio con el usuario, se creó el documento Requerimiento de Producto de Negocio (BI-RPN).

b) **Monitoreo de producción**. Se realizó un programa de revisión periódica a los procedimientos, se realizaron entrevistas para los requerimientos de cambio, se determinaron indicadores de desempeño tales como tiempos de respuesta en alta carga de peticiones de usuario, con base en los resultados se propusieron recomendaciones para cada variación y se generó el documento BI-RPN.

c) **Evaluación interna**. Se evaluó la necesidad de la actualización con base en la satisfacción de usuario, desempeño y eficiencias del sistema, se analizaron actividades previas de soporte y mantenimiento, se evaluó la necesidad de recodificar y documentar el registro de control de cambios, los documentos generados fueron BI-QA (Monitoreo de desempeño del sistema) y BI-RPN.

d) **Análisis de cambios**. Se definió el plan de trabajo determinando el nivel de esfuerzo requerido así como la definición de tareas y pasos a realizar, además se analizó el costo/beneficio y aprobación del usuario, los documentos generados fueron PT-MANTENIMIENTO, BI-PT, BI-ACB.

e) **Diseño de cambios**. Me reuní con el usuario para ver requerimientos de modificación que incluyeron rediseño de reportes, pantallas y formatos, se redefinió el flujo de información, revisé con el usuario los resultados y el diseño técnico mediante inventarios, descripción de archivos y bases de datos, además generé el diccionario de datos, se generó el documento BI-ET.

f) **Modificación al sistema**. Programé y probé cambios de desarrollo, revisé el código, se actualizó la documentación mediante inventario de programas, se actualizó el modelo de prueba, modifiqué el código siguiendo un protocolo de documentación establecida, en muchos casos las modificaciones fueron con impacto desde la capa de presentación hasta la capa de datos, realicé los cambios de forma inmediata ya que estos fueron sobre el ambiente que estaba en producción dejando como segundo plano otras asignaciones menos urgentes con menos prioridad, se generó el documento BI-PC.

g) **Implementación**. Revisé el sistema con el usuario para la aceptación del sistema, capacité al personal, se instalaron modificaciones que incluyó el monitoreo del proceso, actualicé jobs, scripts y hojas de requerimientos, el documento generado fue BI-DA.

Para la metodología apliqué un paradigma de desarrollo, el cual menciono en el siguiente apartado.

2.3 PARADIGMA DE PROGRAMACIÓN

Apliqué la Programación Orientada a Objetos (POO) en la plataforma Java Edición Empresarial (JEE) , este paradigma orientado a objetos me permitió contar con soporte sintáctico para los tipos abstractos de datos más prestaciones asociadas a las jerarquías de clases, además de reutilización de código, mantenimiento accesible y escalable, también el lenguaje POO me permito implementar herencia y polimorfismo, este paradigma tuvo potencial ya que los objetos (instancia de clases) se podían asimilar y

reflejar como problemas reales. Un aspecto importante que se tomó en cuenta fue la arquitectura, columna vertebral del proyecto, la cual menciono en el siguiente apartado

2.4 ARQUITECTURA

Desde la concepción inicial el sistema fue pensado para ser integral en funcionamiento, operatividad y tecnología con una arquitectura n-capas basadas en el paradigma de diseño MVC apoyado de patrones de tipo estructural, de comportamiento y creación, implementé el framework de struts para controlar las peticiones del usuario tal como lo presento en la figura 2.4. Esta arquitectura se diseño para ser un sistema web con acceso desde el protocolo de internet HTTP con peticiones desde un navegador o explorador en cada una de las terminales, las peticiones también se realizaron desde otros sistemas creados en otras plataformas diferentes a java. Dentro del servidor de aplicaciones existieron alojados en diferentes ambientes como el de desarrollo, usuario, control de calidad (QA), auditoría y producción, por otro lado en lo que se refiere al acceso servicios de la aplicación llamados *web services* fueron consumidos por el mismo sistema y por otros creados en plataforma .NET mediante puertos y protocolos HTTP y SOAP, el paradigma MVC me permitió aplicar una estructura de capas las cuales fueron:

a) Capa de presentación

Se integró por archivos dinámicos con extensión *.jsp, el software del navegador, peticiones HTTP por parte del usuario, otros archivos incluidos fueron hojas de estilo, archivos JavaScript, formularios con la información a presentar al usuario, en esta capa inicié la sesión del usuario firmado al sistema.

b) Capa de aplicación

Utilicé el patrón *Command*, librerías API de java para la transaccionalidad, servicios de mensajería listeners mediante Cron, los objetos que contenían la información y que viajaba en todas las capas les llame VOs, además incluí objetos cuya función era proporcionar servicios de conectividad con otros sistemas, otros objetos fueron de tipo **Task*.java para realizar tareas por lotes. En esta capa inicié la sesión para la transaccionalidad.

c) Capa de modelo de negocio

En esta capa integré los objetos e interfases de negocio, además de las clases con los métodos necesarios para que el sistema cumpliera su cometido.

d) Capa de datos

Implementé el *Framework Hibernate*, el patrón DAO, mediante objetos e interfases con métodos específicos de acceso y comunicación a la bases de datos. Apoyándome de la conectividad JDBC, también en los mapeos utilicé objetos VOs de transferencias, pojos u objetos planos llamados persistentes, archivos de configuración XML, se incluyó la librería Log4J para adminsitrar los mensajes de de bitácora de la aplicación para registrar los eventos que se llegaron a presentar, estos se guardaron en archivos con extensión *.log. En esta capa cerre la sesión después de aplicar la transacción y persistencia.

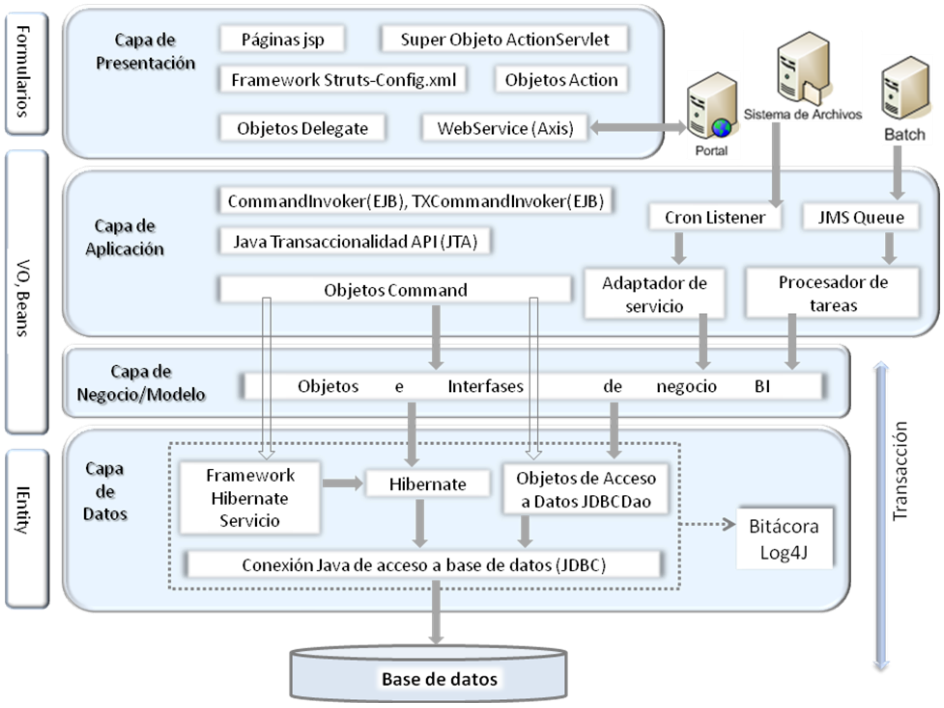


Figura 2.4: Arquitectura de n-capas MVC (Documento interno GF, Procesos, 2005)

En el siguiente apartado mencionaré la aplicación de los estándares internacionales de nombrado que utilicé para el desarrollo del código.

2.5 ESTANDARES DE CODIFICACIÓN

Convenciones de Codificación por Sun Microsystems

En 1997, Sun Microsystems publicó un documento de convenciones de codificación en Java (http://java.sun.com/docs/codeconv/) y se revisó en 1999, estas convenciones fueron utilizadas por el GF con algunas puntualizaciones no cubiertos por dicho documento así como convenciones específicas para los proyectos de la nueva arquitectura. Se consideró obligatorio que todo desarrollador leyera cuidadosamente las convenciones de codificación de Sun Microsystems.

Convención de nombrado

Fueron importantes para definir *Pointcuts* (Unificaciones para asociar expresiones) con base en nombres de clases y métodos. Por ejemplo, se podía definir que todos los métodos que comenzaban con *save* o *update* requerían ejecutarse dentro de una transacción, estas unificaciones fueron:

a) Paquetes.

Los nombres entre los símbolos “< >” fueron obligatorios. Los nombres entre los caracteres “[ ]” podían o no existir dependiendo de las características de la aplicación. La raíz de los paquetes de las aplicaciones del SI comenzaron con:

mx.com.grupofinanciero.si.

b) Servicios de Aplicación

Por consistencia, los verbos que utilicé en los nombres para interfaz y clase fueron en infinitivo:

Intefaz <nombre descriptivo>

Clase <nombre de la interface>Impl.

mx.com.grupofinanciero.subfolder.core.<línea de negocio>.aplicacion.<módulo>.[submódulo[.submódulo]].

c) Servicios de Negocio

mx.com.grupofinanciero.subfolder.core.<línea de negocio>.negocio.<módulo>.[submódulo[.submódulo]].

d) Entidades u objetos de dominio

Las entidades u objetos de dominio fueron el modelo del negocio, el nombrado debía ser de la siguiente manera:

mx.com.grupofinanciero.subfolder.core.<línea de negocio>.datos.[módulo].[submódulo[.submódulo]].persistencia.

e) Data Access Objects (DAO)

El nombrado de paquetes para este tipo de DAO debía ser de la siguiente manera:

mx.com.grupofinanciero.subfolder.core.<línea de negocio>.datos.<módulo>.[submódulo[.submódulo]].dao.

- ✓ **EntityDAO**

  Intefaz: <nombre de la entidad>DAO

  Clase: <nombre de la entidad>HibernateDAO

- ✓ **RepositoryDAO**

  Intefaz: <nombre descriptivo>DAO
  Clase: <nombre de la interface>HibernateDAO

f) Procedure Access Objects (PAO)

El nombrado de paquetes para este tipo de PAO debía tener la siguiente estructura:

mx.com.grupofinanciero.subfolder.core.<línea de negocio>.datos.<módulo>.[submódulo[.submódulo]].pao

Por consistencia, los verbos que utilicé en los nombres debían estar en infinitivo.

Intefaz <nombre descriptivo> PAO
Clase <nombre de la interface>PAOImpl

Como parte del desarrollo incluyo a detalle los patrones de diseño que utilicé los cuales menciono enseguida:

2.6 PATRONES DE DISEÑO

Los patrones de diseño conocidos como "*Dessign Patterns"* son soluciones simples, funcionales y elegantes a problemas específicos comunes del diseño orientado a objetos, además están basados en la experiencia. Con el paso de los años se han presentado problemas en los diseños de las aplicaciones los

cuales responden a ciertos patrones o eventos que se repiten constantemente con características semejantes, en esta sección presentaré los patrones que me ayudaron a que los errores vividos en otros proyectos anteriores no se repitieran en el proyecto SI, permintiéndome que el diseño fuera reutilizable, y modular.

Patrón *Transfer Object* (VO)

Este patrón de la capa lógica de negocio lo implementé para encapsular los datos permitiéndome enviar y recibir el objeto de transferencia que el cliente solicitaba al *bean* de negocio el cual creaba el objeto con sus respectivos atributos y pasarlos por valor al cliente tal como lo presento en los diagramas de la figura 2.5. Este patrón me facilitó el intercambio de datos entre capas reduciendo el diálogo y tráfico de red ya que cuando un *bean* de negocio usaba el objeto de transferencia el cliente invocaba remotamente al bean de negocio para solicitarlo, este *bean* de negocio era instanciado por muchas llamadas de métodos remotos para obtener los valores con atributos específicos del objeto de transferencia.

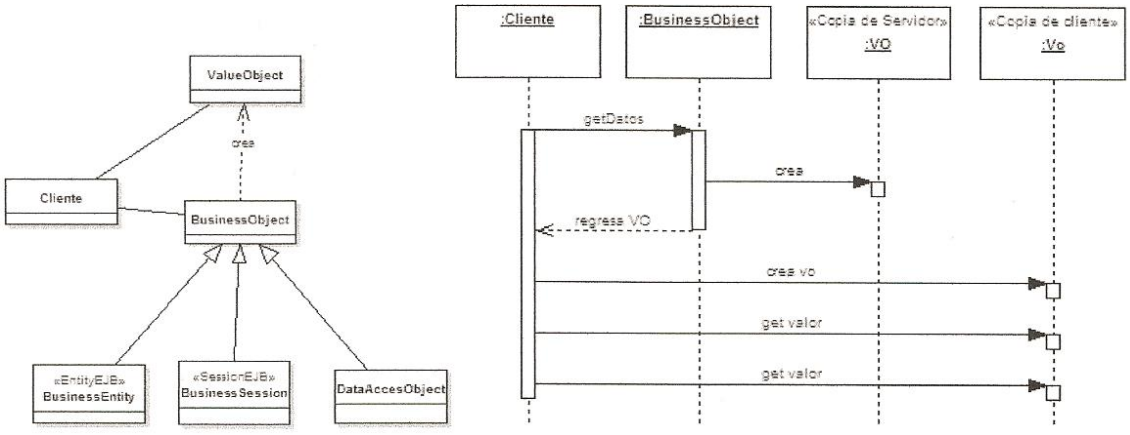


Figura 2.5: Diagrama de clases y secuencia de patrón Transfer Object (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *SessionFacade*

Este patrón estructural de la capa lógica de negocio lo implementé con un sesión *bean* para poder simplificar el acceso a componentes java *beans*, administrar los objetos de negocio, las relaciones entre objetos de negocio y evitar el mal manejo del objeto tal como lo presento en los diagramas de clase y secuencia de la figura 2.6, además este patrón me facilitó la recepción de objetos para ser modificados y delegar el trabajo a la capa de persistencia, este patrón lo relacioné directamente con el *Transfer Object* (TO) para proveer servicios de negocio.

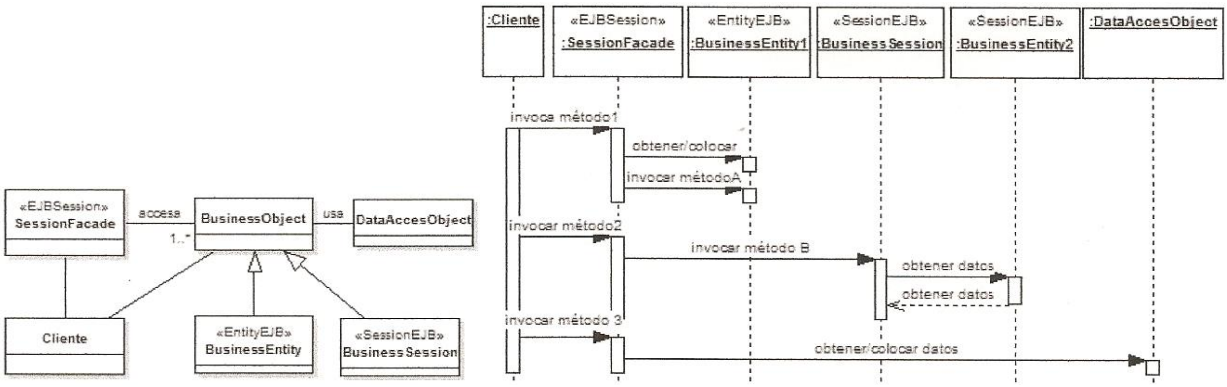


Figura 2.6: Diagrama de clases y secuencia de patrón sessionFacade (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *Transfer Object Assembler* (TOA)

Este patrón de la capa lógica de negocio lo implementé para generar objetos de transferencia (VOs) a partir de un conjunto de datos obtenidos de objetos de negocio, estos objetos de transferencia fueron requeridos para enviar información al servidor, en los diagramas de la figura 2.7 presento cómo generé el objeto por cada petición realizada al servidor, este patrón me ayudó a ensamblar VOs compuestos de multiples fuentes de datos y en la figura 2.8 presento la composición de la clase VO.

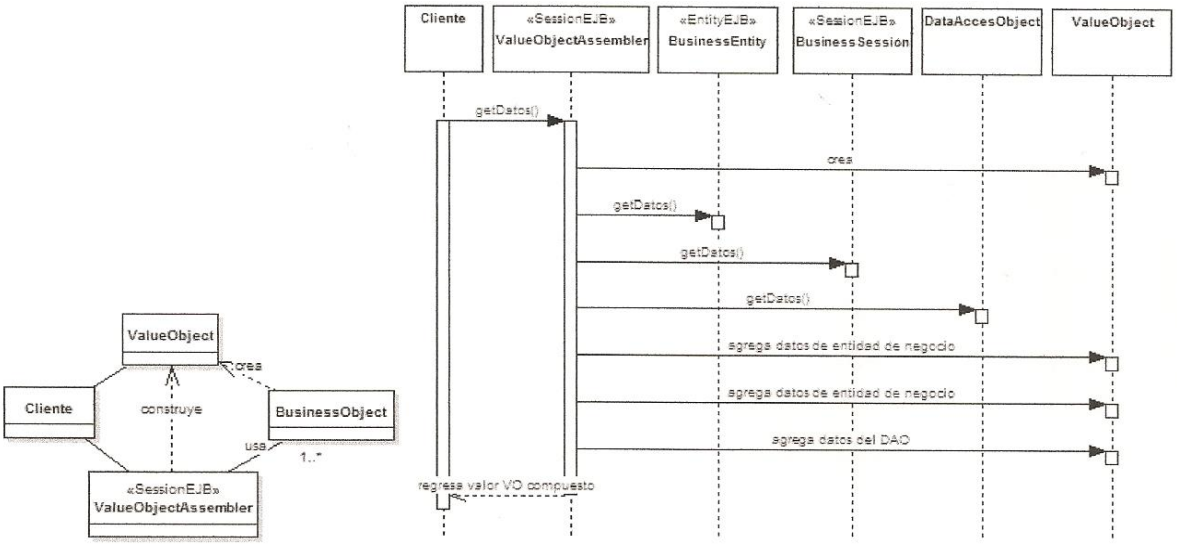


Figura 2.7: Diagrama de clases y secuencia de patrón TOA (Control de versiones GF, Zitziqui Gutiérrez, 2007)

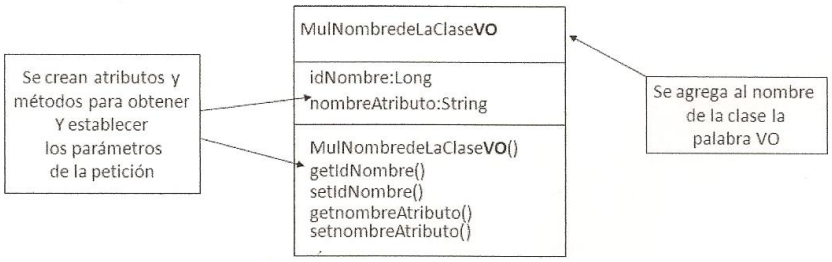


Figura 2.8: Composición de clase VO (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *BusinessDelegate*

Este patrón de la capa lógica de negocio lo implementé para hacer independientes la capa de presentación y los servicios de negocio, además de que me facilitó el acoplamiento entre los clientes ocultando los detalles de invocación de dichos servicios, también me permitió proporcionar a la capa de presentación una abstracción del negocio que ocultó los detalles de invocación a métodos remotos, acceso a servicios de mensajes y componentes EJB utilizados, en los diagramas de clases y secuencia, en la figura 2.9 muestro la parte de vista representada por la clase cliente donde generé una clase padre delegate la cual pasó el control a la capa de negocio que llevaba la información del mismo, esta configuración la integré en el archivo del proyecto llamado CommandMapping.xml donde el servicio de localización buscó el correspondiente objeto de negocio.

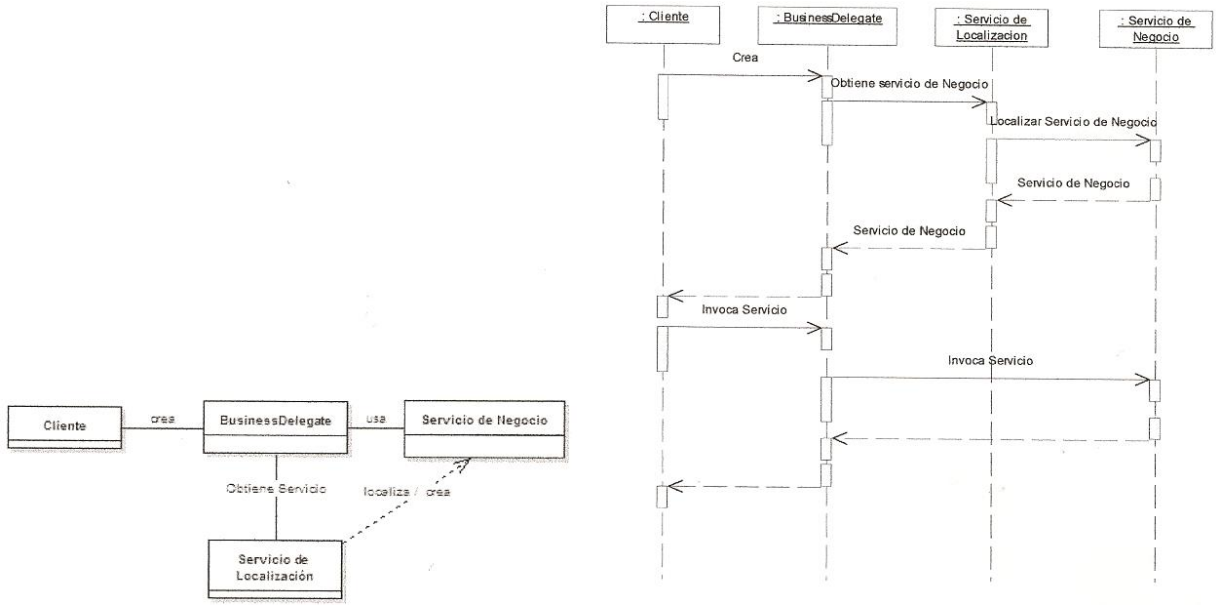


Figura 2.9: Diagrama de clases y secuencia de patrón BusinessDelegate (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *Service Locator*

Este patrón de la capa lógica lo implementé para reducir la complejidad en tareas de mantenimiento al convertir excepciones de sistema y red a excepciones de negocio además de ocultar las conexciones remotas y poder diseñar servicios que utilizaron caché mejorando el *performance* (rendimiento), me permitió localizar recursos en el servidor de Aplicaciones evitando que la cantidad de búsquedas hechas en el contexto JNDI pudiera impactar negativamente en el rendimiento de la aplicación, este patrón estaba relacionado directamente con el patrón de diseño BusinessDelegate, en la figura 2.10 muestro el diagrama de clases y secuencia.

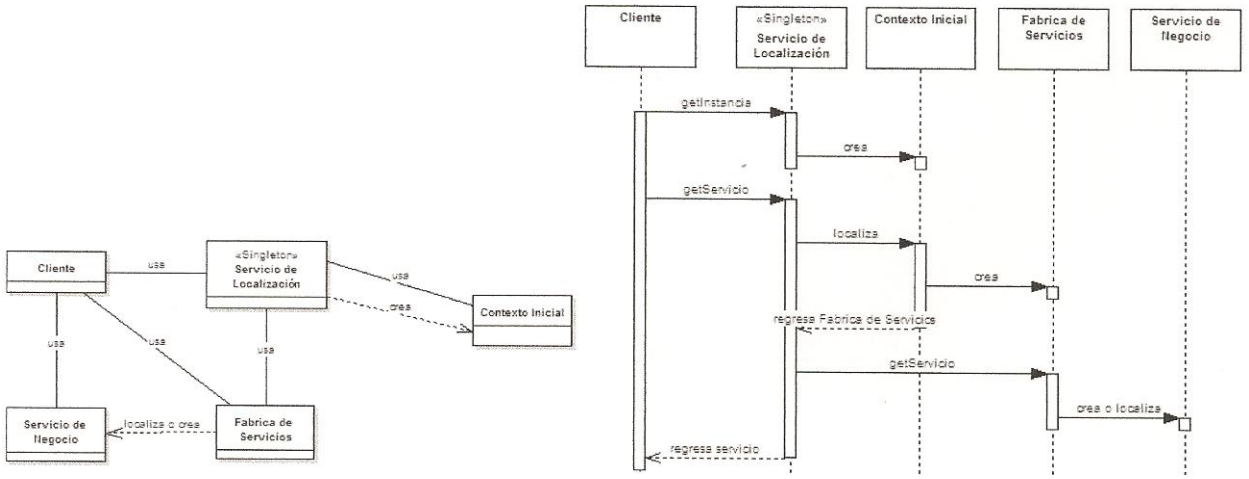


Figura 2.10: Diagrama de clases y secuencia de patrón Service Locator (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *Command*

Este patrón de comportamiento me permitió encapsular una petición en un un objeto y así parametrizar las distintas peticiones de los clientes, encolar y llevar un registro de las peticiones para poder deshacer la transaccionalidad, el acceso a la capa de negocio la realicé mediante objetos de órdenes conocidos como objetos *Command* (Patros de diseño *Command* GoF) que tenían tres roles:

a) **Invocador**. Componente *SyncronousCommandInvoker* o *TxSyncronousCommand Invoker*.

b) **Comando Concreto.** Los Objetos *Command* los generé en cada caso de uso.

c) **Receptor**. Las clases *Busines* las utilicé para ejecutar la lógica de negocio.

