UNIVERSIDAD AUSTRAL DE CHILE

FACULTAD DE CIENCIAS DE LA INGENIERIA

ESCUELA DE INGENIERIA CIVIL EN INFORMATICA

ANALISIS Y DISEÑO MODULAR DE UN SITIO WEB PARA BANCOESTADO.

TESIS DE GRADO PARA OPTAR AL TITULO PROFESIONAL DE

INGENIERO CIVIL EN INFORMÁTICA

PATROCINANTE:
MARTIN SOLAR
DOCENTE INSTITUTO DE INFORMATICA.

COPATROCINANTE:
ADA ESPOZ GARIN
INGENIERO EJECUCION INFORMATICA.

MANUEL IVAN BARRERA CERNA

RODRIGO EDUARDO ROSALES GONZALEZ

VALDIVIA – CHILE

2003

Valdivia, 24 de Junio de 2003

De : Martín Gonzalo Solar Monsalves

A : Directora Escuela Ingeniería Civil en Informática

Ref. : Informe Calificación Trabajo de Titulación

Nombre Trabajo de Titulación:

"ANÁLISIS Y DISEÑO MODULAR DE UN SITIO WEB PARA BANCOESTADO".

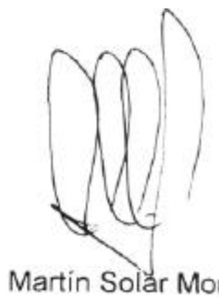
Nombre Alumnos:

Manuel Iván Barrera Cerna - Rodrigo Eduardo Rosales González

Evaluación:

Cumplimiento del objetivo propuesto	7.0
Satisfacción de alguna necesidad	7.0
Aplicación del método científico	6.5
Interpretación de los datos y obtención de conclusiones	6.0
Originalidad	7.0
Aplicación de criterios de análisis y diseño	7.0
Perspectivas del trabajo	7.0
Coherencia y rigurosidad lógica	6.5
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración	6.5
Nota Final	**6.7**

Sin otro particular, atte.:



Martín Solar Monsalves

Fecha : Martes 8 de Julio de 2003

DE : Ada M. Espoz Garín (Jefe Proyecto BancoEstado).

A : DIRECTORA ESCUELA INGENIERÍA CIVIL EN INFORMÁTICA

MOTIVO: Evaluación del Proyecto de Tesis.

INFORME TRABAJO DE TITULACIÓN

Nombre Trabajo de Titulación:

"Análisis y Diseño Modular de un Sitio Web para BancoEstado".

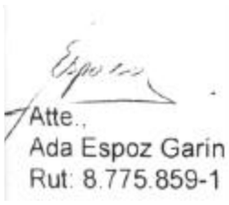
Nombre del Alumno:

Manuel Iván Barrera Cerna.
Rodrigo Eduardo Rosales González.

Nota : 6.9 (Seis coma Nueve).

FUNDAMENTO DE EA NOTA:

Excelente el enfoque dado como equipo de trabajo para cumplir con el Objetivo Propuesto en la elaboración de su seminario. Las ideas estuvieron claras desde un principio lo que conllevó a la aplicación metódica de los puntos planteados en la Tesis en forma rigurosa y analítica.



Universidad Austral de Chile
Instituto de Informática

Valdivia, 07 de julio de 2003
Comunicación Interna Nº 085

De : Luis Hernán Vidal Vidal.
A : Sra. Miguelina Vega R.
Directora de Escuela de Ingeniería Civil en Informática
Ref. : Informa Calificación Trabajo de Titulación

MOTIVO: Informar revisión y calificación del Proyecto de Título "Análisis y diseño modular de un sitio Web para Banco Estado.", presentado por los alumnos Manuel Iván Barrera Cerna y Rodrigo Eduardo Rosales González, que refleja lo siguiente:

Se logró el objetivo planteado de entregar un análisis y diseño modular de un sitio web, sitio que corresponde a una entidad financiera de importancia a nivel nacional.

La revisión hecha sobre los estándares y tecnologías empleadas entregan una base referencial clara y de gran valor al momento de indicar las posibles alternativas de solución.

Aun cuando hubiese sido interesante el haber provisto la información de rendimientos de los distintos entornos evaluados, dadas las características de equipos y licencias de software involucradas, emplear información de los mismos proveedores y refutar esto con benchmark ejecutados por Middleware Company se acepta como una buena solución.

Por todo lo anterior expuesto califico el trabajo de titulación de los señores Manuel Iván Barrera Cerna y Eduardo Rosales González con nota 7,0 (siete como cero).

Sin otro particular, se despide atentamente.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMÁTICA
FACULTAD DE CIENCIAS DE LA INGENIERÍA

Prof. Ing. Luis Hernán Vidal Vidal.
Instituto de Informática
Universidad Austral de Chile.

General Lagos 2086 · Campus Miraflores · Valdivia · Chile
Casilla 567 · Fono: 56 63 221427 · Fax: 56 63 293115 · instituto@inf.uach.cl · www.inf. uach.cl

AGRADECIMIENTOS.

Dedico este trabajo a mi hermano Michael y a mi madre Tránsito Cerna, la que con su esfuerzo y eterno apoyo ayudó a que este momento se hiciera realidad. Todo lo que soy hoy en día te lo debo a ti vieja querida, gracias por ser como eres.