Debido a la necesidad de limitar al número de accesos de la capa de presentación al negocio, el Invoker lo construí utilizando la tecnología de *Enterprise* Java *Beans*, los invocadores de comandos fueron:

a) **SyncronousCommandInvoker.** Lo utilicé cuando el comando a invocar NO NECESITABA ejecutarse en un entorno transaccional como fueron las consultas a la base de datos.

b) **TxSyncronousCommandInvoker**. Lo utilicé cuando el comando que invocaba SI REQUERÍA ejecutarse en un entorno transaccional como una actualización o una inserción a la base de datos.

En la figura 2.11 presento el flujo completo del negocio desde la capa de vista hasta la capa de negocio.

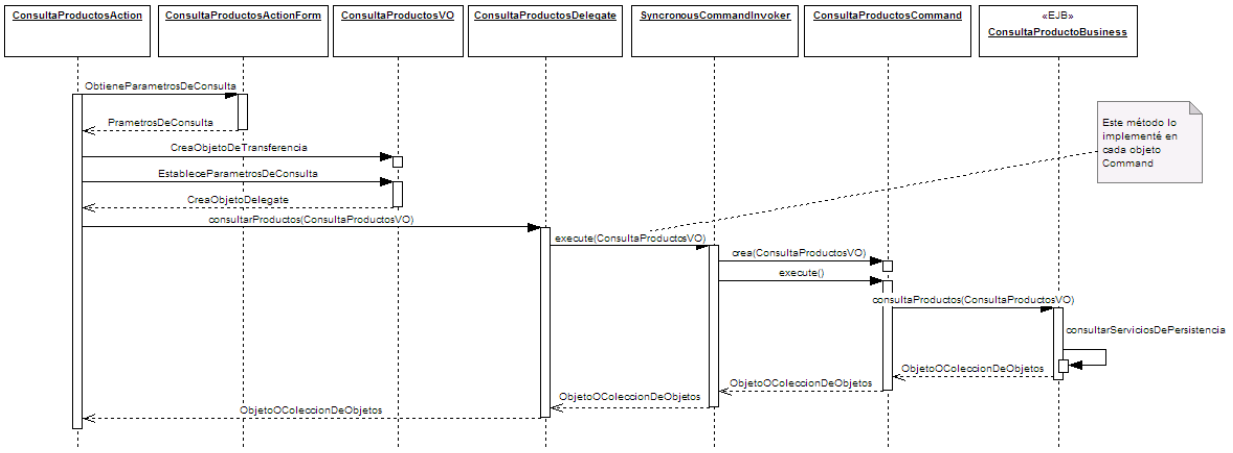


Figura 2.11: Diagrama de secuencia de patrón *Command* (Control de versiones GF, Zitziqui Gutiérrez, 2007)

La función del archivo **CommandMapping.xml** fue ligar un objeto *Command* con el respectivo objeto VO que lo invocaba. La configuración de Comandos se ejecutaba cuando iniciaba la aplicación y los valores se almacenaron en la memoria evitando demasiados accesos al archivo XML tal como lo presento en la figura 2.12.

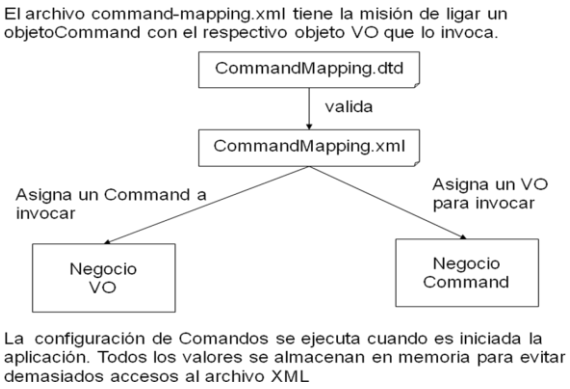


Figura 2.12: Funciones de CommandMapping.xml (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Patrón *Proxy* (GoF)

Este patrón de tipo estructural lo implementé para generar un representante de otro objeto por razones de acceso, seguridad y velocidad, tenía todas las características y funciones del patrón BusinessDelegate.

Patrón DAO

Este patrón de la capa de persistencia me pertimitió utilizar el objeto de acceso a datos (DAO) para abstraer y encapsular todos los accesos a la fuente de datos, obtenerlos de forma transparente y poder almacenarlos con un mecanismo de almacenamiento persistente. Con este patrón empleé la interface de programación (API) nativa del manejador de la base de datos para separar las operaciones de bajo nivel de acceso a datos de las operaciones de alto nivel de la lógica de negocio, en la figura 2.13 presento la relación y secuencia de este patrón.

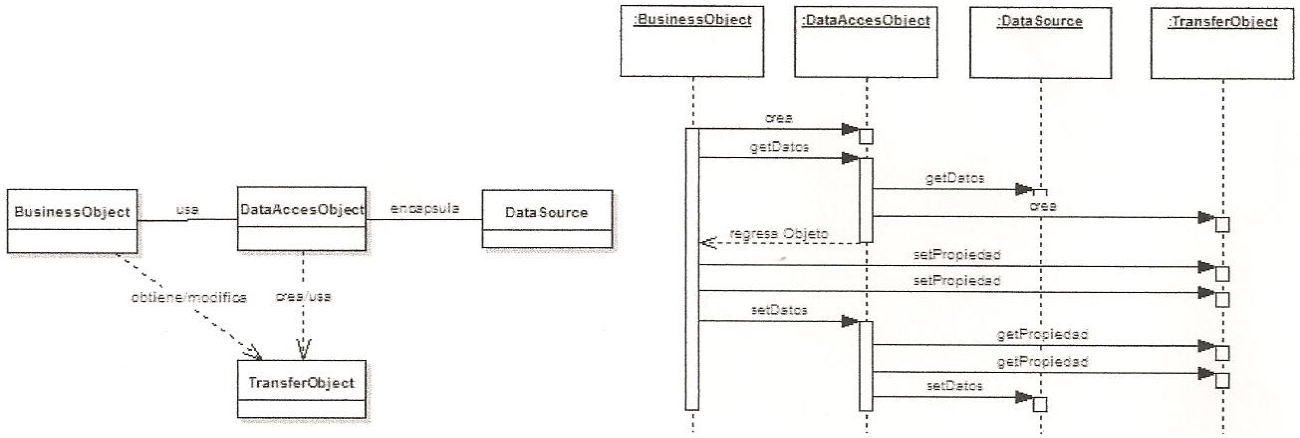


Figura 2.13: Diagrama de clases y secuencia patrón DAO (Control de versiones GF, Zitziqui Gutiérrez, 2007)

Además de los patrones de diseño que utilicé, existieron marcos de trabajo implementados para el desarrollo del proyecto, estos los puntualizo en la siguiente sección.

2.7 MARCOS DE TRABAJO

Struts

Fue el marco de trabajo propuesto por la Dirección de Sistemas, Dirección de Organización y Métodos, como parte de la fase de iniciación que permitió reutilizar arquitecturas que garantizaron un correcto control de la aplicación, desde su vista se podía conocer perfectamente el flujo de negocio, qué información es la que se estaba operando, permitió implementar seguridad generando resultados satisfactorios con capacidades de extensión y expansión. Con Struts utilicé paradigma de diseño MVC (Modelo Vista Controlador) el cuál se integró de las siguientes partes:

a) Vista

Conformada por archivos nutridos de información dinámica y estática con formatos específicos, controles y etiquetas, estos archivos llamados JSPs presentaban la información obtenida de la base de datos, esta información era direccionada por el controlador por medio de formas o formularios, en términos operativos, la vista es lo que el usuario veía como primera instancia, es donde iniciaban las peticiones del usuario.

b) Controlador

Una vez que el cliente realizaba las peticiones desde el navegador o explorador se encargó de redirigir o asignar una aplicación a cada petición y contaba con una especie de “mapa” de correspondencias entre peticiones y respuestas que se les asignaron, definía la lógica de negocio que se ejecutaba y luego delegando las responsabilidades para conducir a la siguiente fase.

c) Modelo

En el llevé a cabo la lógica del negocio aplicada al sistema que se desarrolló aunque Struts no me proporcionó tantos elementos para la implementación del modelo ya que esa fue mi responsabilidad dividiéndose en dos sistemas:

- ✓ El estado interno del sistema.
- ✓ Las acciones que se tomaron para cambiar el estado.

La configuración de Struts fue llevada al nivel de WAR y dentro de éstos se integraron los componentes que presentó en la tabla 2.2

Tabla 2.2: Componentes de Struts (Control de versiones GF, Zitziqui Gutiérrez, 2007)

/WEB-INF/lib/struts.jar	Este *.jar se integró a las clases utilizadas por Struts
/WEB-INF/lib/commons-*.jar	Serie de archivos JAR que estuvieron presentes para la correcta ejecución de Struts, estos archivos se encontraban en la misma distribución de Struts
/WEB-INF/lib/web.xml	Este archivo lo utilicé para configurar varios aspectos de configuración y parametrización de los Servlets de un WAR, incluyó la configuración de la aplicación, conexión a base de datos y seguridad

Hibernate

Este patrón me permitió el mapeo de entidades (tablas) objeto-relacionales (clases java), ofreció ventajas para obtener los objetos con sus correspondientes relaciones, inserciones, actualizaciones y eliminaciones. Me encontré con inconvenientes con mapeos incorrectos para algunas tablas y como el sistema fue de gran volumen, me reducía el desempeño ya que los objetos creados se cargaron en la JVM por lo que el tiempo de consulta era muy alto y bloqueaba los hilos, para poder resolver el problema de memoria y desempeño me apoyé del patrón DAO explicado en el apartado 2.6.

Generé archivos *.hbm.xml que contenian la relación y mapeo objeto relacional entre los objetos de la base de datos, existió el archivo hibérnate.cfg.xml para configurar hibérnate. Las librerías integradas al sistema para utilizar hibernate fueron: antlr-2.7.6.jar, commons-collections-3.1.jar, dom4j-1.6.1.jar, hibernate3.jar, jta-1.1.jar, javassist-3.4.GA.jar, slf4j-api-1.5.6.jar y slf4j-simple-1.5.6.jar.

Los tipos de mapeos los realicé conforme al diseño de los objetos de la base de datos para cumplir con la normalización, integridad referencial y consistencia de los datos fueron: muchos a muchos, uno a uno, uno a muchos, muchos a muchos utilizando una tabla relación. El desarrollo del código de los tres módulos del sistema integral en los cuales participé lo detallo en el siguiente capítulo tres.

CAPÍTULO 3

DESARROLLO DEL SISTEMA

En este capítulo explicaré mi participación en el desarrollo de los módulos que fueron parte del sistema integral, el primero de ellos fue el monitor de la ventanilla, en segundo la ventanilla de divisas y el tercero llamado operaciones intermediación, los usuarios que operaban el sistema desde una interfaz con sus respectivos roles y permisos podían acceder a diferentes módulos sin cambiarse de un sistema o una plataforma a otra. Explicaré el proceso del desarrollo del diseño, diagramas, especificaciones para el desarrollo del código, estructura de documentos y formatos, orden del desarrollo del código fuente (clases java), desarrollo, configuración y parametrización de *web services*, reportes pdf, comprobantes, configuraciones, pruebas, procesos gráficos de los reportes y comprobantes, la configuración utilizada para compilar el código, errores encontrados, soluciones aplicadas, la liberación a producción con las responsabilidades que eso implicó. Conformé equipos de desarrollo, los capacité en el diseñado del sistema con su arquitectura aplicada, posteriormente les expliqué el negocio de cómo funcionaba y finalmente les asigné su respectiva parte y responsabilidad en el desarrollo. Primero explicaré el monitor efectivo en ventanilla en el siguiente apartado.

3.1 MONITOR EFECTIVO VENTANILLA

La función de este módulo fue monitorear la ventanilla aplicando reglas de negocio que detallaron los movimientos de la ventanilla de las sucursales, los usuarios fueron los cajeros, jefes y gerentes de sucursal, el desarrollo implicó nueva funcionalidad y actualización del código fuente en diferentes módulos del SI que se relacionaron con la ventanilla, propuse al equipo de trabajo cambios o ajustes mediante el diseño y rediseño de diagramas de flujo, realicé pruebas de escritorio con los nuevos cambios propuestos y finalmente después del análisis y con los resultados de esta pruebas, se tomó la desición de que sí era posible llevarlo a cabo, definí los tiempos de desarrollo, se realizó la notificación a la gerencia de sucursales quenes fueron los que realizaron el requerimiento, posteriormente la gerencia de sucursales dio la aprobación.

Una vez que la gerencia de sucursales dio su visto bueno al resultado del análisis de factibilidad, la gerencia de sistemas administrativos dio instrucciones para crear un grupo de trabajo, elaboré el plan de trabajo, las actividades que se realizaron incluyendo el riesgo e impacto, cambios a realizar, firmas de conformidad de las gerencias involucradas. Utilicé Microsoft *Project* para la planeación de tiempos inicial y final de cada tarea, así como la cantidad de recursos humanos que se asignaron.

Diseñé tablas, índices, secuencias, *constraint* de la base de datos, rediseño de tablas ya existentes aplicando los principios de la normalización garantizando la integridad referencial sin demasiados índices, ya que el acceso a estas tablas implementaba la concurrencia. El la figura 3.1 presento el modelo entidad-relación.

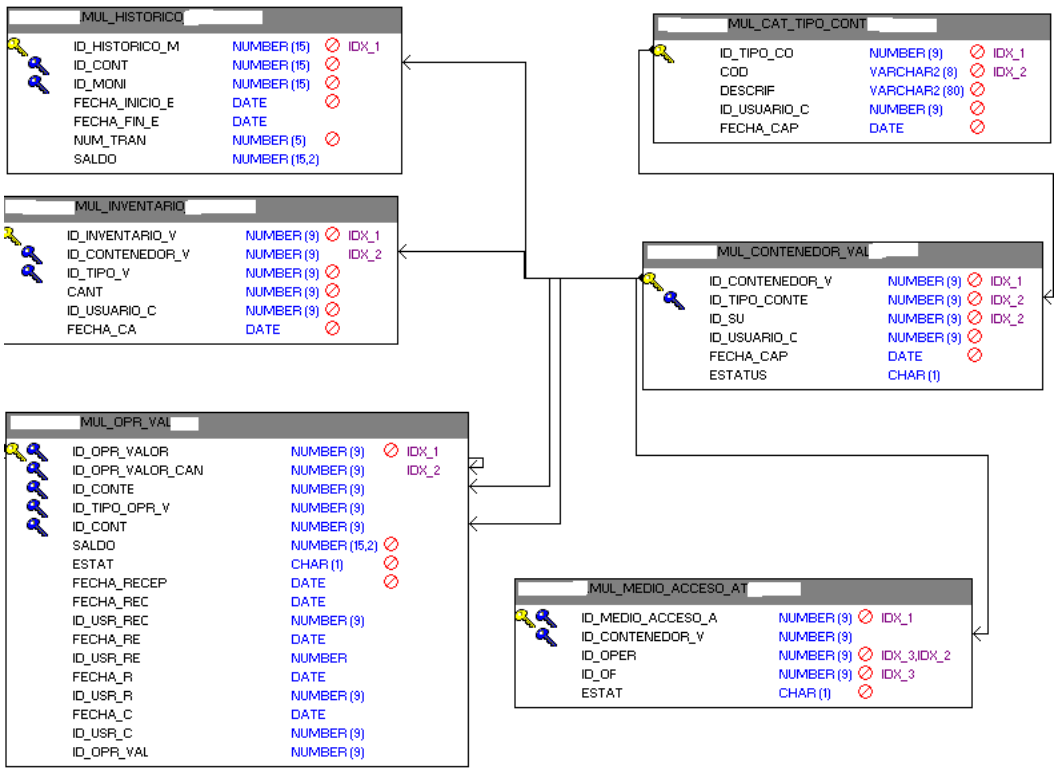


Figura 3.1: Modelo entidad-relación (Sistemas Administrativos GF, Eric López, 2008)

Para el acceso a la base de datos tomé en cuenta las políticas de la tabla 3.1 establecidas por la Gerencia de Control de Versiones, donde decidí utilizar el patrón DAO ya que con esa forma de acceder a la base de datos a un bajo nivel no tuve que obtener todos los campos de las tablas sino sólo los campos que necesité reduciendo el tiempo de respuesta en las consultas a diferencia de *hibernate* que aún contando con el atributo "*lazy*" que funcionaba bajo demanda y solo se consultaba hasta que se realizara alguna operación con los datos, de cualquier forma como la abstracción de la información se ocupó para ser actualizada, la consulta traería todos los objetos relacionados donde estos se cargarían a la máquina virtual de Java y provocarían bloqueos, es por eso que DAO fue la mejor elección.

Tabla 3.1: Políticas de uso de *Hibernate* y DAO (Control de Versiones y Configuración GF, 2007)

	Número Registros						
Relaciones	**10-50**	**50-100**	**100-200**	**200-400**	**400-500**	**500-1000**	**MAS 1000**
1-5	*HIBERNATE*	*HIBERNATE*	*HIBERNATE*	*HIBERNATE*	*HIBERNATE*	DAO	DAO
5-10	*HIBERNATE*	*HIBERNATE*	VISTA-MAP	VISTA-MAP	DAO	DAO	DAO
10-15	*HIBERNATE*	VISTA-MAP	VISTA-MAP	DAO	DAO	DAO	DAO
15-20	VISTA-MAP	VISTA-MAP	DAO	DAO	DAO	DAO	DAO
MAS DE 20	VISTA-MAP	VISTA-MAP	DAO	DAO	DAO	DAO	DAO

Generé diagramas de casos de uso del monitor efectivo ventanilla que me permitieron tener el panorama general del negocio y funcionamiento del nuevo desarrollo tal como lo detallo en el diagrama de la figura 3.2.

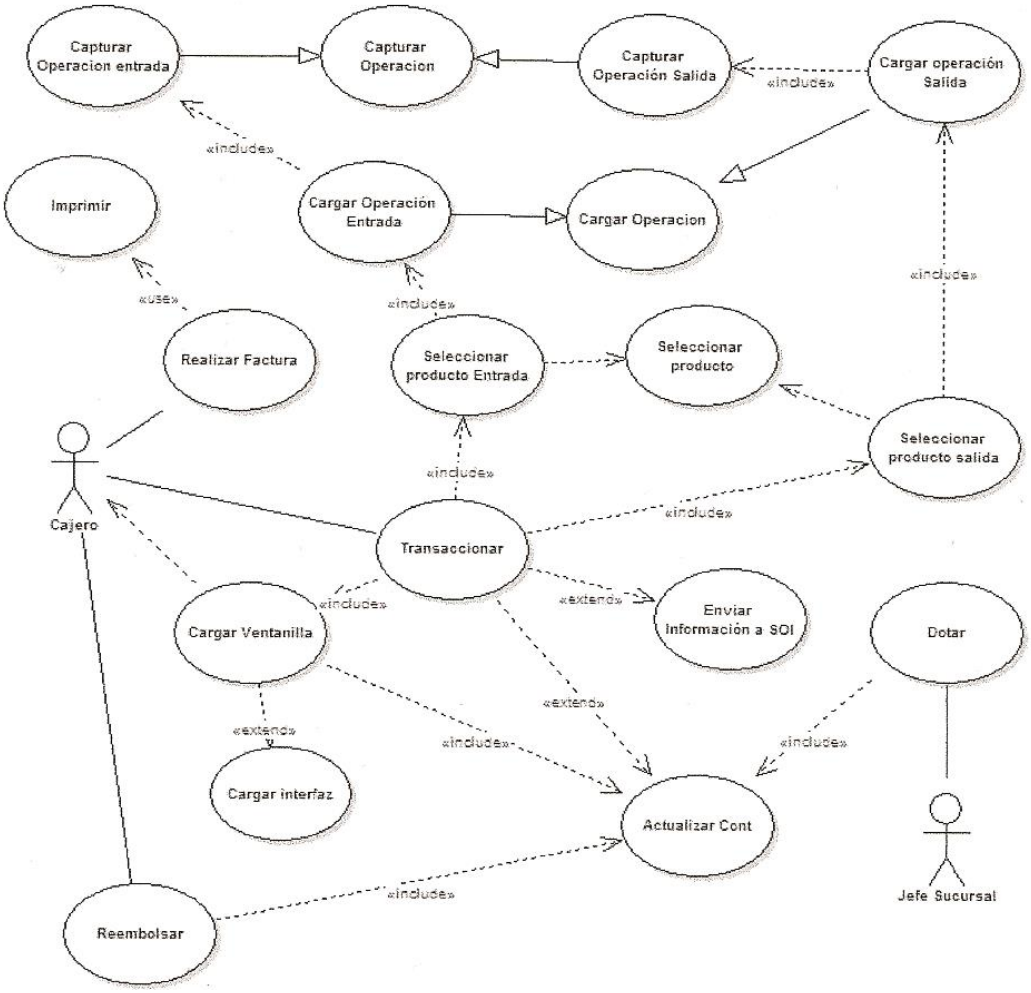


Figura 3.2: Diagrama de casos de uso (Sistemas Administrativos GF, Eric López, 2008)

En la figura 3.3 presento el diagrama de clases del monitor efectivo ventanilla

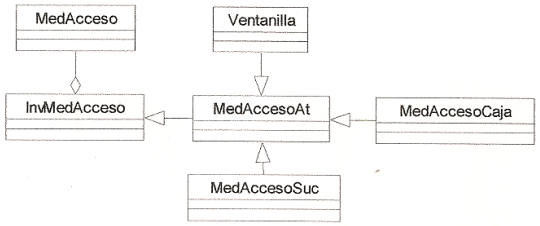


Figura 3.3: Diagrama de clases (Sistemas Administrativos GF, Eric López, 2008)

En la figura 3.4 presento el diagrama de componentes del monitor efectivo ventanilla.

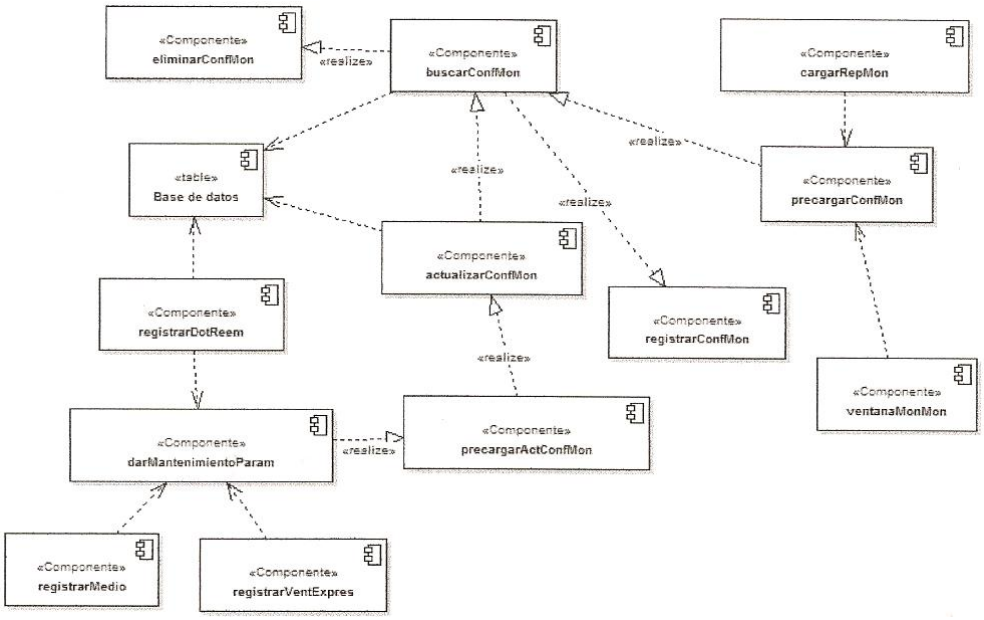


Figura 3.4: Diagrama de componentes de monitor efectivo (Sistemas Administrativos GF, Eric López, 2008)

En la figura 3.5 presento el diagrama de secuencia del monitor efectivo ventanilla.

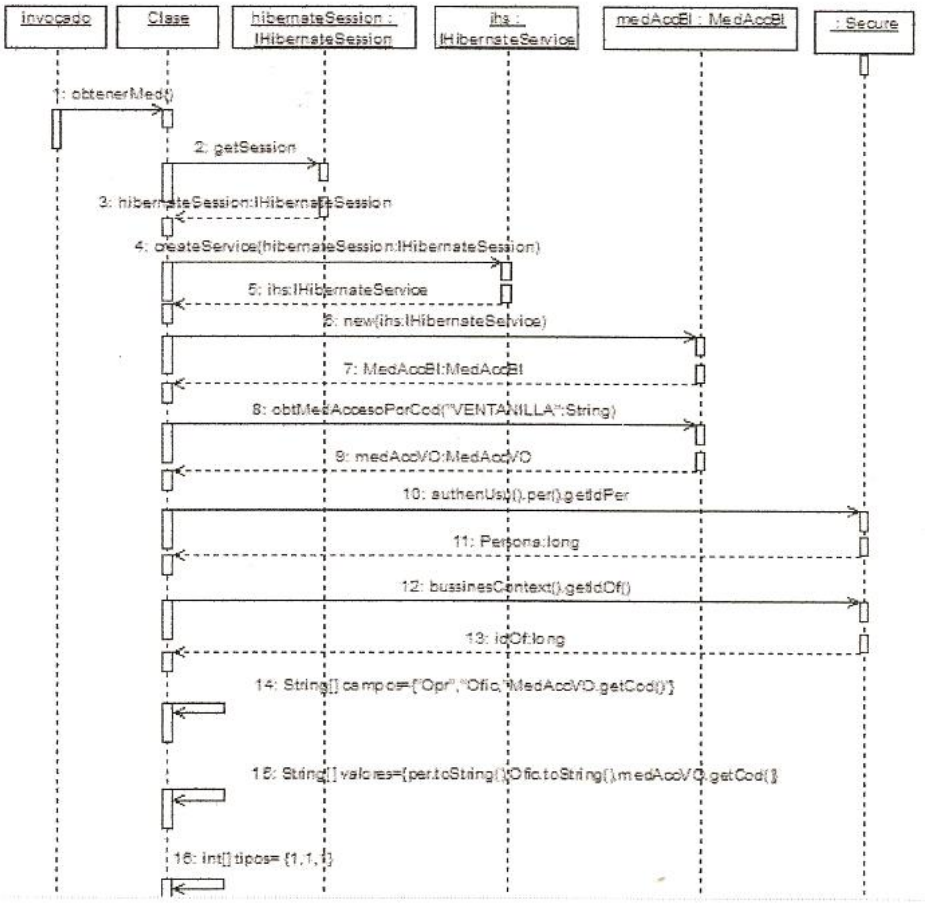


Figura 3.5: Diagrama de secuencia de monitor efectivo (Sistemas Administrativos GF, Eric López, 2008)

Para solicitar los cambios de la base de datos, realicé el documento oficial que contenía los scripts para dar de alta los nuevos objetos así como las modificaciones, estos nuevos objetos y modificaciones diseñadas las agregué al documento con los scripts, este documento contenía los scripts, áreas solicitantes, el objetivo, las áreas responsables e involucradas en de requerimiento así como el área responsable de ejecutar los scripts en la base de datos correcta, primeramente en ambiente de desarrollo, luego de usuario, luego QA, auditoría y finalmente producción. En la tabla 3.2 presento el formato oficial para solicitar los cambios de base de datos mediante scripts.

Tabla 3.2: Formato solicitud de Scripts SQL para cambios a base de datos (Sistemas Administrativos GF, Eric López, 2008)

Clave	Descripción del Cambio	Asignación de Permisos	S*	I*	U*	D*	E*
002-TA*	Crear la tabla: **M_TABLA_A** **L**os siguientes campos: CREATE TABLE NOMBRE_TABLA_A (CAMPO_NUMERICO1 NUMBER(15) NOT NULL, CAMPO_NUMERICO2 NUMBER(15) NOT NULL, CAMPO_FECHA DATE NOT NULL, CAMPO_NUMERICO3 NUMBER(5) NULL, CAMPO_DOUBLE NUMBER(15,2) NULL)	100-NNEG*	X	X	X	X	X
001-ES*	Crear llave primaria PK_ CAMPO_TABLA_A para la tabla M_TABLA_A con el campo CAMPO_TABLA_A	100-NNEG*	X	X	X	X	X
006-IN*	CREATE UNIQUE INDEX CAMPO_TABLA_A ON M_TABLA_A (CAMPO_DE_TABLA_A) CREATE INDEX IDX_M_NOMBRE_INDICE ON M_TABLA_A (CAMPO_IDEXADO, CAMPO2_INDEXADO)	100-NNEG*	X	X	X	X	X
004-SE*	Crear secuencia SEQ_ M_TABLA_A para la tabla M_TABLA_A(CAMPO_DE_TABLA_A)	100-NNEG*	X	X	X	X	X
002-TA*	Agregar a la tabla M_TABLA_B el campo CAMPO_AGREGADO de tipo CHAR(1) NULL ALTER TABLE M_TABLA_B ADD (CAMPO_AGRAGADO CHAR(1) NULL)	100-NNEG	X	X	X	X	X
001-ES*	Crear llave foránea FK_MUL_NOMBRE_DE_LLAVE_FORANEA entre la tabla **M_TABLA_A** y la tabla **M_TABLA_B** a través del campo CAMPO_COMPARTIDO ALTER TABLE MUL_TABLA_A ADD (CONSTRAINT FK_ NOMBRE_DE_LLAVE_FORANEA FOREIGN KEY (CAMPO_DE_TABLA_A) REFERENCES M_TABLA_B (CAMPO_DE_TABLA_B);	100-NNEG*	X	X	X	X	X

Desarrollo

Para el desarrollo del código fuente desde la capa de presentación hasta la capa de datos seguí las siguientes convenciones y especificaciones propuestas en el apartado 2.5, a continuación menciono el orden en el que lo realicé:

a) Especificaciones para el desarrollo de archivos *.jsp

En la parte superior de los archivos *.jsp importé las librerías (taglibs) de código estándar internacional de texto HTML así como las especificaciones de la herramienta que cumplió con el patrón MVC, estos taglib los incluí en el archivo web.xml tal como lo presento enseguida:

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"%>
<%@ taglib uri="/tags/struts-html" prefix="html"%>
<%@ taglib uri="/tags/struts-bean" prefix="bean"%>
<%@ taglib uri="/tags/struts-logic" prefix="logic"%>
<%@ taglib uri="/tags/pager-taglib" prefix="pg"%>
```

Agregé la descripción del tipo de documento (DTD) permitiendo elementos nuevos y elementos antiguos, además incluí la tag de inicio del código HTML y el cuerpo del documento:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html:html>
<head>
```

Incluí un archivo *.jsp predefinido que tenía el encabezado del GF, un archivo *.js que integraba funciones javascript que se podían llamar desde otra jsp y una hoja de estilos *.css que definió el tipo, color y tamaño de letra:

```
<jsp:include page="/jsp/core/multif/header/HeaderUtil.jsp" flush="true" />
  <script language="JavaScript" src="../js/unableControl.js"></script>
<link href="../../../../../../webapp/css/Stilos.css" rel="stylesheet" type="text/css">
```

La parte de funciones javascript se encontraba dentro de las siguientes etiquetas:

```
<script language="javascript">
```

En esta sección llamé funciones javascript ubicadas en el archivo *.js importado al inicio de la jsp:

```
deshabilitaAuto(true);
 creaLoading(true);
```

Realicé funciones dentro de esta misma sección:

```
function cache(){
var form = document.forms[0];
form.setAttribute("autocomplete","off");
}

function doSubmit(){
confirmarFormulario('MulNombreDelPathAction');
}
```

```
function doSubmitAct(){
confirmarFormulario('');
}
```

Cerré la sección de javascript y el encabezado con los siguientes tags:

```
</script>
</head>
```

Posteriormente inicié la parte del cuerpo en donde comencé agregando funciones que me ayudaron a controlar los errores y presentar de acuerdo a los mensajes de qué tipo de excepción se trataba, también me permitió cargar la página con un estilo efecto neblina con la función *blendTrans*(), el tiempo estándar de aparición del contenido fue de 0.5 segundos, al cargar la página, la función *noCache*() permitió limpiar el *caché* entre petición y petición y que no se arrastraran valores entre los formularios, la función de inicialización me permitió cargar las formas con valores refrescados:

```
<body onLoad="noCache(0); inicializacion();">
<jsp:include page="/jsp/error_messages.jsp"/>
<script language="javascript">document.body.style.filter='filter:
blendTrans(duration=0.5)';document.body.filters[0].Apply();</script>
```

Terminado el cuerpo de los archivos *.jsp escribí la etiqueta que cerraba el código HTML:

```
</body>
</html:html>
```

b) Especificaciones para el desarrollo de archivos **Form*.java

Para el desarrollo de las clases *Form.java importé el paquete de Struts que me pertitió heredar las validaciones con las funciones específicas para que el llenado de formularios tuviese los tipos de valores esperados, este paquete lo muestro en la siguiente línea:

```
import org.apache.struts.validator.ValidatorForm;
```

Implementé la herencia entre clases:

```
public class MulNombreDeLaClaseForm
extends ValidatorForm {
```

Estas clases contaron con los atributos necesarios para pintar la información en los archivos *.jsp:

```
 //~ Instance fields
 private String    abreviacionMoneda;
 private boolean muestraMensaje;
```

Y también por cada atributo se tenían los métodos *get*() y *set*():

```
public boolean isMuestraMensaje() {
        return muestraMensaje;
}
```

```
public void setMuestraMensaje(boolean muestraMensaje) {
        this.muestraMensaje = muestraMensaje;
}
public String getAbreviacionMoneda() {
        return abreviacionMoneda;
}
public void setAbreviacionMoneda(String abreviacionMoneda) {
        this.abreviacionMoneda = abreviacionMoneda;
}
```

c) Especificaciones para el desarrollo de archivos **Action*.java

En el desarrollo de estas clases tomé en cuenta la importación de clases que me permitieron utilizar clases y métodos que estaban dentro de estas para realizar las peticiones HTTP, la escritura de los archivos logs a bitácora, la generación de constructores de excepciones e inicializar formulacios controlados por el Servlet:

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.log4j.Logger;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
```

Al declarar la clase, heredé siempre de la clase *Action*:

```
public class MulNombreDeLaClaseAction
extends Action {
```

En el método principal siempre definí los parámetros:

```
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)
```

Las sesiones que aquí inicié no fueron para las conexiones a la base de datos si no del usuario firmado en el sistema, esta sesión me ayudó a guardar valores útiles:

```
final SessionUtil  session = new SessionUtil(request.getSession());
```

Estas clases interactuaron directamente con el archivo Struts y al final de cada clase *Action* cuando en las capas de negocio o datos tuve problemas tomé la decisión de lanzar excepciones.

d) Especificaciones para el desarrollo de archivos **Command*.java

En estas clases agregué código de negocio encapsulado con llamadas a uno o más métodos específicos, a uno o más métodos que conectaban a la base de datos, también dentro de estas clases importé paquetes con clases que utilicé para escribir en bitácora mediante archivos *.log, además en estos archivos inicié las sesiones que utilicé para accesar a la base de datos ya sea mediante *hibernate* o DAO:

```
import org.apache.log4j.Logger;
import org.hibernate.Session;
import com.esiglo.blf.command.CommandException;
```

En las siguientes líneas presento las clases que importé para generar la fabricación de los servicios de sesión y control de errores de *hibernate*:

```
import com.esiglo.plf.hibernate.HibernateServiceException;
import com.esiglo.plf.hibernate.HibernateServiceFactory;
import com.esiglo.plf.hibernate.HibernateSessionFactory;
import com.esiglo.plf.hibernate.IHibernateService;
```

Estas clases *command* las heredé de la clase GenericCommand la cual fue la clase padre que contenía métodos heredados útiles para el negocio:

```
public class MulNombreDeLaClaseCommand
extends GenericCommand {
```

e) Especificaciones para el desarrollo de archivos *VO.java

La misión de estas clases fue integrar todos los atributos de los objetos que viajaban en todas las capas, estos atributos los fui modificando de acuerdo a las necesidades de negocio así como en cada una de las capas desde la presentación hasta la de datos mediante serialización (empacado) de la información para poder ser enviada, posteriormente al llegar a su destino se deserializaba (desempacaba), esta implementación de la interface la describo en las siguientes líneas:

```
package mx.com.grupofinanciero.subfolder.core.multif
import java.io.Serializable;
public class MulNombreDeLaClaseVO
implements Serializable {
```

Para cada clase *Command* existió una clase *bean* llamado VO, esto lo configuré en el archivo **CommandMapping.xml*:***

```
<Command>
  <BuilderClass>
      mx.com.grupofinanciero.subfolder.MulNombreDeLaClaseVO
   </BuilderClass>
   <CommandClass>
    mx.com.grupofinanciero.subfolder.MulNombreDeLaClaseCommand
   </CommandClass>
 </Command>
```

f) Especificaciones para el desarrollo de archivos *BI.java

Estas clases tenían métodos específicos de negocio, también se conectaban a la base de datos vía *hibernate* o DAO. Los paquetes importados me sirvieron para escribir en los archivos de bitácora **ALL_dev.log** todo lo sucedido en la llamada a clases, manejo de advertencias y errores así como en conexiones *hibernate o* DAO:

```
import org.apache.log4j.Logger;
```

```
import com.esiglo.comun.excepcion.BusinessException;
import com.esiglo.plf.hibernate.HibernateServiceException;
import com.esiglo.plf.hibernate.IHibernateService;
```

En las clases BI implementé las interfaces con sus mismos nombres precedidas por la letra I:

```
public class MulNombreDeLaClaseBI
implements IMulNombreDeLaClaseBI {
```

g) Especificaciones para el desarrollo de los archivos *DAO.java

Para estas clases importé los siguientes paquetes para poder realizar conexiones a la base de datos y estas clases no las heredé de ninguna otra:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

Agregué campos de instancia y constructores:

```
//~ Instance fields
 protected Connection     conexion;
 private CallableStatement cst;
 private PreparedStatement preparedStatement;
 private String          sQuery;

 //~ Constructors
 /**
  * Crea un objeto <code>MulNombreDeLaClaseDAO</code>.
  * @param hibernateSession Documentar el parametro!
  */
 public MulNombredeClaseDAO(Session hibernateSession) {
   try {
     this.conexion = hibernateSession.connection();
   } catch(Exception e) {
     logger.error("No se puede establecer la conexion " + e);
   }
 }
```

h) Especificaciones del desarrollo de clases con terminación *.java

Apliqué las políticas de nombrado a las clases persistentes o entidad, estas clases llamadas *beans* viajaban de la capa de negocio a la capa de datos o viceversa para llevar o traer datos, realizando consultas, actualizaciones o eliminaciones a la base de datos, estas clases cumplieron con la importación del siguiente paquete:

```
import java.io.Serializable;
```

Estas clases implementaron otras clases e interfaces, además de que tenían sus propios atributos los cuales eran los datos que viajaban en las capas:

```
public class nombreDeClaseEntidad
implements Serializable, com.esiglo.plf.hibernate.persistence.IEntity {
```

Cada clase tenía sus atributos con sus respectivos métodos *get* y *set*:

```
private Long   idInventario;
private String serie;

/**
  * Documentar el objetivo del metodo!
  * @return Documentar el valor de retorno!
  */
 public Serializable getIdentity() {
  return idInventario;
 }
/**
 * Documentar el objetivo del metodo!
 * @param arg0 Documentar el parametro!
 */
 public void setIdentity(Serializable arg0) {
  idInventario = (Long) arg0;
}
 /**
  * Documentar el objetivo del metodo!
  * @return Documentar el valor de retorno!
  */
 public String getSerie() {
  return serie;
 }
 /**
  * Documentar el objetivo del metodo!
  * @param serie Documentar el parametro!
  */
 public void setSerie(String serie) {
  this.serie = serie;
 }
```

i) Especificaciones del desarrollo en los mapeos con terminación *.hbm.xml

El componente medular de la capa de persistencia estuvo implementado por hibernate, antes de que comenzara a realizar los mapeos ya contaba con el nuevo diseño de la base de datos incluyendo las modificaciones y nuevos objetos, además ya se había enviado al área de base de datos la solicitud con los scripts SQL correspondientes revisados por mí y confirmados por la jefa de sistemas. Con la herramienta Toad verifiqué los cambios que había solicitado estuvieran listos en la base de datos correcta. Cumpliendo con estos requisitos inicié el desarrollo de los mapeos con las siguientes especificaciones de hibernate.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
     <hibernate-mapping>
         <!--
 Created by the Middlegen Hibernate plugin
 http://boss.bekk.no/boss/middlegen/
 http://hibernate.sourceforge.net/
```

Definí la ruta y nombre de la persistente asociada al mapeo con su respectivo identificador y secuencia:

```
<class name="mx.com.grupofinanciero.subfolder.persistencia.NombreDeClasePersistente"
        table="MUL_NOMBRE_DE_LA_TABLA" lazy="false">
        <id name="nombreDelAtributoId" type="java.lang.Long" column="NOMBRE_DEL_ID_DE_TABLA">
                <generator class="sequence">
                <param name="sequence">SEQ_MUL_NOMBRE_DE_LA_TABLA</param>
                </generator>
        </id>
</class>
</hibernate-mapping>
```

Después agregué los atributos asociados a los campos de las tablas, los tipos y la longitud debían ser los mismos en el mapeo y la tabla de la base de datos:

```
<property name="nombreDelAtributo_1" type="java.lang.String" column="NOMBREDELCAMPO_1" not-null="true"
length="2"/>
<property name=" nombreDelAtributo_2" type="java.lang.Long" column=" NOMBREDELCAMPO_2" not-null="true"
length="10"/>
<property name=" nombreDelAtributo_3" type="java.lang.Long" column=" NOMBREDELCAMPO_3" not-null="true"
length="9"/>
…
<property name=" nombreDelAtributo_N" type="java.lang.Long" column=" NOMBREDELCAMPO_N" not-null="true"
length="9"/>
```

Con hibernate transformé la información relacional de la base de datos a objetos java, además de generar las sentencias SQL necesarias para obtener y actualizar los datos relacionales.

Nota. Si la longitud del tipo de datos en el mapeo era menor a la de la tabla sí podía persistir a la base de datos, pero de forma inversa no.

Para este desarrollo incluí características para el buen rendimiento, una de ellas fue que todas las clases tuvieron el comentario inicial como lo ilustro en las siguientes líneas:

```
/** ----------------------------------------------------------------
* Copyright (c) 2009 GF.
* Insurgentes Sur, México, D.F., C.P., MEXICO.
* Todos los Derechos Reservados*
* Este software contiene informacion totalmente confidencial
* propiedad del GF. Queda totalmente
* prohibido su uso o divulgacion en forma parcial o total
* y solamente podra ser utilizada de acuerdo a los términos
* y estatutos que determine el propio GF.
* ----------------------------------------------------------------*/
```

En la estructura genérica de las clases java integré la declaración de logger que es la que me permitió registrar en un archivo bitácora en qué momento iniciaba la llamada a la clase tal como lo muestro en la siguiente línea:

```
private static Logger      logger      = LogConfig.getLogger(MulNombredelaClaseAction.class);
```

La otra parte fue los comentarios de los parámetros del método principal como lo presento en las siguientes líneas:

```
//~ Methods ------------------------------------------------------------------------------------------------------------
/**
* Documentar el objetivo del metodo!
* @param mapping Documentar el parametro!
* @param form Documentar el parametro!
* @param request Documentar el parametro!
* @param response Documentar el parametro!
* @return Documentar el valor de retorno!
* @throws Exception Documentar la excepcion!
*/
```

El método principal llamado *execute*() tenía todos los parámetros, además incluí el objeto *logger* para llevar una bitácora de registro en el archivo ***.log** que registró el inicio de la ejecución de la clase, como lo presento en las siguientes líneas:

```
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse
response)
 throws Exception {
  if(logger.isDebugEnabled()) {
   logger.debug("Inicia Action: MulNombredelaClaseAction");
  }
```

La configuración para pasar de la capa de control a la capa de negocio la realicé en el archivo config/si-config/multif/CommandMapping.xml en donde a cada clase *Delegate* le asigné una clase *Command* y una clase VO, como lo muestro en el siguiente listado:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CommandMapping SYSTEM "./si_config/dtds/CommandMapping.dtd">
<CommandMapping>
    <Command>
        <BuilderClass>
            mx.com.grupofinanciero. subfolder.MulNombredelaClaseVO
        </BuilderClass>
        <CommandClass>
            mx.com.grupofinanciero.subfolder.MulNombredelaClaseCommand
        </CommandClass>
     </Command>

</CommandMapping>
```

Aplicando Struts tuve el control del origen y destino de cada petición del cliente, por donde pasaría el control del negocio, esta configuración la realicé en el archivo config\si-config\multif**struts-config-multif-valores.xml**, tal como lo muestro en las siguientes líneas:

```
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
        <!-- Form Bean Definitions -->
        <form-beans>
<!-- ************* Asignación del nombre del form y definición de su clase form ******** -->
<form-bean
name="MulNombreAsignadoAlForm"
        type="mx.com.grupofinanciero.subfolder.form.MulNombredelaClaseForm"/>
```

```
<!-- ********** Fin de asignación del nombre del form y definición de su clase form********** -->
</form-beans>
        <!-- Global forwards -->
        <global-forwards type="org.apache.struts.action.ActionForward">
        </global-forwards>
        <!--  Action Mapping Definitions  -->
        <action-mappings>
<!-- **********Inicio del negocio********** -->
                    <action path = "/MulNombreDelNegocio"
                            forward= "/jsp/core/multif/subfolder/nombreDeLaJsp.jsp"/>
                            <action path="/MulCargarNombreDelNegocio"
                            type="mx.com.grupofinanciero.subfolder.action.MulNombreDeLaClaseAction"
                    name="MulNombreAsignadoAlForm"
        validate="false"
                    input="/jsp/core/multif/subfolder/pantallaAnterior.jsp"
                      scope="request">
        <forward name="success"
        path="/jsp/core/multif/subfolder/nombre2DeLaJsp.jsp" />
        <forward name="fail"
        path="/jsp/core/multif/subfolder/nombre3DeLaJsp.jsp"/>
        </action>
        </action-mappings>
</struts-config>
```

Para el diseño del reporte utilicé la herramienta Ireport de Japer Reports, donde los parámetros fueron valores que envié desde el código java y los pasé a Jasper como torrente de bits, los *Fields* o campos fueron los valores que envié a la herramienta Jasper como java.lang.*String*, java.lang.*Object*, java.lang.*Boolean*, java.lang.*Byte*, java.util.*Date*, java.sql.*Timestamp*, java.lang.*Long*, java.lang.*Short*, java.math.*BigDecimal*. El archivo que generé con el diseño fue **reporteMonitorEfectivo.jrxml**, las variables como capacidad para realizar operaciones aritméticas así como números de página, número de reporte fueron características propias de IReports. Terminando la construcción y el desarrollo precompilé el reporte, en la figura 3.6 presento el reporte con la información final obtenida de la base de datos.

FECHA DE IMPRESION: jueves 26 marzo 2009
HORA DE IMPRESION: 08:23:41 PM

Banco

DOCUMENTO SIN VALOR OFICIAL

REPORTE DE MONITOR EFECTIVO

USUARIO: AOS

ID HIST	CONTEN	FECHA_INI	FECHA_FIN	NUM_TRAN	MON	SALDO	Razon Soc
47	61	26/03/2009 09:59:56	26/03/2009 10:10:29	5	MXP	120666.5	GCIA PLAZA
59	61	26/03/2009 12:43:37	26/03/2009 13:54:18	5	MXP	22565.95	GCIA PLAZA
61	61	26/03/2009 14:04:02	26/03/2009 14:30:09	5	MXP	21564.95	GCIA PLAZA
63	61	26/03/2009 14:31:31	26/03/2009 15:33:48	3	MXP	22564.95	GCIA PLAZA
68	61	26/03/2009 16:04:56	26/03/2009 16:15:34	2	MXP	22564.95	GCIA PLAZA
69	61	26/03/2009 16:21:47	26/03/2009 16:24:03	0	MXP	22564.95	GCIA PLAZA
70	61	26/03/2009 17:56:49	26/03/2009 19:52:05	2	MXP	22565.95	GCIA PLAZA

Figura 3.6: Reporte del monitor efectivo (Sistemas Administrativos GF, Eric López, 2008)

Una vez que precompilé el archivo del reporte con Ireport, generé el archivo **reporteMonitorEfectivo.jasper** en la carpeta local de desarrollo **C:\multif\banco\archivo con extensión** jasper que fue diferente a la de producción cuya ruta estaba en ***Server_Root*\multif\banco*. jasper.**

Con este precompilado generé las versiones más rápido y la configuración la realicé en el archivo **Proyecto/config/si-config/multif/*application-configuration.xml*** en donde coloqué una propiedad compuesta por la ruta y nombre del reporte con extensión *.xml que correspondió a la carpeta donde se depositaron los archivos *.jasper pre-compilados, esto lo muestro en las siguientes líneas:

```
<?xml version="1.0" encoding="UTF-8"?>
<application-configuration name="SI">
<version>Release 0.4.2.0</version>
<vendor>E</vendor>
<context name="reports">
<property name="jasper.compiler.classpath"><value>jasper.reports.compile.class.path</value></property>
<property name="jasper.jar"><value>./silibs/jasperreports-3.0.1.jar</value></property>
<property name="reports.home"><value>/reports</value></property>
<property name="multif.nombreDelReporte.xml"><value>multif/banco/nombreDelReporte.jasper</value></property>
</context>
</application-configuration>
```

Un problema que encontré en la generación del reporte fue que la librería no estaba bien configurada y tuve que volver a revisar el procedimiento, logré corregir y solucionar el error, esto me sucedió en ambientes locales de desarrollo y de usuario.

Compilación y configuración de negocio y *batch* (por lotes)

La compilación y construcción (*build*) del proyecto la realicé con tareas mecánicas repetitivas conocidas como software para la automatización de procesos java cuyo nombre oficial fue Apache *Ant*, a estos procesos de automatización se les llamó *targets* incluidos en un archivo llamado **dev_*build.*xml** para negocio y ***batch_build*_dev.xml** para batch. Dentro del archivo **dev_*build.xml*** se integró la tarea ***ant -f* dev_*build.xml deploy*_dev_exp** que realizó el deploy del proyecto de negocio, dentro del otro archivo ***batch_build*_dev.xml** estaba definida la tarea ***ant -f batch_build*_dev.xml *all*** utilizada para generar el deploy del proyecto de *batch*.

Para la automatizar la compilación del negocio y *batch* realicé los archivos **nb.cmd** y **compilaBatch.cmd** los cuales mandaron llamar a los archivos **compilaBatch.cmd y compilaNegocio.cmd** respectivamente, también me sirvieron para levantar el ambiente local de desarrollo, en el anexo de compilación de negocio y *batch* presento el contenido de estos archivos. Antes de compilar negocio y *batch* cargué las variables de ambiente y luego ejecuté los archivos **setenv.*cmd*** y ***batch*_setenv.*cmd*** ubicados en la carpeta ***C:**workspace*_Helios_Java5\ProjectoDeTrabajo*config***. En el anexo de configuración y compilación de negocio y *batch* detallo la configuración de estos archivos.

Pruebas

Terminando el desarrollo llevé acabo pruebas locales utilizando los escenarios de la matriz que realicé, las incidencias encontradas las resolví mediante correcciones inmediatas en el ambiente local de

desarrollo, una vez realizados los cambios, solicité permisos al área de arquitectura para subir las modificaciones al branch de trabajo realizando un control de cambios, primeramente sincronicé con el repositorio, luego verifiqué los cambios, posteriormente apliqué *commit*. Al momento en que se generó la versión en el ambiente de desarrollo, nuevamente apliqué los escenarios de la matriz de pruebas para validar los ajustes.

Los problemas que encontré fueron la concurrencia, cuando realicé transacciones al mismo tiempo el componente *dispatcher* (despachador de transacciones) no funcionaba correctamente, el problema es cuando una transacción tomaba un registro con id=x de la base de datos, en ese momento se actualizaban montos y posteriormente se liberaba este registro y si en ese mismo instante, se realizaba otra transacción que involucraba en mismo registro con el id=x, donde el monto tomado no era el de la transacción que entró primero al *dispatcher* si no que tomaba el mismo como si la transacción que entró primero no se hubiese realizado y como consecuencia se recalculaban los montos y se descuadraban los saldos de los movimientos. Para solucionar este problema, realicé lo siguiente:

a) Asigné un segundo de separación entre una transacción y otra al método que aplicó el tiempo de *dispatch*.

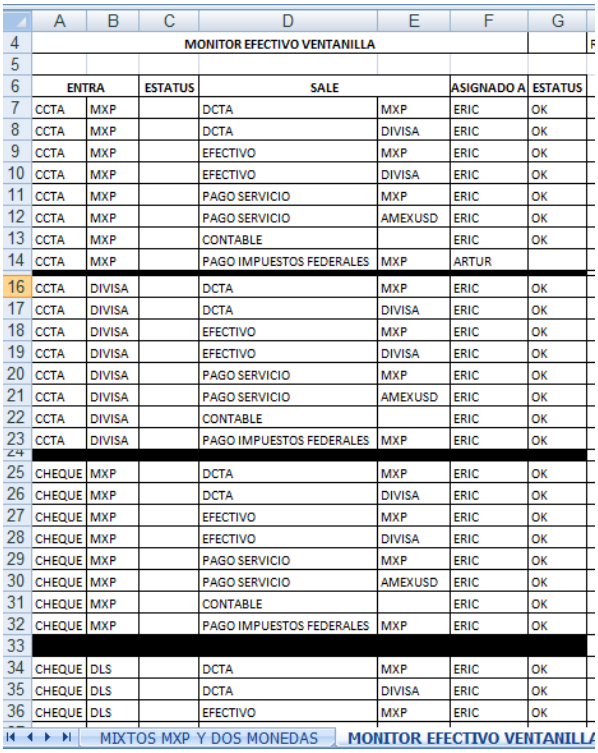
b) Implementé el bloqueo de cada registro identificado por id, sólo que si se persistía en una tabla especial y cada vez que se quería alterar el registro, no se permitía tomarlo ni modificarlo, se enviaba un mensaje en pantalla de que el registro estaba en uso.

c) Implementé bloqueo por registro con llave primaria compartida, es decir dos campos de tipo numérico los definí como llave primaria, mediante la sentencia *UPDATE* TABLA *SET* CAMPO=CAMPO *WHERE* ID_1=X *AND* ID_2=Y, estas medidas de solución las apliqué con la llamada a un método que utilizaba esta sentencia para bloquear el registro.

Realicé las correcciones modificando el código correspondiente a petición del usuario, volví a realizar pruebas locales a nivel desarrollo, cuando corregí las incidencias solicité permisos al área de arquitectura para poder subir las modificaciones al branch de trabajo y que se realizara la versión a desarrollo, al estar lista esta versión, nuevamente realicé las pruebas, al no haber incidencias, le notifiqué al usuario que la versión estaría lista una vez que el área de arquitectura montara el branch de trabajo en ambiente servidor apuntando a la base de datos de usuario, solamente los usuarios podían probar en este ambiente, la gerencia de arquitectura notificó a la gerencia de sistemas administrativos, que la versión ya estaba lista, en ese momento se procedió a realizar las pruebas de forma coordinada, tanto el usuario solicitante del proyecto como el área de desarrollo y control de calidad QA estuvieron presentes en las pruebas. El usuario validó el funcionamiento, la vista final, y verificó que no existieran incidencias, al no presentarse estas se creó el documento BI-PC el cual contenía el objetivo, el equipo de trabajo, condiciones de prueba, datos y escenarios de prueba, los registros de estructura de conversión, programas nuevos, archivos, procesos y procedimientos nuevos, también contenía la aprobación del usuario, del desarrollador y área QA.

Terminadas las pruebas de usuario y QA, el área solicitante que fue la gerencia de sucursales firmó el documento de aceptación BI-DA en donde quedó formalmente aceptado el análisis, diseño y desarrollo

del requerimiento, con este documento, el proyecto quedó listo para su puesta en producción y en consecuencia fue liberado, en la figura 3.7 presento los resultados de los escenarios que tomé en cuenta para llevar a cabo las pruebas finales del monitor de efectivo ventanilla.



	A	B	C	D	E	F	G
4	MONITOR EFECTIVO VENTANILLA						
5							
6	ENTRA		ESTATUS	SALE		ASIGNADO A	ESTATUS
7	CCTA	MXP		DCTA	MXP	ERIC	OK
8	CCTA	MXP		DCTA	DIVISA	ERIC	OK
9	CCTA	MXP		EFECTIVO	MXP	ERIC	OK
10	CCTA	MXP		EFECTIVO	DIVISA	ERIC	OK
11	CCTA	MXP		PAGO SERVICIO	MXP	ERIC	OK
12	CCTA	MXP		PAGO SERVICIO	AMEXUSD	ERIC	OK
13	CCTA	MXP		CONTABLE		ERIC	OK
14	CCTA	MXP		PAGO IMPUESTOS FEDERALES	MXP	ARTUR	
16	CCTA	DIVISA		DCTA	MXP	ERIC	OK
17	CCTA	DIVISA		DCTA	DIVISA	ERIC	OK
18	CCTA	DIVISA		EFECTIVO	MXP	ERIC	OK
19	CCTA	DIVISA		EFECTIVO	DIVISA	ERIC	OK
20	CCTA	DIVISA		PAGO SERVICIO	MXP	ERIC	OK
21	CCTA	DIVISA		PAGO SERVICIO	AMEXUSD	ERIC	OK
22	CCTA	DIVISA		CONTABLE		ERIC	OK
23	CCTA	DIVISA		PAGO IMPUESTOS FEDERALES	MXP	ERIC	OK
25	CHEQUE	MXP		DCTA	MXP	ERIC	OK
26	CHEQUE	MXP		DCTA	DIVISA	ERIC	OK
27	CHEQUE	MXP		EFECTIVO	MXP	ERIC	OK
28	CHEQUE	MXP		EFECTIVO	DIVISA	ERIC	OK
29	CHEQUE	MXP		PAGO SERVICIO	MXP	ERIC	OK
30	CHEQUE	MXP		PAGO SERVICIO	AMEXUSD	ERIC	OK
31	CHEQUE	MXP		CONTABLE		ERIC	OK
32	CHEQUE	MXP		PAGO IMPUESTOS FEDERALES	MXP	ERIC	OK
34	CHEQUE	DLS		DCTA	MXP	ERIC	OK
35	CHEQUE	DLS		DCTA	DIVISA	ERIC	OK
36	CHEQUE	DLS		EFECTIVO	MXP	ERIC	OK

MIXTOS MXP Y DOS MONEDAS | MONITOR EFECTIVO VENTANILLA

Figura 3.7: Resultados de escenarios aplicados a monitor de efectivo (Sistemas Administrativos GF, Eric López, 2009)

Manejo de Errores y Excepciones

Clasifiqué los errores en excepciones de sistema y excepciones de negocio donde conté con un mecanismo para cachar y registrar en bitácora cualquier excepción inesperada que pudiera surgir, las excepciones de negocio no las utilicé para el flujo de la aplicación ya que el mismo flujo de la aplicación guió las operaciones que fueron válidas desde el punto de vista de negocio. Cuando requerí utilizar excepciones de negocio me aseguré que fuese una excepción de este tipo ya que los errores que ocurrieron en la capa de negocio no todos fueron de este tipo también fueron de tipo *NullPointerException* o alguna otra excepción resultado de algún error de programación que no incluí en los escenarios, además tomé en cuenta los siguientes casos:

a) No debí lanzar excepciones de sistema como *SQLException* o *NullPointerException* dentro de una excepción de negocio.
b) En una excepción debí tomar acciones como resultado del error o bien relanzar y en su caso consumir la excepción que registré en bitácora. Si manejaba la excepción tendría que registrarla en bitácora, si re-lanzaba la excepción NO debía registrarla en bitácora para evitar una saturación con información redundante de la misma excepción.
c) Excepto por las excepciones de negocio, que como ya se mencionó se debían de mantener al mínimo, el resto de las excepciones debían ser de tipo *RuntimeException*, es decir que no serían atrapadas ni declaradas en el código invocante. Dado a que algunos de los productos utilizados, incluido el mismo API de Java, arrojaba en ocasiones excepciones de tipo *Exception*, era

necesario envolver dicha excepción en una de tipo *RuntimeExceptionWrapper* para que el código invocante no tuviese que lidiar con el manejo de dichas excepciones (a menos de que fueran recuperables), por ejemplo:

```
} catch (IllegalAccessException e) {
throw new RuntimeExceptionWrapper(exception);
}
```

Una vez terminado el monitor efectivo, continué con el módulo de la ventanilla de divisas el cual describo en la siguiente sección.

3.2 VENTANILLA DIVISAS

Elaboré el plan de trabajo incluido en el documento BI-PT con el porcentaje de responsabilidad de cada gerencia tanto de sucursales como de sistemas administrativos, además de las actividades a realizar, los pendientes de arquitectura, infraestructura, porcentaje de riesgo e impacto y firmas de conformidad.

Realicé la distribución de responsabilidades mediante asignaciones con base en habilidades y experiencias de cada uno de los integrantes del equipo, en el documento BI-ET definí las especificaciones técnicas, el objetivo de trabajo, la aprobación del diseño técnico, este documento me sirvió de referencia para el desarrollo y programación, también propuse el diseño de los nuevos objetos de la base de datos definiendo nombres, longitud y tipos de campos de cada uno de ellos cumpliendo con la nomenclatura estándar del SI.

En la figura 3.8 presento los diagramas de casos de uso y en la figura 3.9 el diagrama de componentes realizados para ventanilla de divisas.

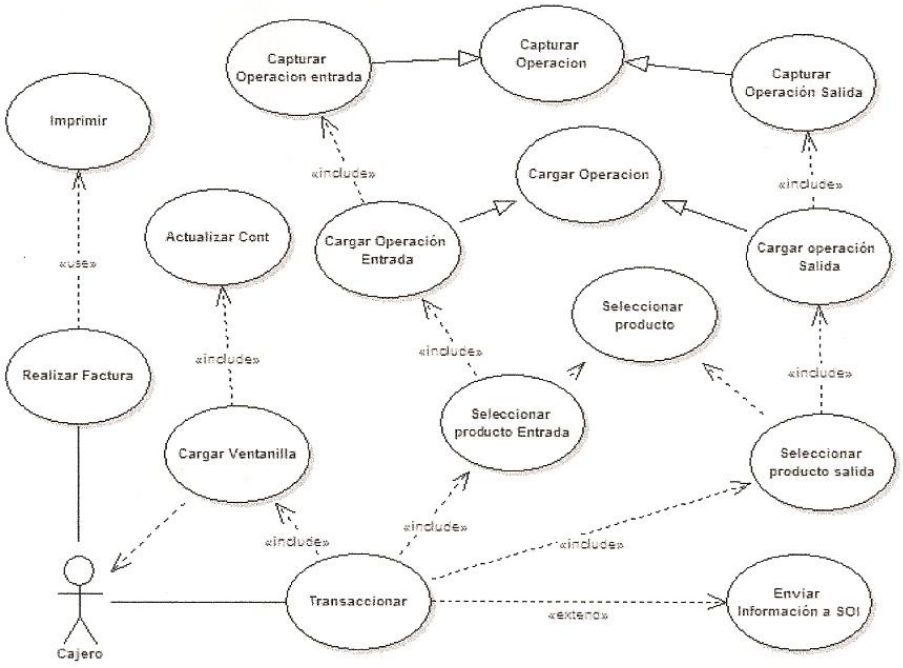


Figura 3.8: Diagrama de casos de uso ventanilla divisas (Sistemas Administrativos GF, Eric López, 2009)

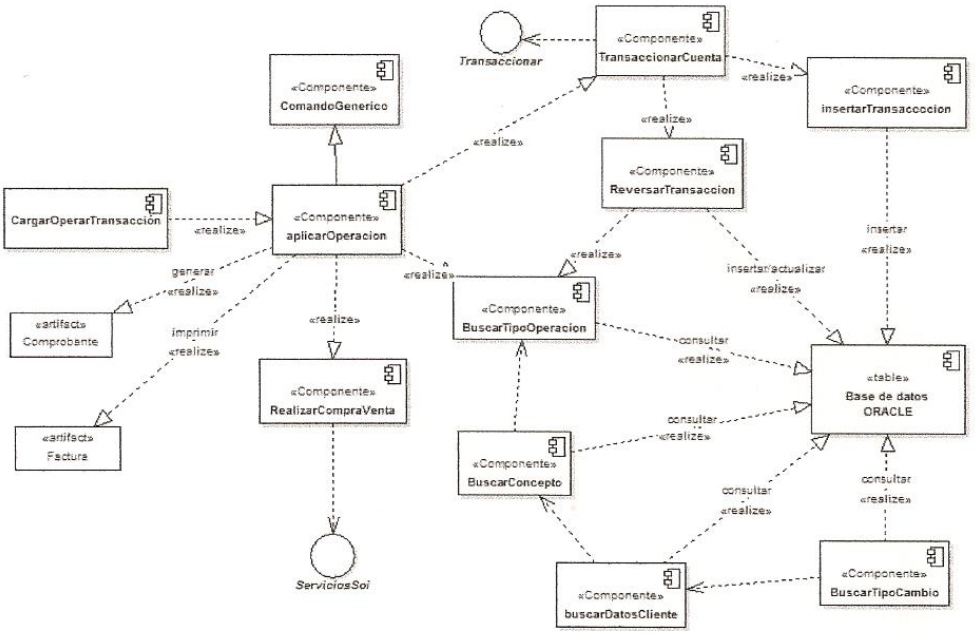


Figura 3.9: Diagrama de componentes ventanilla divisas (Sistemas Administrativos GF, Eric López, 2009)

En la figura 3.10 y 3.11 presento los diagramas de secuencia para ventanilla de divisas.

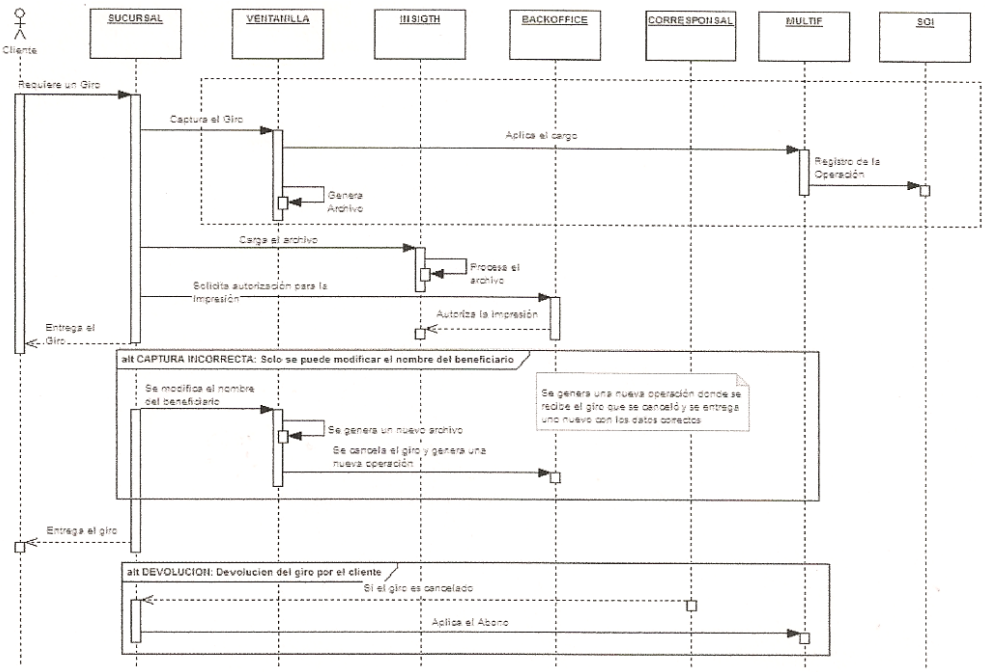


Figura 3.10: Diagrama de secuencia ventanilla divisas (Sistemas Administrativos GF, Eric López, 2009)

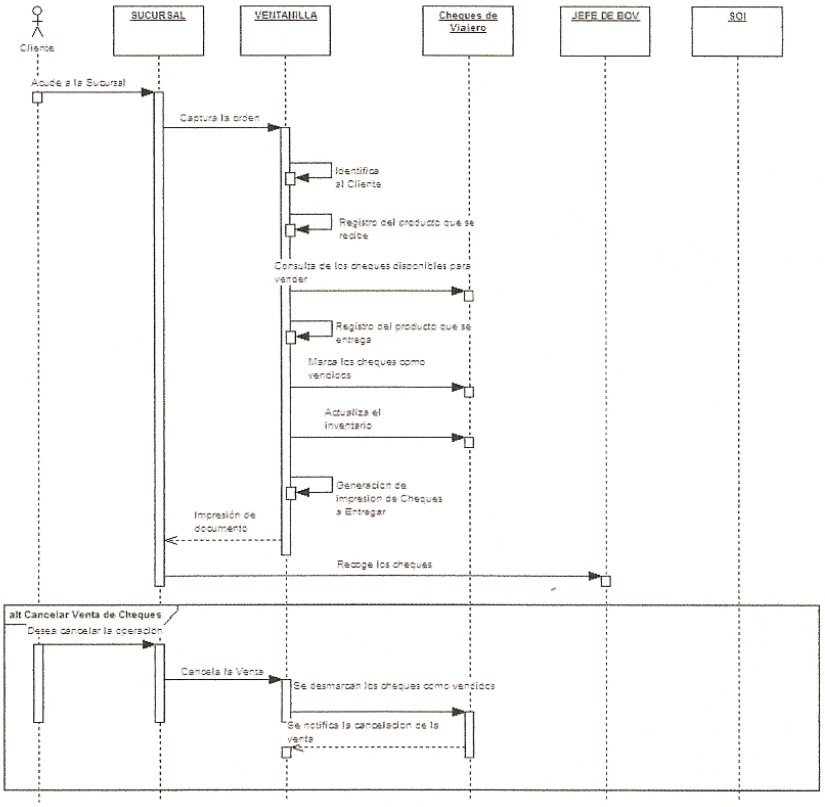


Figura 3.11: Diagrama de secuencia ventanilla divisas (Sistemas Administrativos GF, Eric López, 2009)

En el diagrama de clases de la figura 3.12 presento el diseño final de cómo quedó constituida la relación entre las clases generadas en el diseño para la ventanilla de divisas.

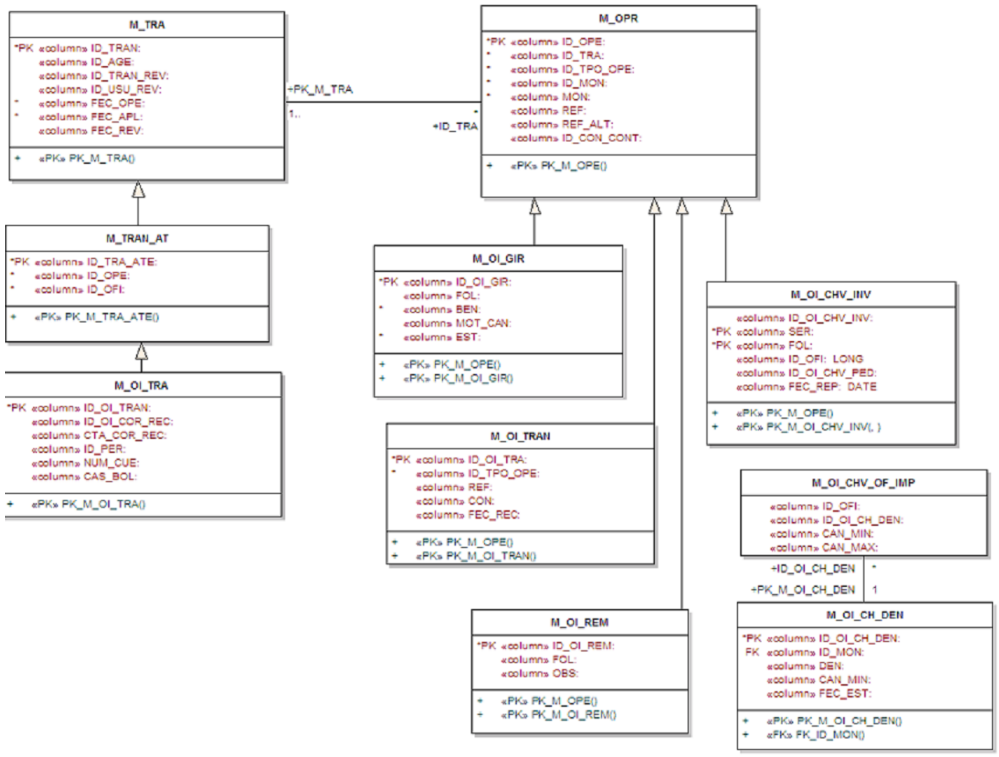


Figura 3.12: Diagrama de clases ventanilla divisas (Sistemas Administrativos GF, Eric López, 2009)

Realicé un archivo *.SQL donde integré los scripts para crear los objetos de la base de datos tanto tablas, *constraints*, secuencias, índices y actualizaciones de los mismos, este archivo me ahorró tiempo de desarrollo ya que lo integré como parte de la solicitud de cambios a la base de datos que se entregó al área de base de datos la cual fue responsabilidad de ejecutar estos scripts.

Desarrollo

Para el desarrollo del código fuente de este módulo utilicé las especificaciones del apartado 3.1 para cumplir con los estándares de codificación y nombrado.

Desarrollo de la Factura de Divisas y Nota de Venta

Para el desarrollo de la factura y nota de venta primeramente revisé que las impresoras estuvieran dedas de alta en central AdminUnix donde estaba configurada la instancia de adobe para imprimir a impresora o convertir archivos de XML a PDF, si no estaba dada de alta la impresora, no se podía imprimir los comprobantes, en la figura 3.13 presentó el flujo de la generación de la factura y nota de venta.

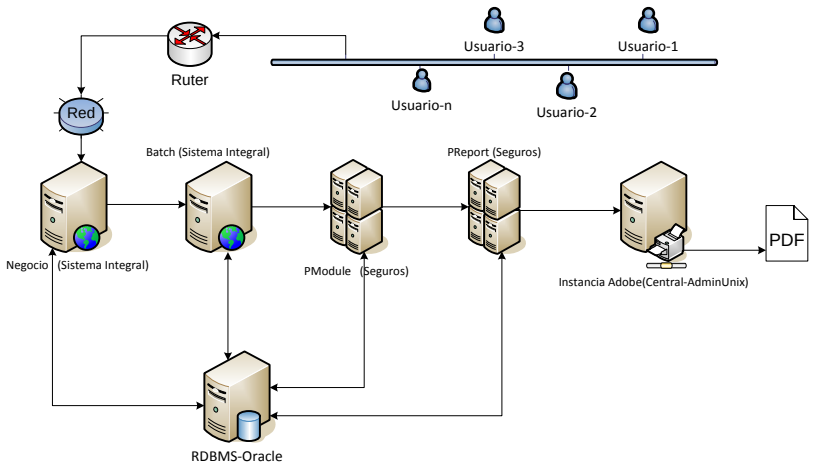


Figura 3.13: Proceso de generación de comprobantes y reportes (Sistemas Administrativos GF, Eric López, 2009)

Los proyectos relacionados con el desarrollo de la factura de divisas y nota de venta los explico enseguida:

a) ***Negocio.*** proyecto integrado por el código fuente como parte del SI del GF.

b) ***Batch***. proyecto parte del SI que integró el código fuente para la llamada al *Job* (tareas en lotes asíncronos), los cuales lanzaron la petición donde el Job tenía un número identificador único con sus perámetros correspondientes y la configuración estaba guardada en la base de datos.

c) ***Pmodule***. Este proyecto contenía el código fuente para configurar e inicializar las carpetas donde se depositaron los archivos *.xml generados y enviados a Central (instancia adobe) así como también asociar el *Job* con la plantilla correspondiente, los cambios en este proyecto fueron responsabilidad del área de Seguros.

d) ***PReport***. Este proyecto tenía el código fuente para recuperar la información necesaria de la base de datos con las rutas incluidas en clases específicas y que el archivo *.xml ya tuviera la información para colocarlo en la ruta correspondiente, yo realicé cambios en este proyecto, el personal de seguros se encargó de subirlos al branch ya que ellos eran responsables del proyecto PReport.

e) ***Central***. El área responsable de administrar este proyecto fue AdminUnix donde existía la instancia para poder generar archivos *.pdf o enviarlas a impresora.

Para mapear los datos obtenidos de la base de datos con el diseño utilicé dos diccionarios de datos y dos archivos que fungieron como plantillas con extensiones *.MDF, estas plantillas me las proporcionó la gerencia de sucursales y los diccionarios tenían el nombre del campo, número de caracteres, línea y número de pagina.

Compartí tres branch de trabajo, uno fue el SI que era el enlace entre el sistema del banco y los componentes, otro llamado Pmodule el cual tenía la configuración de las conexiones hacia las carpetas donde se depositarían los archivos procesados XML, el otro *branch* se llamó Preports el cual recibía

parámetros enviados por el módulo de banco del SI, estos parámetros los envié mediante un objeto *Hashtable*(). Dentro del proyecto del proyecto Preports existió un archivo llamado **config.xml***,* el cual configuré para poder realizar las pruebas locales. Los proyectos que descargué mediante SVN los presento en las siguientes líneas:

http://146.219.2.1/com/Preports/branch/ Para Preports

http://146.219.2.1/com/Pmodule/branch/ Para Pmodule

Las clases que relicé para iniciar el proceso de generación de archivos *.xml fueron FacDivisa.java y NotaVentaChViajero.java que heredaron de la clase AbstractPrinter.java que contenía el método *public void* **printReport***(PrintContext context, Writer writer) d*onde ***context*** contenía todos los parámetros a imprimir y ***writer*** era la acción a realizar.

Para llevar a cabo las pruebas locales configuré el tipo de impresión dentro del archivo config.xml del proyecto Preports, esta configuración la conforme de la clase y los parámetros que se escribieron en los archivos cuyos nombres fueron FacDivisa.xml y NotaVenta.xml, en las siguientes líneas presento el contenido de la configuración:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
```

Agregué la clase que llamé para generar el comprobante como lo muestro enseguida:

```
<bean class="java.lang.String" name="clase.reporte">
        <constructor-arg>
        <!-- Este es el nombre de la clase que generó el reporte, actualizar con el nombre calificado de la clase que se desea probar -->
         <value>mx.com.grupofinanciero.subfolder.rep.FacDivisa</value>
</constructor-arg>
</bean>
```

Para iniciar el contexto agregué la siguiente línea:

```
<bean class="mx.com.grupofinanciero.subfolder.print.module.job.PrintContextImpl" name="context">
```

La configuración de las propiedades que identificaron al usuario, sucursal y tipo de documento la presento en las siguientes líneas:

```
<property name="codigoUsuario"> <value>1</value>        </property>
<property name="oficina"> <value>100</value> </property>
<property name="tipoDocumento"> <value>xml</value> </property>
```

Configuré los parámetros del objeto *Hashtable* que viajaron desde la capa de negocio hasta el negócio de Preports, estos parámetros los explico enseguida:

```
<property name="parameters">
<bean class="java.util.Hashtable">
<constructor-arg>
        <map>
```

```
        <entry key="fechaOper"><value>jun 21 2009</value>       </entry>
        <entry key="montoRecibir"><value>50.0</value></entry>
        <entry key="montoEntregar"><value>745.9</value></entry>
        <entry key="tipoCambio1"><value>1.211868</value></entry>
        <entry key="tipoCambio2"><value>12.31</value></entry>
        <entry key="divisaRecibir"><value>EUR</value></entry>
        <entry key="divisaEntregar"><value>MXP</value></entry>
         <entry key="observaciones"><value>EFECTIVO Recibido EUR 50.0</value></entry>
        </map>
</constructor-arg>
</bean>

</property>
```

En las siguientes líneas presento la configuración del job dentro de la etiquete ***<value>***:

```
<!-- Datos referentes al header -->

        <bean class="java.lang.String" name="job">
        <constructor-arg>
        <value>FACTURA_TRANSF</value>
</constructor-arg>
</bean>
```

Coloqué el nombre del archivo Factura.XML que tenía todos los datos en etiquetas para ser convertidas en un reporte PDF o enviadas a impresora tal como lo presento en las siguientes líneas:

```
<!-- Ruta del archivo de salida -->
<bean class="java.io.OutputStreamWriter" name="archivo.salida">
        <constructor-arg>
        <bean class="java.io.FileOutputStream">
                <constructor-arg>
                <value>C:\si_spool\adobe\central\server\dat\Factura.XML</value>
        </constructor-arg>
</bean>
</constructor-arg>
</bean>
```

En las siguientes líneas presento la configuración de conexión a la base de datos:

```
<!-- This part of the configuration is NOT to be modified -->
<bean class="org.springframework.jdbc.datasource.DriverManagerDataSource" name="data.source">
<constructor-arg index="0"><value>oracle.jdbc.OracleDriver</value></constructor-arg>
<constructor-arg          index="1"><value>jdbc:oracle:thin:@nombreservidor.banco.gf.mx:1521:SID</value></constructor-arg>
<constructor-arg index="2"><value>USER</value></constructor-arg>
<constructor-arg index="3"><value>PASSWORD</value></constructor-arg></bean>
```

Estructura de los archivos XML generados para cada comprobante

Al realizar el proceso completo y las pruebas, los archivos se generaron con la información correspondiente para que se procesaran en formato *.PDF ya que en el encabezado le coloqué el

nombre del job a ejecutar y la ruta en donde se depositaría el archivo procesado y convertido a PDF como lo muestro en las siguientes líneas:

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^ ***job*** **FACTURA_TRANSF** -c1 -z***/si_spool/adobe/central/server/dat/Factura.PDF -aspPDF*** -aspPDF -ass108 useFormCommand=1 data.record=2

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^***job NOTA_VEN*** -c1 -z***/si_spool/adobe/central/server/dat/NotaVenta.PDF*** -aspPDF -ass108 useFormCommand=1 data.record=2

Cuando quise que los archivos se enviaran a impresora, en el encabezado asigné que fueran a impresión, además coloqué el nombre del job a ejecutar, nombre de la impresora y ruta donde se depositaría el archivo procesado tal como lo muestro a continuación:

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^***job*** **FACTURA_TRANSF** -c1 -z"***lp -DESA*** -o -dp -o -Z!,p" -ass5 useFormCommand=1 data.record=2

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^job **NOTA_VEN** -c1 -z"**lp –DESA** -o -dp -o -Z!,p" -ass5 useFormCommand=1 data.record=2?>

Después del encabezado cloqué las etiquetas <CUERPO> y <NOMBREDELJOB>, seguidos de los parámetros enviados tal como lo describo en las siguientes líneas:

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^**job FACTURA_TRANSF** -c1 -z/si_spool/adobe/central/server/dat/**FACTURA.PDF** -asp**PDF** -ass108 useFormCommand=1 data.record=2 ?>
<**CUERPO**>
<**FACTURA_TRANSF**>
<IMP_FECHA_VALOR_RECEP>*06/09/2008*</IMP_FECHA_VALOR_RECEP>
<IMP_FECHA_OPERACION_RECEP>*06/09/2008*</IMP_FECHA_OPERACION_RECEP>
<IMP_NOMBRE_CLIENTE_RECEP>*ERIC*</IMP_NOMBRE_CLIENTE_RECEP>
<IMP_OBSERVACIONES_RECEP>*PAGO DEL CLIENTE*</IMP_OBSERVACIONES_RECEP>
<IMP_PRODUCTO_RECEP> *ENTRA EFECTIVO* </IMP_PRODUCTO_RECEP>
</**FACTURA_TRANSF**>
</**CUERPO**>

Para las pruebas locales de desarrollo, la nota de venta la envié a impresora, le asigné el nombre DESA como lo ilustro enseguida:

<?xml version="1.0" encoding="ISO-8859-1" ?><?jetform ^job **NOTA_VEN** -c1 -z"**lp –DESA** -o -dp -o -Z!,p" -ass5 useFormCommand=1 data.record=2?>
<**CUERPO**>
<**NOTA_VEN** >
<IMP_DIVISAS>*DOLARES*</IMP_DIVISAS>
<IMP_NOM_CLIE>*ERIC*</IMP_NOM_CLIE>
<IMP_DENOM_1>*20*</IMP_DENOM_1>
<IMP_PAIS>*MEXICO* </IMP_PAIS>
<IMP_DENOM_4>*1000*</IMP_DENOM_4>
<IMP_TOT_VENT>*1190*</IMP_TOT_VENT>
<IMP_FECH>*16/02/2010*</IMP_FECH>
<IMP_TIP_CAM>*10.43*</IMP_TIP_CAM>
</**NOTA_VEN**>
</**CUERPO**>

Para los archivos generados en formato *.pdf, sus correspondientes *.xml se colocaron en la ruta FTP ***/si_spool/adobe/central/server/dat/*** donde se encontraba el Cron (Administrador regular de procesos en segundo plano), cuando el *thread listener* identificaba la existía un archivo xml, los tomaba y los procesaba para colocar otro archivo con extensión *.pdf. Cuando ocurrieron errores se escribió en archivos botácora con el nombre ***jfserver.log*** en un directorio arriba de la carpeta dat, la configuración la explico enseguida:

Configuración de los reportes:

Solicité al área de Producción realizar las siguientes tareas para generar la factura de divisas:

1. Asignar el nombre del reporte como: **FACDIV**

2. Colocar la nomenclatura inicial al archivo XML:
   FACDIV**_XXXXX.XML**
3. Las rutas en donde se depositaron los archivos XML fueron:
   /si_multif/estados_cta/00010/FACDIV_XXXXX.XML
   /si_spool/adobe/central/server/dat/FACDIV_XXXXX.XML
4. Empatar las versiones de Preports Pruebas Usuario con la de Producción.
5. La clase que se mandaba llamar desde el reporte FACDIV fue:
   package mx.com.grupofinanciero.subfolder.rep.**Factura.java**

Solicité al área de Producción realizar las siguientes tareas para generar la nota de venta:

1. Asignar el nombre del reporte como: **NOTACH**

2. Colocar la nomenclatura inicial al archivo XML:
   NOTACH_XXXXX.XML
3. Las rutas en donde se depositaron los archivos XML fueron:
   /si_multif/estados_cta/00010/ NOTACH_XXXXX.XML
   /si_spool/adobe/central/server/dat/ NOTACH_XXXXX.XML
4. Empatar las versiones de Preports Pruebas Usuario con la de Producción.
5. La clase que se mandaba llamar desde el reporte NOTACH fue:
   package mx.com. grupofinanciero.subfolder.rep.**NotaDeVen.java**

Solicité al área deAdminUnix realizar las siguientes tareas para generar la factura de divisas:

1. Pasar el archivo **FAC_DIVISA.mdf** a Usuario y Producción
2. Generar un nuevo Job denominado **FACTURA_TRANSF**
3. Colocar el archivo **FAC_DIVISA.mdf** en la subcarpeta **… /FORMAS/DIVISAS/**
   si-spool/adobe/central/server/FORMAS/FAC_DIVISA.mdf
4. Configurar el archivo **conf.jmd**

Solicité al área de AdminUnix realizar las siguientes tareas para generar la nota de venta:

1. Pasar el archivo **NOTA_VEN.mdf** a Usuario y Producción
2. Generar un nuevo Job denominado **NOTA_VEN**
3. Colocar el archivo **NOTA_VEN.mdf** en la subcarpeta **… /FORMAS/DIVISAS/**
   si-spool/adobe/central/server/FORMAS/NOTA_VEN.mdf
4. Configurar el archivo **conf.jmd**

Compilación de negocio y batch

Para la compilación utilicé el mismo procedimiento que llevé a cabo en el apartado 3.1.

Pruebas

Una vez que configuré el archivo **config.xml**, busqué la clase StandAloneReportCreationTest.java del proyecto Pmodule, posteriormente le dí click derecho sobre la clase, seleccioné *run as -> Junit test*, luego *debug as -> Junit test*, con esto inicialicé la instancia del objeto requerido el cual puse en el archivo config.xml y mandé llamar al método correspondiente. La clase **StandAloneReportCreationTest.java** la utilicé para ejecutar la aplicación además de que coloqué *breakpoints* en las clases que probe.

Al solicitar las configuraciones al área de base de datos se presentaron errores por que no las realizaron correctamente y existieron incidencias donde los mensajes los presento enseguida:

[16-02-10 16:10:51] **ERROR** (subfolder.print.module.job.PrintReportTask- copyFromSpoolerToCentral: 413) -> No se ejecuto correctamente copyFromSpoolerToCentral -> ComandoShellReplace: replace1 /si-multif/estados_cta/00010/FACDIV_1745716.XML ComandoCopia: cp /si-multif/estados_cta/00010/FACDIV_1745716.XML /si-spool/adobe/central/server/dat/FACDIV_1745716.XML waitFor:1

Otro error fue no utilizar correctamente los tipos de datos como describo en las siguientes líneas:

[24-02-10 13:42:38] ERROR (subfolder.print.module.job.PrintReportTask- executeTask: 278) -> LoadReportTask Failure!:

java.lang.RuntimeException: java.lang.Double at mx.com.grupofinanciero.subfolder.rep.FacDivisa.printReport(FacDivisa.java:367)
at mx.com.grupofinanciero.subfolder.print.module.impl.AbstractPrinter.preprintReport(AbstractPrinter.java:79) at mx.com. grupofinanciero.subfolder.print.module.job.PrintReportTask.executeTask(PrintReportTask.java:264)

El error donde la impresora no estaba dada de alta en el servidor de AdminUnix lo presento en las siguientes líneas:

[25-02-10 12:21:09] ERROR (si.print.module.job.NotifyFinishedReportTask- executeTask: 67) -> LoadReportTask Failure!: mx.com.grupofinanciero.subfolder.print.module.dao.DaoException: Imposible obtener el id de la impresora PLAZA1

Cuando generé la nota de venta encontré un error al consultar la información ya que esta consulta buscaba en una vista que ya no se debía utilizar por que no estaba en el esquema de producción, para resolver este error busque en los archivos log de bitácora y encontré que el error se encontraba justamente al realizar la consulta a la base de datos por eso recomendé que se revisaran todos los objetos consultados para ver cuales tenían utilidad y cuales ya no. Otro problema que encontré fue que al tratar de generar la nota de venta no se realizó el *.pdf y tampoco se imprimió en la impresora, esto lo resolví revisando los archivos *.log bitácora de negocio y batch, además solicité al área de arquitectura debuguear el ambiente de usuario y encontré que la plantilla o modelo de diseño con extensión *.mdf no estaba colocado en la ruta correcta del servidor. Al momento de terminar el desarrollo de ventanilla de divisas continué con el desarrollo del módulo de operaciones intermediación

el cual fue uno de los más importantes para el SI, mi participación en él lo detallo en el siguiente apartado.

3.3 OPERACIONES INTERMEDIACIÓN

Para este módulo realicé el análisis de impacto, viabilidad y factibilidad ya que significó un cambio importante en el funcionamiento y estructura del código fuente de la ventanilla, el análisis lo realicé desde la capa de vista hasta la capa de datos, además revisé el funcionamiento y flujo de la información, los cambios a realizar, la persistencia, revisión de los catálogos y constantes. El equipo de trabajo se apegó al siguiente procedimiento:

1. Asigné a cada integrante del equipo un tipo de operación para que conociera el negocio.
2. Cada desarrollador se enfoco en conocer el flujo del negocio y tipo de operación que le asigné así como los objetos de la base de datos involucrados.
3. Cada desarrollador identificó cuáles y cuántos objetos se utilizaban en sesión relacionados con el tipo de operación que le asigné.
4. Identifiqué cuáles fueron los puntos de convergencia en donde los tipos de operación compartieron negocio, objetos, tablas, lógica, validaciones, pantallas, cargas, configuraciones en el archivo struts*.xml, asi como constantes, formularios, objetos persistentes y mapeos.
5. Cada desarrollador me informó de los cambios y avances realizados una vez por día.
6. Cuando los desarrolladores tuvieron dudas sobre el negocio del tipo de operación diferente al que les asigné me podían preguntar si tenían dudas en cualquier momento.
7. Diariamente cada desarrollador realizó un reporte del avance de las asignaciones para llevar un seguimiento del desempeño y efectividad para conocer el estatus del desarrollo.
8. Se realizaron juntas semanales para verificar avances y pendientes del desarrollo, en ocaciones requerí realizar juntas extraordinarias cuando se presentaron inconvenientes como desconocimiento del negocio en partes de alto impacto.
9. Los desarrolladores podían realizar sugerencias y se analizaban.
10. El equipo se debía apoyar mutuamente unos a otros.

Estos diez pasos los propuse en una junta inicial para definir el camino a seguir tanto en conocimientos de negocio y técnicos del lenguaje del desarrollo, también propuse reutilizar las cargas de las pantallas, que sirvieron para presentar los tipos de productos solo con la diferencia que unas operaciones se realizaron en línea y otras fueron programadas. Para no afectar el rendimiento y la seguridad del sistema utilicé el mínimo de objetos de sesión ya que la seguridad del sistema fue una prioridad. En la figura 3.14 presento el diagrama de casos de uso y en la figura 3.15 los diagramas de componentes de las operaciones intermediación.

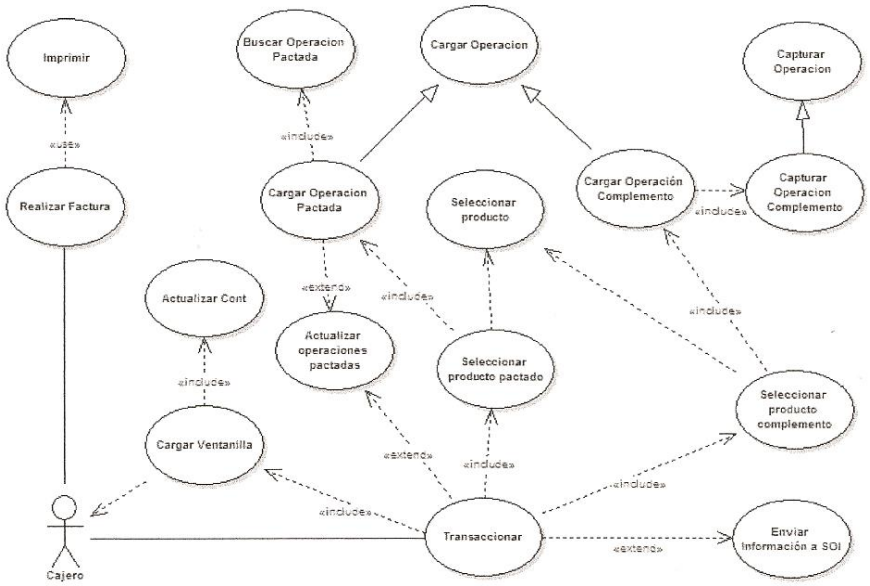


Figura 3.14: Diagrama de casos de uso operaciones intermediación (Sistemas Administrativos GF, Eric López, 2009)

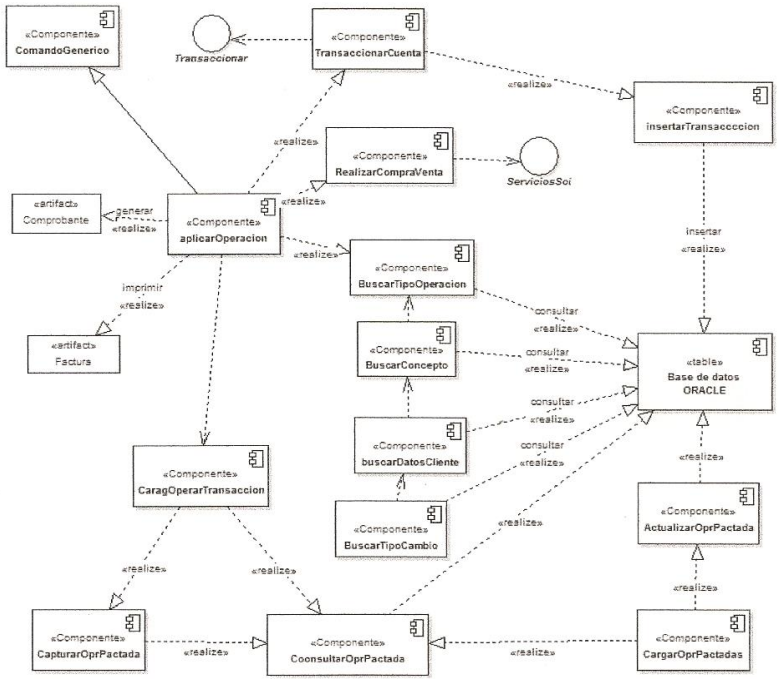


Figura 3.15: Diagrama de componentes operaciones intermediación (Sistemas Administrativos GF, Eric López, 2009)

En la figura 3.16 y 3.17 presento los diagramas de secuencia de operaciones intermediación.

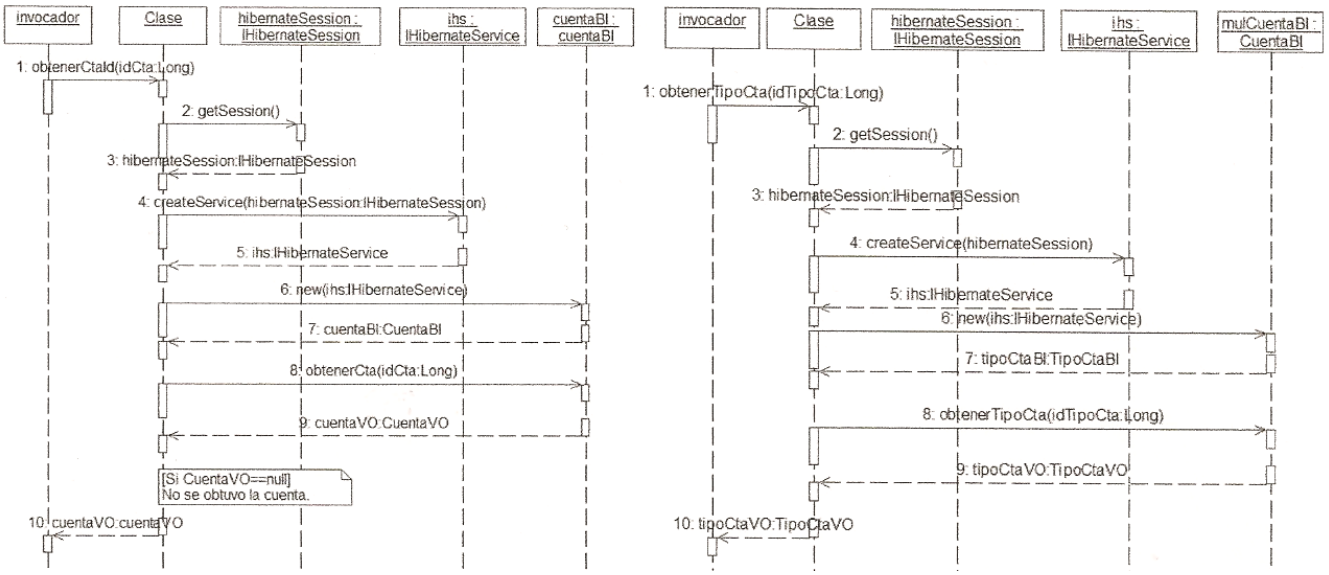


Figura 3.16: Diagrama de secuencia operaciones intermediación (Sistemas Administrativos GF, Eric López, 2009)

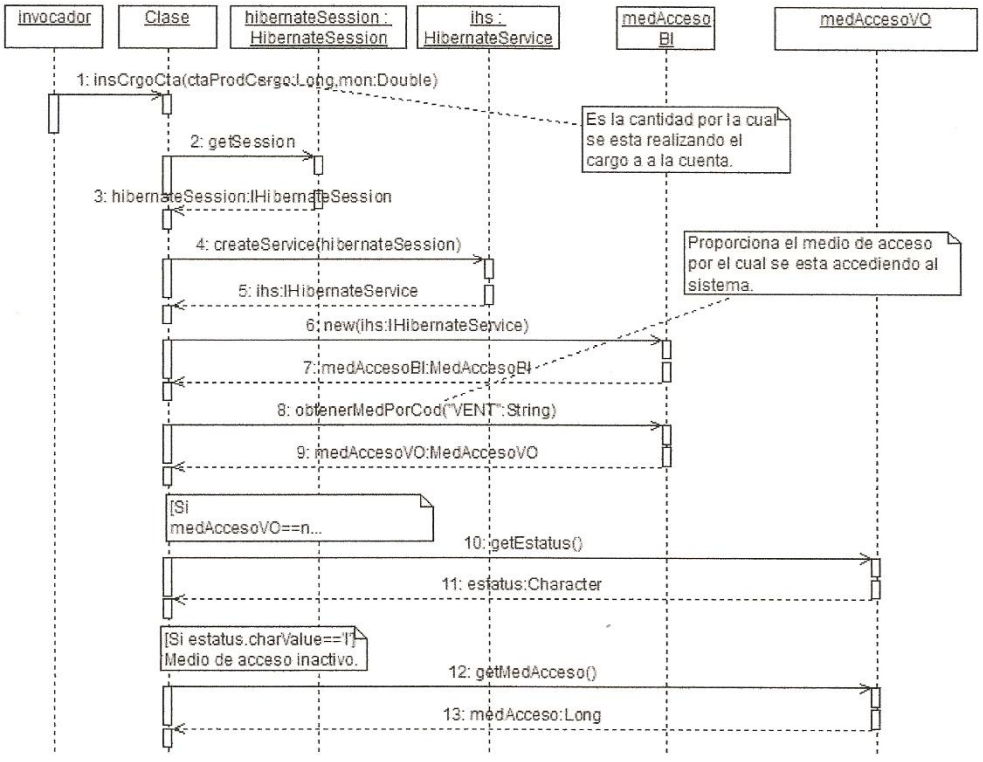


Figura 3.17: Diagrama de secuencia operaciones intermediación (Sistemas Administrativos GF, Eric López, 2009)

Desarrollo

Para el desarrollo del código fuente de este módulo utilicé las especificaciones del apartado 3.1 para culplir con los estándares de codificación y nombrado.

Web Services

En la figura 3.18 presento el diccionario de datos que utilicé para desarrollar el código de conectividad del lado del banco ya que fui el consumidor de los *web services* ComprayVentWS, cancelaComprayVentWS y ServicioGiroWS, estos servicios se integraron con datos de entrada, de salida, códigos y mensajes de error.

INPUT

Descripción del atributo			Tipo de dato	Obligatorio/Opcional	Longitud mínima	Longitud máxima
<Suc> <//Suc>		Número de sucursal de la operación	string	Obligatorio	4	4
<Usu> <//Usu>		Clave de usuario que está operando la transacción	string	Obligatorio	4	20
<Cliente> <//Cliente>		Número de cliente que pacta la operación.	string	Obligatorio	10	10
<FechOp> <//FechOp>		Fecha de la operación en formato MMDDYYYY	string	Obligatorio	8	8
<Pzo> <//Pzo>		Número de días para la liquidación	short	Obligatorio	1	1
<MoRec> <//MoRec>		Importe en la divisa a recibir	decimal	Obligatorio	1	29
<MoEnt> <//MoEnt>		Importe en la divisa a entregar	decimal	Obligatorio	1	29
<TpoCamb> <//TpoCamb>		Tipo de cambio de la operación	double	Obligatorio	1	16
<TpoCamb2> <//TpoCamb2>		Existirá un tipo de cambio 2 cuando se trate de una operación cruzada	double	Opcional	1	16

OUTPUT

	Descripción del atributo	Tipo de dato
Procedimiento realizado correctamente		
<Autoriza> <//Autoriza>	Folio de autorización del sistema proveedor	int

	Descripción del atributo	Tipo de dato
Mensaje de error		
<CodError> <//CodError>	Código de error	string
<MsgError> <//MsgError>	Descripción del error presentado	string

Figura 3.18: Diccionario de datos de *web services* (Sistemas Especializados GF, María del Carmen, 2009)

Configuración y parametrización de los *Web Services*

En los archivos ***config\batch\application-configuration.*xml** y ***config\application-configuration.*xml** revisé que los desarrolladores configuraran correctamente las URL de la publicación de los *web services* que muestro en las siguientes líneas:

```
<!--        Configuracion para comunicación con sistema Proveedor    -->
  <context name="nombreDelContexto">
        <property name=" nombreDelServicioComprayVentWS ">
              <value>
                    http://@PROXY_WS_URL@/nombreDelServicioComprayVentWS/ComprayVent.asmx?wsdl
              </value>
        </property>
        <property name=" nombreDelServicioCancelaComprayVentWS ">
              <value>
              http://@PROXY_WS_URL@/nombreDelServicioCancelaComprayVentWS/cancelaComprayVent.asmx?wsdl
              </value>
```

```
        </property>
        <property name=" nombreDelServicioGiroWS ">
                <value>
                http://@PROXY_WS_URL@/nombreDelServicioGiroWS/servicioGiro.asmx?wsdl
                </value>
        </property>
</context>
  <!-- Termina Configuración para comunicación con sistema proveedor          -->
```

context name.- Nombre del contexto que asigné para identificar con qué sistema se conectó la aplicación del banco ya que no debía existir otro nombre de contexto igual a este en el mismo archivo.

property name.- Fue el nombre que le asigné a cada servicio para saber a cuál se estaba referenciando y la petición no se confundiera, no debía existir ningún otro nombre de propiedad igual en el archivo.

***value*.-** Fue la URL completa donde se publicaron los *web services* del sistema proveedor.

@PROXY_WS_URL@.- Representó la IP donde se publicó cada *web service* del sistema proveedor.

Encontré problemas con nombres repetidos de los contextos, por lo tanto estas configuraciones las revisé contexto por contexto, una vez corregidos, revisados y confirmados los envié al área de arquitectura quienes se encargaron de montar la aplicación en ambiente desarrollo, usuario, QA, auditoría y producción, además se encargaron de cambiar el valor de la variable **@PROXY_WS_URL@** dentro del archivo **setenv.cmd** colocando la IP correspondiente a cada ambiente. Para complementar la parametrización correcta de esta variable propuse al área de arquitectura realizar algunos cambios físicos en el proyecto:

1. En el archivo ***build/*batch_build.properties** (binario con extensión *.bin) que se encontraba en el proyecto, coloqué la variable:

   proxy.ws.url=${env.**PROXY_WS_URL**}

2. En el archivo ***build/*batch_build.xml** que se encontraba en el proyecto, coloqué:

   ```
   <target name="prepare_config">
   <echo message="Generando la configuracion de la aplicacion ..."/>
   <echo message="${build.config.dir}"/>
   </target>
   <target name="prepare_config_ext">
   <echo message="Generando la configuracion externa de la aplicacion ..."/>
   <echo message="${build.configper.dir}"/>
   <filter token="PROXY_WS_URL" value="${proxy.ws.url}"/>
   <filter token="PROXY_WS_URL" value="${proxy.ws.url}"/>
   </target>
   ```

3. En el archivo ***build/*build.properties** (binario con extensión *.bin) que se encontraba en el proyecto, coloqué la siguiente variable:

   proxy.ws.url=${env.**PROXY_WS_URL**}

4. En el archivo ***build/*build.sh** (de ejecución Shell) coloqué la siguiente línea, aunque este valor IP cambiaría según el parámetro:

   export **PROXY_WS_URL= IPdondeEstabaPublicado**

5. En el archivo ***build/build.*xml** que se encontraba en el proyecto, agregué las siguientes líneas:

```
<target name="prepare_config" description="Genera la configuracion del servidor como jar dentro del ear">
<echo message="Generando la configuracion de la aplicacion ..."/>
<filter token="PROXY_WS_URL" value="${env.PROXY_WS_URL}"/>

</target>

<target name="prepare_config" description="Genera la configuracion del servidor como jar dentro del ear">
<echo message="Generando la configuracion de la aplicacion ..."/>
<filter token="PROXY_WS_URL" value="${proxy.ws.url}"/>
</target>
<target name="prepare_config_ext" description="Genera la configuracion del servidor como jar dentro del ear" >
<echo message="Generando la configuracion externa de la aplicacion ..."/>
<filter token="PROXY_WS_URL" value="${proxy.ws.url}"/>
</target>
```

6. En el archivo ***build/*mul_dev_*build.*xml** que se encontraba en el proyecto, agregué las siguientes líneas:

```
<target name="config" description="Genera la configuracion del servidor como jar dentro del ear">
<echo message="Generando la configuracion de la aplicacion ..."/>
<filter token="PROXY_WS_URL" value="${env.PROXY_WS_URL}"/>
<filter token="PROXY_WS_URL" value="${env.PROXY_WS_URL}"/>
<filter token="PROXY_WS_URL" value="${proxy.ws.url}"/>
</target>
```

Desarrollo de clientes *.Jar para comunicar los *Web Services*

Para la generación de las clases cliente del *Web Service* que me sirvieron para realizar la comunicación entre el sistema consumidor SI y el sistema proveedor SOI quien publicó el servicio al momento de ser terminado, requirí de un archivo *.wsdl (se debía conocer la URL donde se publicó el servicio), para esto les expliqué a los desarrolladores como realizarlo y así pudieron crear en el directorio *build* del proyecto un archivo *.xml con la descripción de la tarea que se ejecutó para generar un archivo *.jar que integró todas las clases *.class para enviar y recibir información, estas clases fueron las de comunicación *Stub.class, de protocolo *Soap.class, de lenguaje web services *.wsdl, la de serialización y deserialización *Codec.class, este xml lo llamé **ws_*client_build.*xml** y lo detallo en las siguientes líneas:

```
<project name="buildWeb Service" default="build-client">

  <!--   No olvidar ejecutar el ws_setenv.cmd   -->
<target name="build-client">
<clientgen     wsdl="http://IPdondeEstabaPublicado/ComprayVentWS/ComprayVent.asmx?WSDL"
packageName="mx.com.grupofinanciero.subfolder.integracion.ComprayVent.sistemaproveedor"
              clientJar="../lib/ws_proveedor /nombreDelClienteDeComprayVent.jar" />
```

</**target**>
</**project**>

El significado de cada etiqueta fue:

1. **project name.** Nombre que le dí al proyecto de compilación.
2. **Default.** Tarea que se ejecuté primero.
3. **target name.** Nombre de la tarea que ejecuté para generar el archivo *.jar.
4. **wsdl.** Dirección en la que se encontraba publicado el archivo *.wsdl del *web service*.
5. **packageName.** Nombre del paquete en donde generé las interfaces y las implementaciones de los métodos del w*eb service*, métodos de conexión con el sistema a que se hacía referencia.
6. **clientJar.** Fue la ruta en donde generé el archivo *.jar.

Después en el menu de eclipse, en Project desactivé la opción ***building Automatically***, modifiqué el archivo **ws_*setenv.*cmd** que se encontraba en el directorio ***build*** del proyecto de trabajo para colocar el valor correcto donde estaba declarado BEA_HOME, abrí una consola con la ruta del directorio ***build*** del proyecto, posteriormente ejecuté el archivo ws_setenv.cmd el cual contenía la tarea ***ant –f ws_client_build.*xml *build-client*** y genere el cliente nombreDelCliente.jar.

En el árbol de carpetas del proyecto dí clic derecho sobre la carpeta **lib** seleccionando la opción "**Refresh**" para actualizar la carpeta **ws_proveedor** que es donde especifiqué se crearan los archivos nombreDelClienteDeComprayVent.*jar*, nombreDelClienteDeC*ancelaComprayVent.jar* y nombreDelClienteDe*Giro.jar,* posteriormente en el archivo ***.classpath*** agregué la configuración de los *web services* como lo presento enseguida:

<classpathentry kind=*"lib"* path=*"lib/* ws_proveedor/ nombreDelClienteDeComprayVent.*jar"*/>
<classpathentry kind=*"lib"* path=*"lib/* ws_proveedor/ nombreDelClienteDeC*ancelaComprayVent.jar"*/>
<classpathentry kind=*"lib"* path=*"lib/* ws_proveedor/ nombreDelClienteDe*Giro.jar"*/>

En el menú de eclipse, activé la opción ***building Automatically*** para que eclipse realizara la compilación interna del proyecto y en ese momento ya aparecieron los clientes *.jar en las librerías referenciadas. Un inconveniente con el que me encontré fue la parametrización de los *web services*, para resolver este problema les sugerí a los responsables creadores de los servicios que en la parte de ***targetNamespace,*** **xmlns:s0** y dentro de ***definitions*** **xmlns** colocaran el **nombredelbanco** y de esta forma al generar las versiones no importando el ambiente el valor de **IPdondeEstabaPublicado** no influyó en la parametrización es decir éste sí podía tener la IP donde estaba publicado en servicio, esta parametrización la muestro en las siguientes líneas:

<?xml version="1.0" encoding="utf-8" ?>

- <**definitions xmlns**:http="http://schemas.xmlsoap.org/wsdl/http/"xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.**nombredelbanco**.com/sistemapriveedor/webservices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"**targetNamespace**="http://www.**nombredelbanco**.com/sistemaprivee dor /webservices/" xmlns="http://schemas.xmlsoap.org/wsdl/">

- <types>

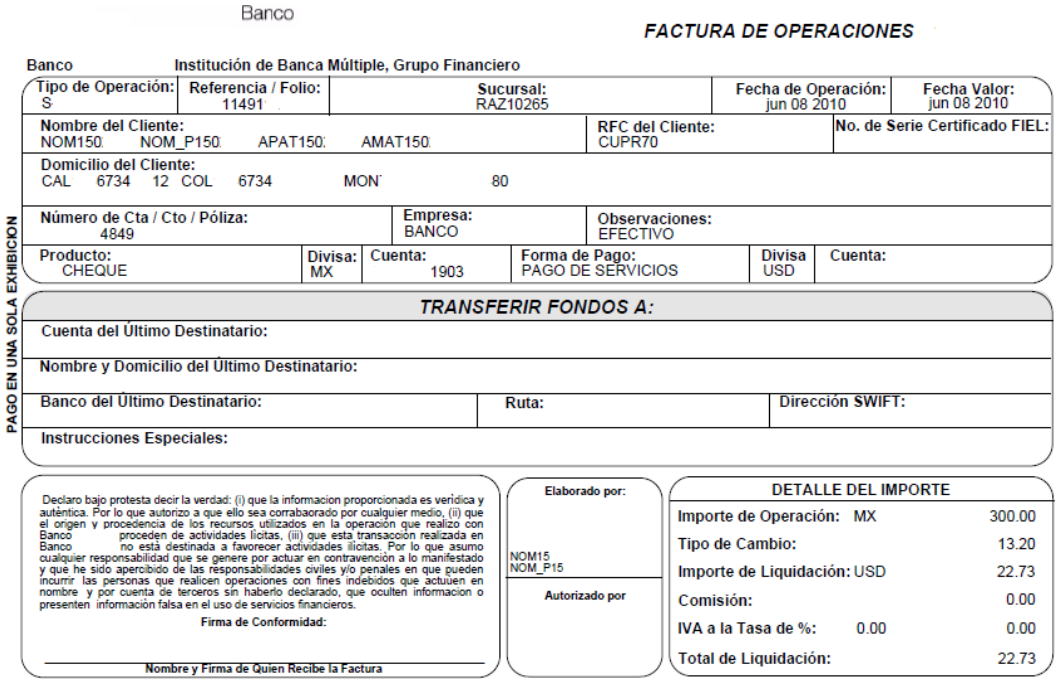
```
-<s:schema elementFormDefault="qualified"
        targetNamespace="http://www.nombredelbanco.com/sistemapriveedor/webservices/">

- <s:element name="ComprayVent">
            - <s:complexType>
            - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="EntradaVO" type="s0:ComprayVentEntrada" />
        </s:sequence>
        </s:complexType>
                            </s:element>
                             - <s:complexType name="ComprayVentEntrada">
        - <s:sequence>
             <s:element minOccurs="0" maxOccurs="1" name="Sucursal" type="s:string" />
                              <s:element minOccurs="0" maxOccurs="1" name="Usuario" type="s:string" />
                    <s:element minOccurs="0" maxOccurs="1" name="FechaOper" type="s:string" />
                      - <service name="ComprayVentWS">
                      - <port name="ComprayVentWSSoap" binding="s0:ComprayVentWSSoap">
        <soap:address location="http://IPdondeEstabaPublicado/ComprayVentWS/ComprayVentWS.asmx" />

        </port>
        </service>
        </definitions>
```

Compilación de negocio y batch

Para la compilación de negocio y batch utilicé el mismo procedimiento del apartado 3.1, una vez terminada la compilación procedí a generar el reporte de operaciones con divisas realizadas en ventanilla, en la figura 3.19 presento el resultado final de la factura, para ver más, consultar el apéndice B de Resultados de Comprobantes y Reportes.



Banco

FACTURA DE OPERACIONES

Banco Institución de Banca Múltiple, Grupo Financiero

Tipo de Operación:	Referencia / Folio:	Sucursal:	Fecha de Operación:	Fecha Valor:
S	11491	RAZ10265	jun 08 2010	jun 08 2010

Nombre del Cliente:	RFC del Cliente:	No. de Serie Certificado FIEL:
NOM150 NOM_P150 APAT150 AMAT150	CUPR70	

Domicilio del Cliente:
CAL 6734 12 COL 6734 MON 80

Número de Cta / Cto / Póliza:	Empresa:	Observaciones:
4849	BANCO	EFECTIVO

Producto:	Divisa:	Cuenta:	Forma de Pago:	Divisa	Cuenta:
CHEQUE	MX	1903	PAGO DE SERVICIOS	USD	

PAGO EN UNA SOLA EXHIBICION

TRANSFERIR FONDOS A:

Cuenta del Último Destinatario:

Nombre y Domicilio del Último Destinatario:

Banco del Último Destinatario:	Ruta:	Dirección SWIFT:

Instrucciones Especiales:

Declaro bajo protesta decir la verdad: (i) que la informacion proporcionada es verídica y auténtica. Por lo que autorizo a que ello sea corrabaorado por cualquier medio, (ii) que el origen y procedencia de los recursos utilizados en la operación que realizo con Banco proceden de actividades lícitas, (iii) que esta transacción realizada en Banco no está destinada a favorecer actividades ilícitas. Por lo que asumo cualquier responsabilidad que se genere por actuar en contravención a lo manifestado y que he sido apercibido de las responsabilidades civiles y/o penales en que pueden incurrir las personas que realicen operaciones con fines indebidos que actuúen en nombre y por cuenta de terceros sin haberlo declarado, que oculten informacion o presenten información falsa en el uso de servicios financieros.

Firma de Conformidad:

Nombre y Firma de Quien Recibe la Factura

Elaborado por:

NOM15
NOM_P15

Autorizado por

DETALLE DEL IMPORTE

Importe de Operación:	MX	300.00
Tipo de Cambio:		13.20
Importe de Liquidación:	USD	22.73
Comisión:		0.00
IVA a la Tasa de %:	0.00	0.00
Total de Liquidación:		22.73

Figura 3.19: Comprobante Factura Divisas (Sistemas Administrativos GF, Eric López, 2010)

Pruebas

Para las pruebas de la factura de divisas apliqué una matriz cuyo contenido incluyó diferentes escenarios con las divisas internacionales, conforme realizaba las pruebas fui aplicando cambios a los comprobantes ya que inicialmente no contaban con la información completa, los fui apegando a los reglamentos fiscales establecidos por la SHCP y CNBV, los resultados de los reportes inicialmente eran incompletos por lo que solicité junto con el área de sucursales que el área de diseño modificara los archivos modelo *.mdf, como responsable del desarrollo tuve que realizar cambios en el código para reestructurar las consultas SQL y obtener la información de la base de datos, una vez realizados estos ajustes, la factura y los reportes quedaron completos como el área de sucursales los había solicitado.

Errores de parámetros

Este tipo de errores se me presentaron al momento de probar la comunicación en línea de los *web services*, por lo que solicité al área de sistemas especializados creadores de los servicios y quienes relizaron las modificaciones me informaran sobre los cambios aplicados a los servicios, esto me permitió saber en que parte tenía que modificar las peticiones del lado de banco, en las siguientes líneas presento el error:

codigoError=”013”
mensajeError=”Se presentó un error al registrar la información de entrada. “

Errores de redondeo y tipo de datos

Estos errores se me presentaron por que no definí correctamente el tipo de dato y redondeo, para solucionarlo la gerencia de sistemas especializados y yo realizamos varias juntas para revisar la unificación de los tipos y longitudes en cada uno de los parámetros de entra y salida de los *web services*, también revisé que no impactaran a los campos persistidos a base de datos, el error lo presento en las siguientes líneas:

System.SystemException: Implicit conversion from datatype 'VARCHAR' to 'INT' is not allowed. Use the CONVERT function to run this query. | Sentencia: INSERT INTO CH_Inven (Ord_Id, CH_S, CH_NumIni, CH_Can, CH_Den)VALUES(3170, 'G', '4359',1,10)

Errores de base de datos

Cuando quise eliminar operaciones vía *web service*, al enviar los parámetros de entrada, me encontré errores por que en las consultas tomé encuenta campos que aún no se había dado de alta en la base de datos, esto lo resolví solicitando al área de base de datos dieran de alta correctamente los objetos, este error lo describo en la siguiente línea:

Invalid column name 'OperI'. | Sentencia: DELETE Opr_Canc WHERE OperI = '3170640'

Errores de NullReference

Al enviar una petición vía *web service* para cancelar una transacción la instancia no encontraba el identificador único, esto porque del lado del servidor la consulta estaba incorrecta, para resolver esto

solicité a los creadores de los servicios web modificaran su consulta y que me avisaran de sus cambios, el error lo muestro en las siguientes líneas:

codigoError=”-21”
mensajeError=”System.NullReferenceException: Object reference not set to an instance of an object.\n

Una vez finalizado el desarrollo de este módulo en el siguiente capítulo cuatro explico la evaluación al sistema SI, las desiciones estratégicas que tomó la organización para que el proyecto cumpliera los requerimientos establecidos así como el proceso que seguí y las herramientas que utilicé.

CAPÍTULO 4

EVALUACIÓN Y DESICIONES ESTRATEGICAS

En este cápítulo detallaré la evaluación del sistema el cual lo realice mediante matrices de prueba que arrojaron resultados medibles, calidad del tipo de errores ya sea bloqueantes y no bloqueantes, además de tipo visual y de negocio, para llevar a cabo esta métricas de desempeñó y terminado el desarrollo del sistema utilicé herramientas como Jmeter que también detallaré en este capítulo, además explicaré los modelos y diciplinas de software que implemento el GF como estrategia. A continuación comenzaré explicando la evaluación del sistema.

4.1 EVALUACIÓN DEL SISTEMA

Se evaluaron las necesidades de la actualización del sistema basándose en el desempeño, se analizaron actividades previas de soporte y mantenimiento, se determinó la satisfacción del usuario, en la bitácora del control de cambios se examinaron las modificaciones requeridas, se evaluó la necesidad de recodificar y documentar el registro de control de cambios, se resumieron los problemas así como las deficiencias del sistema y la efectividad del equipo de mantenimiento, los documentos generados fueron BI-QA (Monitoreo de desempeño del sistema) y BI-RPM.

Aplicación de JMeter

Esta herramienta me permitió simular la operación del sistema como si el usuario lo estuviera operando en tiempo real aportando resultados, con base en estos resultados realicé una evaluación para medir el porcentaje y tipos de error que se presentaron, esto me ayudo a evaluar los *web services* y peticiones HTTP, esta evaluación la presenté al gerente para que escalara los resultados a la dirección de sistemas y que ellos tomaran las desiciones finales. Para estas evaluaciones generé *scripts*, los ejecuté y levanté los ambientes locales, en el caso de ventanilla el ambiente estaba localmente en la máquina del desarrollador, para el portal existió una máquina especial, las partes importantes de los *scripts* las menciono acontinuación:

a) ***Test Plan.*** Controlador principal que definí al inicio del *script*, le asigné el nombre de la prueba, agregué variables definidas por el usuario llamadas ***User Defined*** **Variables**, ***host*** y **puerto** con su respectivo ***Value*** o valor.

b) ***User Parameters.*** En este procesador especifiqué los nombres de los *web services* o nombres de variables y sus valores los definí en la columna ***User_1.*** La estructura de cadena XML que utilicé para ingresar los parámetros la muestro a continuación:

<![CDATA[<XMLRoot><InputData><Usuario>${Usuario}</UserID><Clave>${Clave}</Clave>***</InputData></XMLRoot>]]>***

Para cada *web service* lo único que cambió fue la cantidad de etiquetas, sus nombres y sus valores de las mismas, no había límite máximo de *web services* declarados y definidos, también definí variables globales tipo *string*, *integer*, *long* y *double*.

c) **JDBC** ***Conection Configuration.*** Me sirvió para configurar la conexión a la base de datos Oracle.

d) ***Threads Group.*** Este *thread* lo agregué en el *Test Plan*, fueron *Threads* o hilos de módulo.

e) ***Loop Controller.*** Dentro de cada controlador simple agregué los controladores lógicos llamados *Loops Controller*s.

f) ***JDBC Request.*** Estas peticiones de consulta a la base de datos me permitieron obtener los datos, con esta petición evalué la carga de consultas, estas peticiones las agregué dentro de los *Loop Controller*.

g) ***CSV Data Set Config.*** Este elemento lo integré con archivos *.txt que estuvieron en la ruta del archivo *.jmx, esto por características de la herramienta, el contenido del archivo *.txt lo llamé desde el *script* y lo separé por tabulador, el número de ciclos fue el mismo número de filas que se encontraban en el archivo *.txt.

h) ***Web Service (SOAP) Request.*** En la tabla 4.1 detallo la configuración para peticiones de los *web services*.

Tabla 4.1: Configuración de *web service* (SOAP) *Request* (Sistemas Administrativos GF, Eric López, 2009)

Name	Definí el **nombre** para identificar el *sampler* de WS
WSDL URL	Definí la **URL** de la publicación del los *web services*
Protocolo(default http)	Establecí el **http**
Server Name or IP	Coloqué ***${host}***
Port Number	Coloqué ***${puerto}***
Path:	Coloqué el ***path*** **exacto** donde se encontraba publicado el *web service*, pero **sin IP ni Puerto**
SOAPAction	Coloqué el **nombre del servicio**
Soap*/XML-RPC Data*	Agregué el cuerpo completo del esquema SOAP del *web service*: *<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">* *<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">* *<ns:verifyCu xmlns:ns="http://ws.accesoportal.portal.multif.core.web service.subfolder.grupofinanciero.com.mx">* *<mulVerifyCu InputStringRO xsi:type="xsd:string">${Verify}</mulVerifyInputStringRO>* *</ns:verifyCu >* *</SOAP-ENV:Body>* *</SOAP-ENV:Envelope>*
Read SOAP *Response*	Habilité **checkbox**
Memory Cache	Habilité **checkbox**

i) ***Post Processors.*** Este procesador lo agregué en peticiones SOAP para extraer valores de las variables que cambiaron.

j) ***View Results Tree*** **en peticiones SOAP.** Este listener me permitió ver el proceso, los resultados, las peticiones, los valores en tiempo de ejecución, iteraciones, contenidos de las peticiones SOAP, consultas SQL, cantidad de errores, excepciones y los mensajes de excepciones.

k) **HTTP** ***Request*** **de los** ***Actions Path.*** Este sampler lo utilicé para las peticiones HTTP de la capa de control y la capa de presentación para que de esta forma se cargaran las páginas jsp.

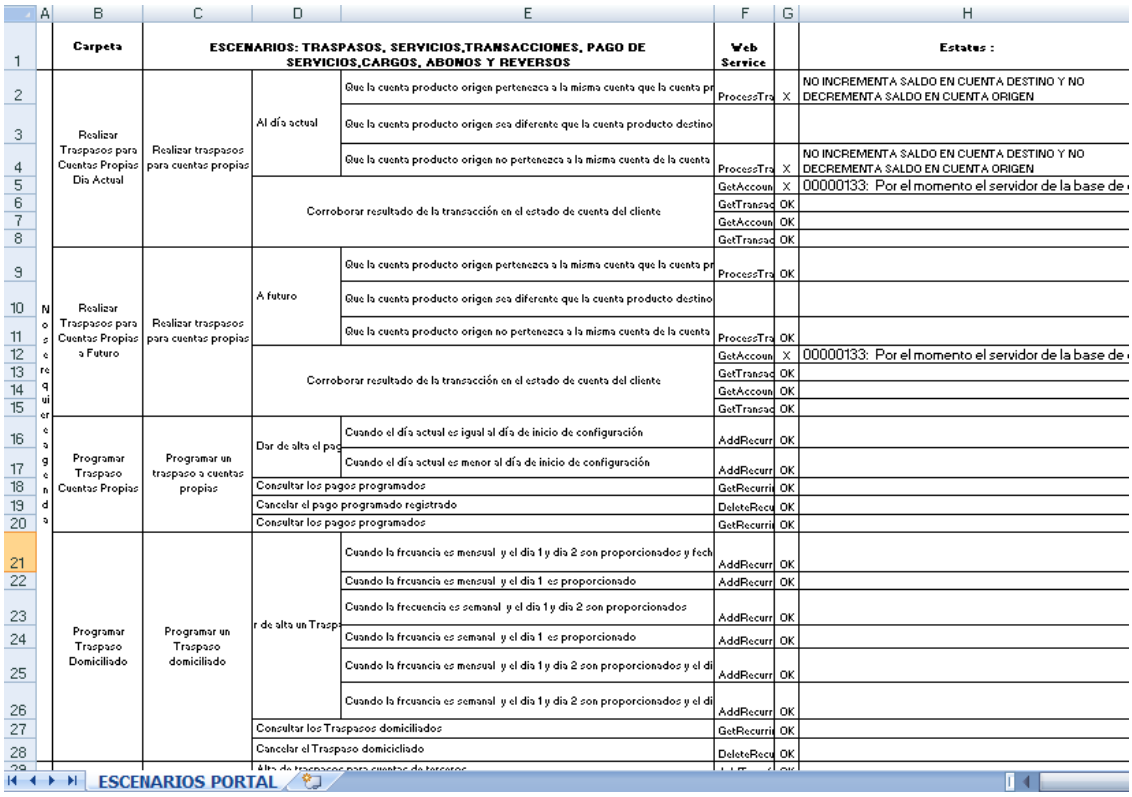
Resultados

Los resultados de los scripts los registré en los documentos **ESTATUS SCRIPTS VENTANILLA.xls** y **ESTATUS SCRIPTS PORTAL.xls** que presento en la figura 4.1 y 4.2 respectivamente los cuales me sirvieron para analizar, identificar y canalizar los errores e incidencias minimizando los tiempos de prueba sin necesidad de operar todo el sistema, posteriormente a la obtención de los resultados los escalé con el gerente y el a su vez lo presentó a la dirección de sistemas quienes evaluaron las posibilidades de liberar el proyecto en producción.

						ESCENARIO		PATH/ACTION	
CARPETA						Link(INCLUDE)		MOperarTransac	OK
				AGREGAR		CARGAR Operar Transaccion		MCargarOperarT	OK
					VIA COMBO	OPERAR Click Combo		MOperarTransac	OK
						CARGAR JSP CAPTURAR CARGO A Cta		MBuscarCta	OK
						BUSCAR	POR Cta	MBuscarCtaCrite	OK
							POR NOMBRE	MBuscarCtaCrite	OK
							POR RFC	MBuscarCtaCrite	OK
							POR CONTRATO	MBuscarCtaCrite	OK
							POR TARJETA	MBuscarCtaCrite	OK
						MOSTRAR		MCargarCaptura	OK
						AGREGAR Y MOSTRAR		MInsertarCapturs	OK
						REGRESAR(CARGAR Operar Transaccion)		MCargarOperarT	OK
					VIA CLICK EN CARGO A Cta	OPERAR Click CARGO A Cta		MOperarTransac	OK
						CARGAR JSP CAPTURAR CARGO A Cta		MCargarCaptura	OK
						BUSCAR	POR Cta	MBuscarCta	OK
								MBuscarCtaCrite	OK
							POR NOMBRE	MBuscarCta	OK
								MBuscarCtaCrite	OK
							POR RFC	MBuscarCta	OK
								MBuscarCtaCrite	OK
							POR CONTRATO	MBuscarCta	OK
								MBuscarCtaCrite	OK
							POR TARJETA	MBuscarCta	OK
								MBuscarCtaCrite	OK
						MOSTRAR		MCargarCaptura	OK
						AGREGAR Y MOSTRAR		MInsertarCapturs	OK
						REGRESAR(CARGAR Operar Transaccion)		MCargarOperarT	OK
				ACTUALIZAR	OPERAR CLICK COMBO	OPERAR Click Combo y dibujar capturar CARGO A Cta		MOperarTransac	OK
						CLICK EN ELINDICE DEL CARGO A MODIFICAR		MCargarActualiz	OK
						REALIZAR ACTUALIZACION Y MOSTRAR EN LA VENTANA CAPTURA DE CARGO A Cta		MActualizarCapt	OK
						REGRESAR(CARGAR Operar Transaccion)		MCargarOperarT	OK
					OPERAR CLICK CARGO Cta	OPERAR Click Combo y dibujar capturar CARGO A Cta		MOperarTransac	OK
						CLICK EN ELINDICE DEL CARGO A MODIFICAR		MCargarActualiz	OK
			CARGO A Cta			REALIZAR ACTUALIZACION Y MOSTRAR EN LA VENTANA CAPTURA DE CARGO A Cta		MActualizarCapt	OK
						REGRESAR(CARGAR Operar Transaccion)		MCargarOperarT	OK
						OPERAR Click Combo		MOperarTransac	OK
						CARGAR JSP CAPTURAR CARGO A Cta		MCargarCaptura	OK

ESCENARIO FINAL DE VENTANILLA

Figura 4.1: Resultados de Ventanilla (Sistemas Adminsitrativos GF, Eric López, 2010)



	Carpeta	ESCENARIOS: TRASPASOS, SERVICIOS,TRANSACCIONES, PAGO DE SERVICIOS,CARGOS, ABONOS Y REVERSOS			Web Service		Estatus :
No se requiere agenda	Realizar Traspasos para Cuentas Propias Dia Actual	Realizar traspasos para cuentas propias	Al día actual	Que la cuenta producto origen pertenezca a la misma cuenta que la cuenta pr	ProcessTra	X	NO INCREMENTA SALDO EN CUENTA DESTINO Y NO DECREMENTA SALDO EN CUENTA ORIGEN
				Que la cuenta producto origen sea diferente que la cuenta producto destino			
				Que la cuenta producto origen no pertenezca a la misma cuenta de la cuenta	ProcessTra	X	NO INCREMENTA SALDO EN CUENTA DESTINO Y NO DECREMENTA SALDO EN CUENTA ORIGEN
			Corroborar resultado de la transacción en el estado de cuenta del cliente		GetAccoun	X	00000133: Por el momento el servidor de la base de
					GetTransac	OK	
					GetAccoun	OK	
					GetTransac	OK	
	Realizar Traspasos para Cuentas Propias a Futuro	Realizar traspasos para cuentas propias	A futuro	Que la cuenta producto origen pertenezca a la misma cuenta que la cuenta pr	ProcessTra	OK	
				Que la cuenta producto origen sea diferente que la cuenta producto destino			
				Que la cuenta producto origen no pertenezca a la misma cuenta de la cuenta	ProcessTra	OK	
			Corroborar resultado de la transacción en el estado de cuenta del cliente		GetAccoun	X	00000133: Por el momento el servidor de la base de
					GetTransac	OK	
					GetAccoun	OK	
					GetTransac	OK	
	Programar Traspaso Cuentas Propias	Programar un traspaso a cuentas propias	Dar de alta el pag	Cuando el día actual es igual al día de inicio de configuración	AddRecurr	OK	
				Cuando el día actual es menor al día de inicio de configuración	AddRecurr	OK	
			Consultar los pagos programados		GetRecurri	OK	
			Cancelar el pago programado registrado		DeleteRecu	OK	
			Consultar los pagos programados		GetRecurri	OK	
	Programar Traspaso Domiciliado	Programar un Traspaso domiciliado	r de alta un Traspa	Cuando la frcuancia es mensual y el dia 1 y dia 2 son proporcionados y fech	AddRecurr	OK	
				Cuando la frcuancia es mensual y el dia 1 es proporcionado	AddRecurr	OK	
				Cuando la frecuencia es semanal y el dia 1 y dia 2 son proporcionados	AddRecurr	OK	
				Cuando la frcuancia es semanal y el dia 1 es proporcionado	AddRecurr	OK	
				Cuando la frcuancia es mensual y el dia 1 y dia 2 son proporcionados y el di	AddRecurr	OK	
				Cuando la frcuancia es semanal y el dia 1 y dia 2 son proporcionados y el di	AddRecurr	OK	
			Consultar los Traspasos domiciliados		GetRecurri	OK	
			Cancelar el Traspaso domiciliado		DeleteRecu	OK	
			Alta de traspasos para cuentas de terceros				

ESCENARIOS PORTAL

Figura 4.2: Resultados de Portal (Sistemas Administrativos GF, Eric López, 2010)

Otro de los aspectos importantes incluidos en el proyecto fueron las decisiones estratégicas, gracias a estas, el GF pudo lograr las metas establecidas y colocar a la organización en una posición de alta competitividad, estas decisiones las mencionaré en el siguiente apartado.

4.2 DESICIONES ESTRATÉGICAS

La dirección de sistemas del GF generó bases y condiciones que permitieron alcanzar el desarrollo de la institución ya que se comprometió en llevar a cabo el proyecto enfocado a mejorar los procesos de desarrollo de software elevando la calidad de los productos y servicios.

Para la obtención de resultados, lograr la visión y metas de la dirección de sistemas del GF se tomaron en cuenta aspectos importantes como la forma en que el personal realizaba su trabajo, se promovió el cambio de cultura hacia una medición del desempeño de los procesos y calidad de los productos y servicios que se entregaron a los clientes (Documento interno GF, Procesos, 2005). Se implemetaron las siguientes diciplinas:

1. **Administración de requerimientos**

   Se estableció entendimiento común entre el cliente y el equipo de desarrollo sobre los requerimientos del cliente que se cubrieron con el proyecto.

2. **Planeación del proyecto**

   Se establecieron planes razonables para ejecutar la ingeniería del software y administrar el proyecto.

3. **Seguimiento del proyecto**

   Se estableció visibilidad dentro del progreso vigente del proyecto.

4. **Administración de proveedores del software.**

   Se seleccionaron proveedores de software calificados y se administraton de forma efectiva.

5. **Aseguramiento de calidad**

   Se aseguró el cumplimiento en los requerimientos de las aplicaciones permitiendo a la administración un nivel adecuado de visibilidad en el proceso usado por los proyectos y en los productos construidos.

6. **Administración de la configuración del software**

   Se estableció y mantuvo la integridad de los productos desarrollados a través del ciclo de vida del sistema.

Los marcos normativos aplicados en el proyecto SI fueron:

a) **Seguridad Informática**

   El primer paso para garantizar la seguridad de la información fue identificar los riesgos y su impacto en el negocio ya que este fue uno de los activos más importantes para la organización por que se había visto que cada vez que se exponía a elevadas amenazas y vulnerabilidades de ahí que haya sido necesaria protegerla sin grandes inversiones en software, hardware o personal llevando a cabo una buena administración de seguridad

b) **Gobierno Corporativo**

   Este concepto estableció una estructura de relaciones y procesos para dirigir a la empresa hacia sus objetivos controlando el riesgo contra su entorno sobre la tecnología de información y sus procesos.

COMENTARIOS FINALES

Al redactar esta memoria, recapitulé la experiencia obtenida en el desarrollo del Sistema Integral del Grupo Financiero en el cual laboré del año 2007 al año 2010 esta experiencia vivida acentúa la realidad de que cada vez que se proponga la reingeniería de sistemas en una empresa o negocio, se tiene que tomar en cuenta si ya existe un sistema y cómo funciona, ventajas , desventajas al considerar aplicar una metodología y que esta sea clara para saber en todo momento, qué se tiene que hacer, cómo se tiene que hacer, quien lo tiene que hacer y los tiempos de entrega.

Algunos desarrolladores tenían desconocimiento del negocio y de la tecnología java, desconocimiento de los tiempos de entrega para los módulos a desarrollar ya que los requerimientos especificaban un tiempo de inicio y fin para realizar la entrega ya que estos últimos fueron planteados como objetivos, se tenía que realizar una planificación con base en los recursos humanos disponibles y con las habilidades necesarias para dar y evaluar el soporte a corto, mediano y largo plazo utilizando las herramientas propuestas que fueron transparentes para poder comunicarse entre plataformas aunque haya sido diferentes proveedores.

Existió otro punto clave que fue la metodología aplicada que me permitió administrar grupos de personas con orden y planear correctamente los proyectos, requerimientos y solución de incidencias, además de que la utilización de la arquitectura propuesta de n-capas que me garantizó una codificación controlada, clara, segura, fácil de entender, extensible, expansible, con capacidad de rediseño y reutilización.

En las organizaciones del personal como parte de la responsabilidad de las personas que las integran se debe explicar a la gente de nuevo ingreso el funcionamiento del negocio, apoyarlos, orientarlos, encaminarlos y explicarles las herramientas que se utilizan para desarrollar los proyectos de la empresa así como identificar el área de desarrollo donde una persona se desenvuelva mejor con base en sus aptitudes, tomar en cuenta estos aspectos me ayudó a planificar correctamente, realizar una distribución de asignaciones y tareas de cada persona del equipo de trabajo, para la creación de código se debe saber qué y como se tiene que realizar.

La herramienta JMeter fue muy importante con características de código abierto para realizar las pruebas necesarias al sistema y de esta forma poder evaluarlo y que la dirección de sistemas tomara las desiciones pertinentes.

Finalmente menciono que el objetivo de este trabajo se cumplió satisfactoriamente describiendo así mi participación en la implantación del sistema integral del banco del Grupo Financiero

APÉNDICE A

RECOMENDACIONES

En el actual apéndice mencionaré las recomendaciones aplicadas que me permitieron mejorar el rendimiento de la aplicación, los tiempos de respuesta conservando la integridad de los datos así como la aplicación de mayor seguridad al sistema, además de que las configuraciones y mapeos fueron de forma transparente evitando errores bloqueantes o inconsistencias de los datos consultados actualizados o eliminados, las recomendaciones las detallo enseguida:

1. **Modelo Entidad-Relación**

Un buen mapeo objeto-relacional me ayudó a tener de manera gráfica la relación de objetos, las recomendaciones las explico a continuación:

a) Realicé el mapeo hasta que el Modelo de Clases estuviera completo.
b) Por cada clase del modelo de clases generé una entidad en el diagrama entidad-relacion.

2. Multiplicidad del modelo de clases

Agregué los identificadores a cada tabla con correspondencia uno a uno con el modelo de clases para darle identidad al objeto, respeté la multiplicidad del modelo de clases. En las entidades hijo agregué la llave primaria del padre como una llave externa de la entidad.

3. Entidad asociativa sin id propio

Apliqué reglas de modelado entidad–relación ya que por cada relación muchos a muchos generé una entidad que rompió la relación y guardo las incidencias de la misma relacionando las entidades asociadas de uno a muchos contra la nueva entidad de relación.

4. Relación subtipo

Apliqué reglas de herencia en el modelo relacional por lo que la llave de la superclase la agregué a las clases derivadas como llave primaria pero externa (*Primary Foreign Key*).

5. Entidad asociativa con id propio

Cuando detecté relaciones muchos a muchos entre dos clases donde la relación generaba atributos propios en una nueva clase realicé el mapeó generando una entidad propia para la clase asociativa donde la llave primaria fue el id propio de la clase y las llaves primarias de cada clase de la asociación las agregué como llaves externas a la entidad asociativa.

6. Vocabulario Estandarizado recomendado para Operaciones de Negocio

Apliqué el vocabulario para estandarizar la clasificación de paquetes donde al momento del desarrollo agregué las las clases a los paquetes correctos, de esta manera el árbol del código lo homologué, en la tabla A1 presento los nombres y descripciones de este vocabulario.

Tabla A1. Nomenclatura recomendada para homologar código (Control de versiones GF, Zitziqui Gutierrez, 2007)

Nombre Operación Autorizada	**Descripción**	**Nombres Homologados**	**Ejemplo**
Registrar	Verbo empleado para nomenclatura de métodos que representaron la acción de crear o capturar nuevos registros en alguna entidad.	Insertar, Capturar, Alta Registro, Nuevo Registro, Agregar Registro, etc.	registrarSolicitud()
Eliminar	Verbo empleado para nomenclatura de métodos que representaron acciones de suprimir registros de alguna entidad.	Borrar, Baja Registro.	eliminarSolicitud()
Modificar	Verbo empleado para nomenclatura de métodos que representaron la acción de cambio de valores en registros de alguna entidad.	Actualizar, Cambiar Registro.	modificarSucursal()
Consultar	Verbo empleado para nomenclatura de métodos que representaron la obtención de un grupo de registros obtenidos a partir de alguna(s) entidad(es) basado en un determinado criterio de selección.	Desplegar, Extraer, Obtener, Mostrar.	consultarCuenta()
Imprimir	Verbo empleado para nomenclatura de métodos que representaron la acción de ejecutar rutinas de Impresión.	Impresión.	imprimirReporte()
Procesar	Verbo empleado para nomenclatura de métodos que representaron la acción de invocar la ejecución otros procesos (subprogramas, batchs, etc) u operaciones masivas que tuvieron como resultado la generación de nuevos registros, modificaciones, eliminaciones o el proceso de información para impresiones masivas.	Generar, Ejecutar, Disparar, Aplicar procesos masivos.	procesarAutomatico()
Asignar	Verbo empleado para nomenclatura de métodos que representaron la acción de relacionar, asignar o asociar objetos o atributos.	Relacionar, Asociar, Asignar.	asignarNivelEjecucion()
Calcular	Verbo empleado para nomenclatura de métodos que representaron la acción de efectuar operaciones que arrojaron resultados totalizados.	Totalizar, Afectar, ObtenerCalculo, etc.	calcularMonto()
Tarificar	Verbo empleado para nomenclatura de métodos que representaron la acción de calcular tarifas de un contrato en el negocio de banco.		tarificarIva()
Seleccionar	Verbo empleado para nomenclatura de métodos que representaron la acción de elegir una o más opciones de un conjunto posible de elecciones.	Marcar, Elegir, Depurar, Indicar.	seleccionarCobertura()
Validar	Verbo empleado para nomenclatura de métodos que representaron la acción de revisar si determinados datos cubrían con determinadas características para proseguir el flujo o detenerlo en caso contrario.	Verificar, Revisar Valores, AplicaCondiciones, etc.	validarContrato()
Emitir	Verbo empleado para nomenclatura de métodos que representaron la acción Impresión de comprobantes en el negocio de banco		emitirComprobante()
Bloquear	Verbo empleado para nomenclatura de métodos que representaron la acción bloquear el pago de comisiones o incentivos de los asesores.		bloquearPagos()
Liberar	Verbo empleado para nomenclatura de métodos que representaron la acción liberar el pago de comisiones o incentivos de los asesores que se encuentran bloqueados.		liberarPagos()

ERGONOMIA PARA EL DISEÑO DE LAS PANTALLAS

Introducción

Utilizando el concepto de ergonomía como ciencia del diseño y ajuste óptimo de las capacidades y limitaciones del cuerpo humano (Documento interno GF, Procesos, 2008) pude diseñar pantallas confortables para los usuarios quienes necesitaron estar cómodos con la forma de presentar información en pantallas, fue necesario reglamentar la forma en que la información se desplegó, la ergonomía en las pantallas tuvo los beneficios de aumentar la calidad visual y mejorar la distribución de la información en las interfaces evitando mezclar de forma incorrecta tipos de información que fuese poco clara al usuario.

Balance

El balance me permitió equilibrar de forma céntrica la información teniendo elementos con igual carga de izquierda a derecha y de arriba hacia abajo. En el diseño de páginas web el balance vertical o de izquierda a derecha es usualmente el concepto más importante dado que la mayoría de páginas tienen barras de desplazamiento, de tal forma que puede desplazar el centro de balance horizontal, o de arriba abajo, en la figura A1 presento este principio.

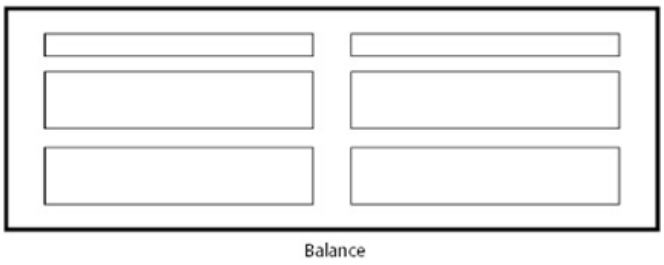


Figura A1: Balance (Créditos GF, Juvenal Guzman, 2008)

Simetría

Con la simetría determiné una unidad en un lado de la línea central y otra exactamente al otro lado de la interfaz donde esta replicación permitió un balance formal, con la diferencia que con el balance pude obtenerlo sin aplicar simetría, en la figura A2 ilustro este principio.

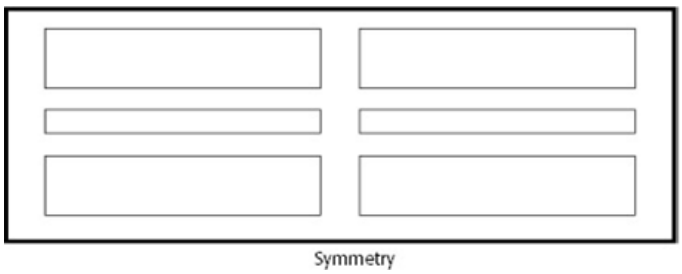


Figura A2. Simetría (Créditos GF, Juvenal Guzman, 2008)

Regularidad

Este principio lo apliqué en el diseño de las pantallas consiguiendo estándares y puntos de partida para columnas y filas espaciadas consistentemente, de esta forma obtuve regularidad utilizando elementos de tamaño, forma, color y espacio similar.

Predictibilidad

Este principio me sugirió orden convencional ya que el usuario al ver una pantalla podía predecir como era la siguiente y al ver una parte de la pantalla se podía predecir el resto de la misma.

Secuencialidad

Este principio fue un plan de presentación que guiaba los ojos a través de la pantalla en un orden lógico y rítmico ubicando la información más importante en un lugar significativo. A su vez la secuencialidad la obtuve aplicando alineaciones, espaciado, agrupamiento e ilustraciones.

Economía

Con este principio apliqué un uso austero y prudente de elemento de despliegue lo más simple posible, en el diseño de las pantallas llevé únicamente los mensajes deseados y no más, el uso de colores violaba este principio, en la figura A3 ilustro este principio.

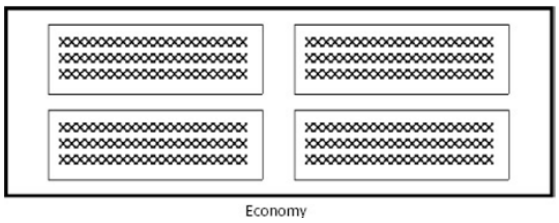


Figura A3: Economía (Créditos GF, Juvenal Guzman, 2008)

Unidad

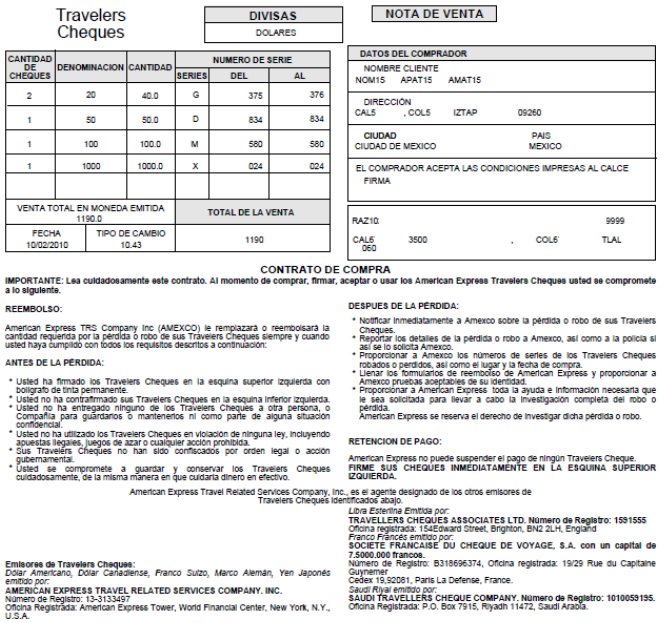
La unidad es coherencia entre una totalidad de elementos que son visualmente una sola pieza. Con unidad, los elementos parecían que que pertenecían a un conjunto dando así la impresión de que fueran una misma cosa donde cada pieza tienía sus propias características. En el diseño de pantallas los tamaños, formas y colores similares promovieron la unidad, así como bordes en blanco para limitar los despliegues.

Simplicidad

Con la simplicidad combiné elementos que fueron fácil de entender basándome en un patrón de lo que pretendí proyectar al usuario.

APÉNDICE B

RESULTADO DE COMPROBANTES Y REPORTES



Travelers Cheques

DIVISAS
DOLARES

NOTA DE VENTA

CANTIDAD DE CHEQUES	DENOMINACION	CANTIDAD	NUMERO DE SERIE		
			SERIES	DEL	AL
2	20	40.0	G	375	376
1	50	50.0	D	834	834
1	100	100.0	M	580	580
1	1000	1000.0	X	024	024
VENTA TOTAL EN MONEDA EMITIDA 1190.0			TOTAL DE LA VENTA		
FECHA 10/02/2010	TIPO DE CAMBIO 10.43		1190		

DATOS DEL COMPRADOR

NOMBRE CLIENTE
NOM15 APAT15 AMAT15

DIRECCION
CAL5 , COL5 IZTAP 09260

CIUDAD: CIUDAD DE MEXICO
PAIS: MEXICO

EL COMPRADOR ACEPTA LAS CONDICIONES IMPRESAS AL CALCE
FIRMA

RAZ10 9999

CAL6 3500 , COL6 TLAL
060

CONTRATO DE COMPRA

IMPORTANTE: Lea cuidadosamente este contrato. Al momento de comprar, firmar, aceptar o usar los American Express Travelers Cheques usted se compromete a lo siguiente.

REEMBOLSO:

American Express TRS Company Inc (AMEXCO) le remplazara o reembolsara la cantidad requerida por la pérdida o robo de sus Travelers Cheques siempre y cuando usted haya cumplido con todos los requisitos descritos a continuación:

ANTES DE LA PÉRDIDA:

* Usted ha firmado los Travelers Cheques en la esquina superior izquierda con bolígrafo de tinta permanente.
* Usted no ha contrafirmado sus Travelers Cheques en la esquina inferior izquierda.
* Usted no ha entregado ninguno de los Travelers Cheques a otra persona, o Compañía para guardarlos o mantenerlos ni como parte de alguna situación confidencial.
* Usted no ha utilizado los Travelers Cheques en violación de ninguna ley, incluyendo apuestas ilegales, juegos de azar o cualquier acción prohibida.
* Sus Travelers Cheques no han sido confiscados por orden legal o acción gubernamental.
* Usted se compromete a guardar y conservar los Travelers Cheques cuidadosamente, de la misma manera en que cuidaría dinero en efectivo.

DESPUES DE LA PÉRDIDA:

* Notificar inmediatamente a Amexco sobre la pérdida o robo de sus Travelers Cheques.
* Reportar los detalles de la pérdida o robo a Amexco, así como a la policía si así se lo solicita Amexco.
* Proporcionar a Amexco los números de series de los Travelers Cheques robados o perdidos, así como el lugar y la fecha de compra.
* Llenar los formularios de reembolso de American Express y proporcionar a Amexco pruebas aceptables de su identidad.
* Proporcionar a American Express toda la ayuda e información necesaria que le sea solicitada para llevar a cabo la investigación completa del robo o pérdida.

American Express se reserva el derecho de investigar dicha pérdida o robo.

RETENCION DE PAGO:

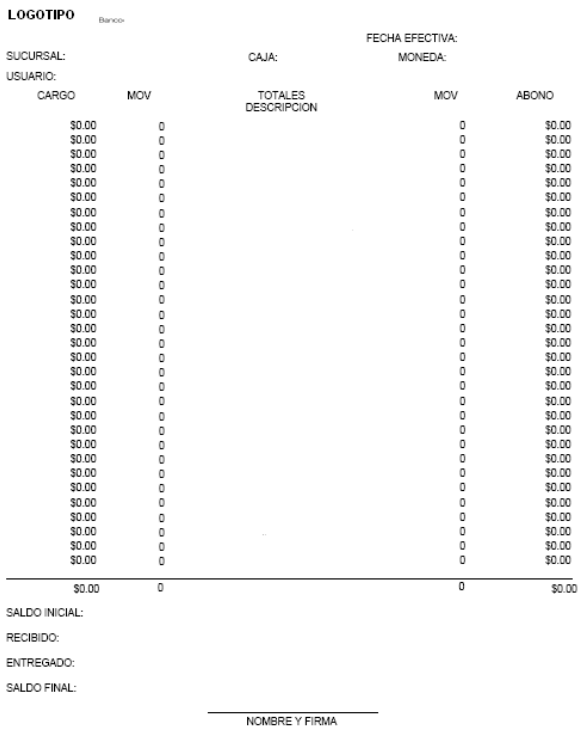
American Express no puede suspender el pago de ningún Travelers Cheque.
FIRME SUS CHEQUES INMEDIATAMENTE EN LA ESQUINA SUPERIOR IZQUIERDA.

American Express Travel Related Services Company, Inc., es el agente designado de los otros emisores de Travelers Cheques identificados abajo.

Emisores de Travelers Cheques:
Dólar Americano, Dólar Canadiense, Franco Suizo, Marco Alemán, Yen Japonés emitido por:
AMERICAN EXPRESS TRAVEL RELATED SERVICES COMPANY. INC.
Número de Registro: 13-3133497
Oficina Registrada: American Express Tower, World Financial Center, New York, N.Y., U.S.A.

Libra Esterlina Emitida por:
TRAVELLERS CHEQUES ASSOCIATES LTD. Número de Registro: 1591555
Oficina registrada: 15/Edward Street, Brighton, BN2 2LH, England
Franco Francés emitido por:
SOCIETE FRANCAISE DU CHEQUE DE VOYAGE, S.A. con un capital de 7.5000.000 francos.
Número de Registro: B318696374, Oficina registrada: 19/29 Rue du Capitaine Guynemer
Cedex 19,92081, Paris La Defense, France.
Saudi Riyal emitido por:
SAUDI TRAVELLERS CHEQUE COMPANY. Número de Registro: 1010059135.
Oficina Registrada: P.O. Box 7915, Riyadh 11472, Saudi Arabia.

Figura B1. Comprobante Nota de Venta (Sistemas Administrativos GF, Eric López, 2009)



LOGOTIPO Banco

FECHA EFECTIVA:

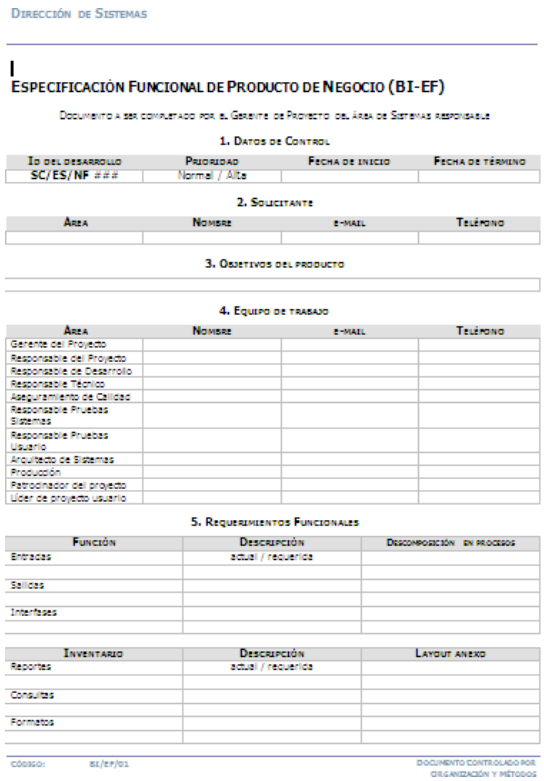
SUCURSAL: CAJA: MONEDA:

USUARIO:

CARGO	MOV	TOTALES DESCRIPCION	MOV	ABONO
$0.00	0		0	$0.00
$0.00	0		0	$0.00
$0.00	0		0	$0.00

APÉNDICE C

FORMATOS DEL MATERIAL DE LA METODOLOGÍA



DIRECCIÓN DE SISTEMAS

ESPECIFICACIÓN FUNCIONAL DE PRODUCTO DE NEGOCIO (BI-EF)

DOCUMENTO A SER COMPLETADO POR EL GERENTE DE PROYECTO DEL ÁREA DE SISTEMAS RESPONSABLE

1. DATOS DE CONTROL

ID DEL DESARROLLO	PRIORIDAD	FECHA DE INICIO	FECHA DE TÉRMINO
SC/ES/NF ###	Normal / Alta		

2. SOLICITANTE

ÁREA	NOMBRE	E-MAIL	TELÉFONO

3. OBJETIVOS DEL PRODUCTO

4. EQUIPO DE TRABAJO

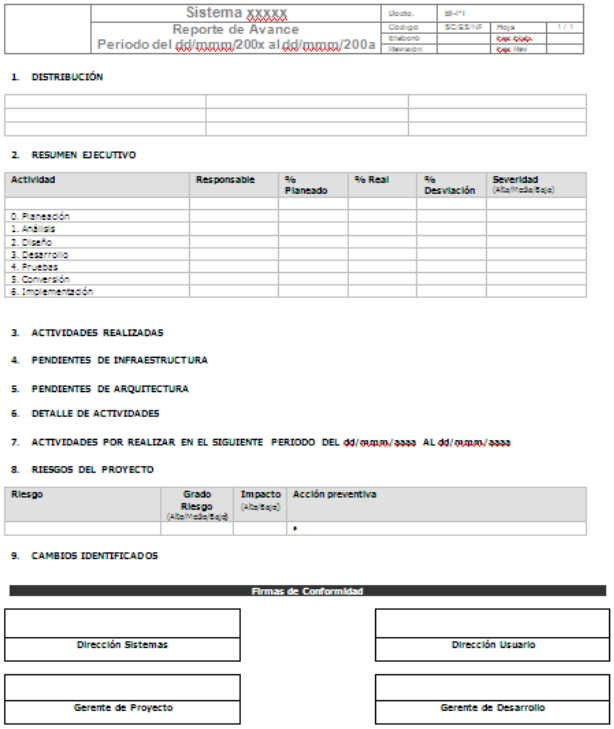
ÁREA	NOMBRE	E-MAIL	TELÉFONO
Gerente del Proyecto			
Responsable del Proyecto			
Responsable de Desarrollo			
Responsable Técnico			
Aseguramiento de Calidad			
Responsable Pruebas Sistemas			
Responsable Pruebas Usuario			
Arquitecto de Sistemas			
Producción			
Patrocinador del proyecto			
Líder de proyecto usuario			

5. REQUERIMIENTOS FUNCIONALES

FUNCIÓN	DESCRIPCIÓN	DESCOMPOSICIÓN EN PROCESOS
Entradas	actual / requerida	
Salidas		
Interfases		

INVENTARIO	DESCRIPCIÓN	LAYOUT ANEXO
Reportes	actual / requerida	
Consultas		
Formatos		

CÓDIGO: BI/EF/01 DOCUMENTO CONTROLADO POR ORGANIZACIÓN Y MÉTODOS



Sistema xxxxx
Reporte de Avance
Periodo del dd/mmm/200x al dd/mmm/200a

1. DISTRIBUCIÓN

2. RESUMEN EJECUTIVO

Actividad	Responsable	% Planeado	% Real	% Desviación	Severidad (Alta/Media/Baja)
0. Planeación					
1. Análisis					
2. Diseño					
3. Desarrollo					
4. Pruebas					
5. Conversión					
6. Implementación					

3. ACTIVIDADES REALIZADAS

4. PENDIENTES DE INFRAESTRUCTURA

5. PENDIENTES DE ARQUITECTURA

6. DETALLE DE ACTIVIDADES

7. ACTIVIDADES POR REALIZAR EN EL SIGUIENTE PERIODO DEL dd/mmm/aaaa AL dd/mmm/aaaa

8. RIESGOS DEL PROYECTO

Riesgo	Grado Riesgo (Alta/Media/Baja)	Impacto (Alta/Baja)	Acción preventiva

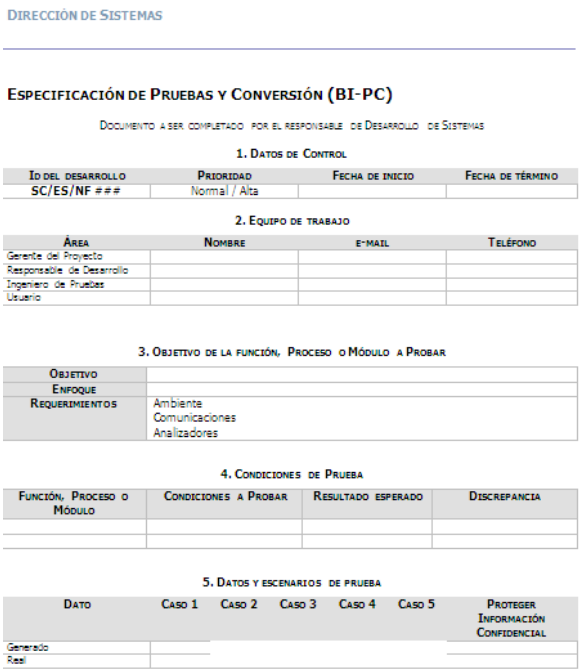
9. CAMBIOS IDENTIFICADOS

Firmas de Conformidad

Dirección Sistemas	Dirección Usuario
Gerente de Proyecto	Gerente de Desarrollo

Figura C1. Formato Especificación funcional (Procesos GF, 2006)

Figura C2. Formato Reporte de avance (Procesos GF, 2006)



DIRECCIÓN DE SISTEMAS

ESPECIFICACIÓN DE PRUEBAS Y CONVERSIÓN (BI-PC)

DOCUMENTO A SER COMPLETADO POR EL RESPONSABLE DE DESARROLLO DE SISTEMAS

1. DATOS DE CONTROL

ID DEL DESARROLLO	PRIORIDAD	FECHA DE INICIO	FECHA DE TÉRMINO
SC/ES/NF ###	Normal / Alta		

2. EQUIPO DE TRABAJO

ÁREA	NOMBRE	E-MAIL	TELÉFONO
Gerente del Proyecto			
Responsable de Desarrollo			
Ingeniero de Pruebas			
Usuario			

3. OBJETIVO DE LA FUNCIÓN, PROCESO O MÓDULO A PROBAR

OBJETIVO	
ENFOQUE	
REQUERIMIENTOS	Ambiente Comunicaciones Analizadores

4. CONDICIONES DE PRUEBA

FUNCIÓN, PROCESO O MÓDULO	CONDICIONES A PROBAR	RESULTADO ESPERADO	DISCREPANCIA

5. DATOS Y ESCENARIOS DE PRUEBA

DATO	CASO 1	CASO 2	CASO 3	CASO 4	CASO 5	PROTEGER INFORMACIÓN CONFIDENCIAL
Generado						
Real						

DIRECCIÓN DE SISTEMAS

FORMA DE REPORTE DE INCIDENTE SISTEMA XXXX BI-RI- xxxx

Fecha:

A la atención de:

Producto:
Prioridad: Alta / Media / Baja
Código: Incidente / Consultoría / Mejora
Contacto:

Descripción del incidente:

Módulo:

Bitácora de cambios:

Revisado por
Nuevo - @ por

Firma: _____

Fecha: _____

Figura C3. Formato pruebas y conversión (Procesos GF, 2006)

Figura C4. Formato Reporte de Incidencia (Procesos GF, 2006)

APÉNDICE D

COMPILACIÓN Y CONFIGURACIÓN DE NEGOCIO Y *BATCH*

```
SETLOCAL
@echo off
cd "C:\bea92\user_projects\domains\DOMINIO_BATCH"

@echo *         Escribe branc ...       *
set /P brach=Branch:

@echo ************************************************************************
@echo *                  Limpiando carpetas de negocio ...                   *
@echo ************************************************************************
@echo.

rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_DOMAIN\autodeploy\negocio-SI.ear
@echo *                 Archivo [negocio-SI.ear] ...borrado                  *
@echo.
del /Q   C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\logs\*.*
@echo *                          ...carpeta limpia                           *
@echo.
del /Q   C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\logs\multif\*.*
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\servers\SI_Server\tmp
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\servers\SI_Server\logs
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\servers\SI_Server\cache
@echo *                          ...carpeta limpia                           *
@echo.

@echo ************************************************************************
@echo *                  Limpiando carpetas de Batch ...                     *
@echo ************************************************************************
@echo.

del /Q/s "C:\bea92\user_projects\domains\DOMINIO_BATCH\autodeploy\DOMINIO_BATCH.ear"
@echo *                Archivo [DOMINIO_BATCH.ear] ...borrado                *
@echo.
del /Q   C:\bea92\user_projects\domains\DOMINIO_BATCH\logs\*.*
@echo *                          ...carpeta limpia                           *
@echo.
del /Q   C:\bea92\user_projects\domains\DOMINIO_BATCH\logs\multif\*.*
@echo *                          ...carpeta limpia                           *
@echo.
del /Q   C:\bea92\user_projects\domains\DOMINIO_BATCH\logs\batch\*.*
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_BATCH\servers\DOMINIO_BATCH_Server\cache
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_BATCH\servers\DOMINIO_BATCH_Server\stage
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_BATCH\servers\DOMINIO_BATCH_Server\tmp
@echo *                          ...carpeta limpia                           *
@echo.
rd /Q/s  C:\bea92\user_projects\domains\DOMINIO_BATCH\servers\DOMINIO_BATCH_Server\logs
@echo *                          ...carpeta limpia                           *
@echo.
@echo.
cd "%brach%\build"
call "setenv.cmd"
@echo.
call  "C:\COMPILACION\compilaNegocio.cmd"
@echo *                          ...Negocio Compilado                        *
@echo.
@echo ************************************************************************
@echo *                          ...Quitando [ ventanilla-SI.war ]           *
cd "C:\bea92\user_projects\domains\DOMINIO_NEGOCIO\autodeploy\negocio-SI.ear"
del ventanilla-SI.war
```

Figura D1. Archivo de compilación nb.cmd (Sistemas Administrativos GF, Eric López, 2007)

Tabla D1. Configuración variables de ambiente (Sistemas Administrativos GF, Eric López, 2007)

Configuración de archivos setenv.cmd y *batch*_setenv.cmd	
Set BEA_HOME=C:/bea92 set ANT_HOME=C:/Server_Util/apache-ant-1.6.2 set CHECKSTYLE_HOME=C:/Server_Util/checkstyle-3.4 set JUNIT_HOME=C:/Server_Util/junit3.8.1	Se crearon variables de ambiente para BEA (servidor de aplicación), ANT (tareas automatizadas hormiga), *CHECKSTYLE* (especificaciones de java) y JUNIT (pruebas).
set CMP_VERSION=V1-9-9-C	Fue el código de identificación de la versión del branch de trabajo.
set PROXY_URL=14.9.11.7 set WS_BAN_URL=14.9.1.5:4111 set MAIL=14.9.2.235 set HOST=10.1.5.1 set SMS=14.9.20.18:9011 set WS_VERIFICA=14.9.3.85:6010	Se asignó el valor a la variable de ambiente para conectar con otros sistemas a través de los *web services*, estas variables fueron tomadas para el *application-configuration*.xml donde existieron más de diez variables de ambiente para conectar varios sistemas externos y cada sistema externo tenía hasta seis *web services* cada uno, como estaban en servidores coincidentes, con una variable se podía conectar a varios servicios ya que estaban en el mismo servidor.
set LOG4J_CONFIG=../config/blf/log4j/log4j-config.xml	Se asignó el valor de la variable de ambiente para Biblioteca de salida de los mensajes o "logs".
set JAVA_HOME=%BEA_HOME%/jdk150_10	Se asignó el valor de la variable de JAVA_HOME.
set PATH=%JAVA_HOME%/bin;%PATH%	Se integró al PATH el valor de JAVA.
set PATH=%ANT_HOME%/bin;%PATH%	Se integró al PATH el valor de ANT.
set PATH=%CHECKSTYLE_HOME%;%PATH%	Se integró al *PATH* el valor de CHECKSTYLE.
echo CLASSPATH=%CLASSPATH%	Se especificó pintar en consola el valor de CLASSPATH.
echo PATH=%PATH%	Se especificó pintar en consola el valor de PATH.

REFERENCIAS

Aguado F.E. (2010). Reporte Anual del Grupo Financiero 2009, [en línea]. Recuperado el 15 de noviembre de 2010, de http://www.inbursa.com/ReIn/infoanual2009.pdf.

Albiol, F. R. (2004). *JasperReports, iReports y subreportes*.

Apache Jmeter Project (2006), [en línea]. USA.: Apache Foundation. Recuperado el 28 de Abril del 2007 http://jakarta.apache.org/jmeter/usermanual/intro.html

Armstrong, E.,Bodoff, S., Carson D., Fisher, M., Fordin, S., Green, D., Haase, K. y Jendrock, E. (2003). *The Java Web Services Tutorial*. California, USA.: Sun Microsystems Inc.

Bales, D. (2002) (1a ed.). *Java Programming with Oracle JDBC*. California, USA.: O'reilly.

Bea Systems. (2006). *Bea Products Installation Guide*. USA.: Bea Systems, Inc.

Chappell, D, y Jewell, T. (2002)(1a ed). *Java Web Services*. California, USA.: O'Reilly & Associates Inc.

Cooper, J. W. (1998). *Design Patterns Java Companion*. USA.: Addison Wesley.

Coronel C., E (2005). *ORACLE 9i-SQL & Program Language/Structured Query Language*. Perú Universidad Nacional de Ingeniería Centro de Extensión y Proyección Social.

Dominguez, G. A. (2009). *Aprenda Reportes con el plugin de Ireport para Netbeans y MySQL*. Mexico, Df.: Dominguez, G. A.

Goodwill, J. y Hightower, R. (2004). *Professional Jakarta Struts*. Indiana, USA.: Wiley Publishing, Inc.

Kline, K. E. (2004) (2da ed.). *SQL in a Nutshell*. California, USA.: O'reilly Media Inc.

Philipe Kruchten, (2007) (11a ed.). *The Rational Unified Process Made Easy*: Massachusetts, USA.:Addison Wesley.

Pilone, D. y Pitman, N. (2005) (1ª ed.). *UML 2.0 in a Nutshell*. California, USA.: O'reilly Media Inc.

Powell, G. y McCulloght-Dieter, C. (2005). *Oracle SQL jumpstart with Examples*. MA., USA.: Elsevier Digital Press.

Shohoud, Y. (2003). *Real World XML Web Services*. Pearson Education, Inc.

Sinan, S. A. (2003). Learning UML. Sebastopol, CA, USA.: O´Reily & Associates, Inc.

Wutka, M., Moffet, A. y Mittal, K. (2004). *Sams Teach Yourself JavaServer Pages 2.0 With Apache Tomcat in 24 Hours*. USA.: Sams Publisher.

GLOSARIO

API	Interfaz de programación de aplicaciones o conjunto de funciones y procedimientos de la plataforma java.
ATM	Cajeros automáticos mediante transacciones asíncronas.
Back-end	Comprende los componentes que procesan consultas a la base de datos.
BANXICO	Banco de México.
CNBV	Comisión Nacional Bancaria y de Valores.
Commit	Confirmar o comprometer la transacción.
Conversión	Fase de Liberación como parte de la metodología RUP.
Cron	Administrador regular de procesos en segundo plano.
DAO	Patrón utilizado para el acceso a datos.
Deployar	Montar o distribuir el proyecto en el servidor de aplicación.
Deserializar	Proceso de decodificación de un objeto para extraer su información.
Dispatcher	Componente despachador de transacciones.
DTO	Patrón de diseño *Data transfer object para* transferir datos entre capas.
EJB	Enterprise java *beans*, son vean de negocio ubicados del lado del servidor.
Framework	Marco de trabajo basado en modelos previamente creados.
Front-end	Parte del sistema de software que interactúa visualmente con el usuario.
GF	Grupo Financiero.
GoF	patrón *Gang Of Four.*
HTTP	*Hypertext Transfer Protocol*, protocolo de transferencia de hipertexto.
IAVE	Tarjetas prepago para peaje en autopistas.
IDEAL	Impulsora del Desarrollo y el Empleo en América Latina.
IoC	Inversión of Control o control invertido, es un concepto utilizado en el desarrollo de una aplicación, en otras palabras las clases no buscan las clases dependientes de ellas.
IWB	Sistema Instantáneo Mundial de pago único que operaba en el GF.

JNDI	Java *Naming* and *Directory* Interface, es Interfaz de Nombrado y Directorio Java.
Job	Tarea por lotes programada de forma asíncrona.
JPA	Java *Persistence* API, más conocida por sus siglas JPA, es la librería de java para persistencia.
JTA	Java *Transaction* API, es la librería de java para transacciones.
JVM	Maquina Virtual de Java.
LOG4J	Biblioteca de código abierto que permite escribir en archivos bitácora *.log
MVC	Modelo Vista Controlador.
Patrón	Solución de diseño de software a un problema.
Performance	Rendimiento.
Plugin	Aplicación que interactuó con otra aplicación para aportarle una función o utilidad específica.
Pointcuts	Unificaciones para asociar expresiones
Pojo	Objeto plano de java.
QA	Área encargada de realizar las pruebas de control de calidad.
Request	Peticion enviada por medio de formularios por URL.
Serializar	Proceso de codificación usado para transportar objetos a través de una red.
SHCP	Secretaria de Hacienda y Crédito Público.
SI	Sistema Integral del GF.
SOAP	*Simple Object Access Protocol,* es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse mediante intercambio de datos XML.
SOI	Sistema de Operaciones Internacionales.
SVN	Repositorio vía subversión para el control de versiones de proyectos.
VB.NET	*Visual Basic* .NET, es un lenguaje de programación.
VO	Objeto de java que contiene atributos con sus respectivos métodos *get*() y *set*().

WSDL	son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web.
XML	*e**X**tensible **M**arkup **L**anguage* (lenguaj de marcas extensible).

FORMATOS

BI-RPN	Requerimiento de Producto de Negocio.
BI-PT	Administración del Plan de trabajo.
BI-EE	Estimación del esfuerzo.
BI-ACB	Análisis costo-beneficio.
BI-ET	Especificación técnica.
BI-PC	Especificación de pruebas y conversión.
BI-QA	Monitoreo de desempeño del sistema.
BI-DA	Documento de aceptación.
BI-RI	Documento Requerimiento de incidencia.

DOCUMENTOS

RUP-VISION	Documento de Visión y *stakeholder´s* o patrocinador del proyecto.
RUP-MODELO DE NEGOCIO	Documento descriptivo del Negocio.
RUP-ARQUITECTURA	Documento de Arquitectura tecnológica.
BI-ODS	Documento de Orden de Servicio.
BI-DOC ARTE	Documento de diseño de Interfaz gráfica.
BI-CVS	Esquema de trabajo con CVS.
RUP-M PRUEBAS U	Matriz Pruebas Unitarias.
RUP-M PRUEBAS F	Matriz Pruebas Funcionales.

PLANES DE TRABAJO

PT-PLANEACION	Plan de trabajo.
PT-MEDIDA	Plan de trabajo.
PT-PAQUETE	Plan de trabajo.
PT-ITERATIVO	Plan de trabajo.

DIAGRAMAS

D-CASO DE USO	Diagrama de caso de uso.
D-SECUENCIA	Diagrama de secuencia.
D-ESTADOS	Diagrama de transición de estados.
D-CLASES	Diagrama de Clases persistentes.
D-ER	Diagrama de entidad relación.

CLAVES PARA SOLICITAR CAMBIOS EN LA BASE DE DATOS ORACLE

Clave	**Descripción**
001-ES	Modificación, Borrado o Creación de una nueva Relación (PK, FK, AK etc.).
002-TA	Modificación, Borrado o Creación de una nueva Tabla.
003-VI	Modificación, Borrado o Creación de una nueva Vista.
004-SE	Modificación, Borrado o Creación de una nueva Secuencia.
006-IN	Modificación, Borrado o Creación de un nuevo Índice.
007-DA	Modificación, Borrado o Inserción de datos en una Tabla.

CLAVES DE PERMISOS DE BASE DE DATOS ORACLE

Clave	**Descripción**
100-NNEG	NEGOCIO – NEGOCIO
101-SNEG	SERVICIOS – NEGOCIO
102-BNEG	BATCH – NEGOCIO
103-BBCH	BATCH – JOBS
104-SESC	ESTRUCTURA CORPORATIVA
105-BIMP	IMPRESIÓN
106-BREP	REPORTES
107-BSEG	SEGURIDAD

CLAVES DE PRIVILEGIOS DE BASE DE DATOS ORACLE

S	*Select*
I	*Insert*
U	*Update*
D	*Delete*
E	Ejecución