A la directora y personal del hogar Luterano de Valdivia, a todos aquellos que ayudaron a mi formación profesional y en especial a la memoria de mi abuelo Pablo Cerna.

Manuel

A mis padres, Eduardo y Luisa, por el apoyo incondicional.

A Paola, mi compañera de siempre, por todo su amor y a nuestro futuro hijo.

Rodrigo.

En conjunto quisiéramos agradecer sinceramente a Ada Espoz, Martín Solar y a la gente de BancoEstado, por todo su apoyo.

INDICE

INDICE DE FIGURAS Y TABLAS		4
RESUMEN		6
SUMMARY		7
	INTRODUCCION	9
1.1	Antecedentes existentes	12
1.2	Importancia y naturaleza del estudio	14
1.3	Descripción de la metodología	15
1.4	Objetivos	18
1.5	Inclusiones y exclusiones	19
1.5.1	Inclusiones	19
1.5.2	Exclusiones	19
2.	**LEVANTAMIENTO DE REQUERIMIENTOS**	22
2.1	Estado actual del sitio web de *BECH*	23
2.1.1	Hardware y sofware utilizado	23
2.1.2	Conexiones	25
2.1.3	Arquitectura actual de *BECH*	25
2.1.3.1	*DNA*	26
2.1.3.2	Implementación de *DNA* en *BECH*	29
2.2	Detección de los principales errores y problemas del sitio web	30
2.2.1	Motor de presentación	30
2.2.2	Motor de negocio	31
2.2.3	Manejo de errores	32
2.2.4	Sistema de *log*	34
2.3	Otras falencias del sitio web	36
2.3.1	Control de código fuente	36
2.3.2	Unificación de ambientes	37
2.3.3	Problemas con las bases de datos	38
3.	**ANÁLISIS Y SOLUCIÓN TEÓRICA**	40
3.1	Motor para la capa de presentación	40
3.1.1	Que se entiende por motor para la capa de presentación	40
3.1.2	Objetivos del Motor de Presentación	40
3.1.3	Entidades relacionadas con el motor de presentación	41
3.1.3.1	Funcionamiento del motor de presentación	42
3.1.3.2	Funcionamiento en el sitio privado	43
3.1.3.3	Funcionamiento en el sitio público	43
3.1.4	Entradas y salidas del motor de presentación	44
3.1.4.1	Motor de presentación – Páginas dinámicas	44
3.1.4.2	Motor de presentación – Capa de negocio	44
3.1.5	Modelo Final	45
3.2	Motor de negocios	46
3.2.1	Que se entiende por motor de negocios	46
3.2.2	Objetivos del motor de negocios	46
3.2.3	Entidades relacionadas con el motor de negocio	47
3.2.4	Entradas y salidas del motor de negocio	48
3.2.4.1	Motor de negocio – Página dinámica	48
3.2.4.2	Motor de negocio – Capa de datos	49
3.2.5	Modelo Final	49
3.3	Manejador de errores o excepciones	50
3.3.1	Qué es una excepción	50
3.3.1.1	Qué se entiende por manejador de errores	51
3.3.2	Objetivo del manejador de errores	51
3.3.3	Entidades relacionadas con el manejador de errores	52
3.3.4	Entradas y salidas del manejador de errores	53
3.3.4.1	Manejador de errores – Páginas dinámicas	53
3.3.4.2	Manejador de errores – Componentes	54
3.3.5	Modelo final	55
3.4	Sistema *log* transaccional	56
3.4.1	Que se entiende por sistema *log* tradicional	57
3.4.2	Que se entiende por sistema log transaccional	58
3.4.3	Objetivo del sistema de log transaccional	59
3.4.4	Entidades relacionadas con el sistema de log	60
3.4.5	Entradas y salidas del sistema de log	61

3.4.5.1 Sistema de log – Páginas dinámicas 61
3.4.5.2 Sistema de log – Componentes 61
3.4.6 Modelo Final 62
3.5 Otras falencias del sitio web de BECH 63
3.5.1 Falencias que no serán resueltas 63
3.5.1.1 Problemas con los servicios de datos 64
3.5.1.2 Ambientes de desarrollo, test y producción 64
3.5.1.3 Pasos a producción 65
3.5.2 Solución a otros problemas del sitio web 65
3.5.2.1 Manejador de fuentes 66
4. IMPLEMENTACION TEORICA 68
4.1 Definición de conceptos 68
4.1.1 Especificación de seudocódigo 69
4.2 Motor para la capa de presentación 70
4.2.1 Implementacion del motor de presentación 71
4.2.1.1 Relación entre archivo XSL e información solicitada 72
4.2.1.2 Aplicar una plantilla XSL a un XML 73
4.2.1.3 Generador de Presentación 74
4.3 Motor para la capa de negocios 74
4.3.1 Modelo motor para la capa de negocios 75
4.3.2 Implementación del motor para la capa de negocios 75
4.3.2.1 DLL única para la obtención de datos. 75
4.3.2.1.1 Obtener tipo de base y servicio de datos a utilizar 76
4.3.2.1.2 Método para ejecución de servicio de datos 76
4.3.2.2 DLL única para lógica de negocios 77
4.3.2.2.1 Método para llamar a la DLL única para la obtención de datos 77
4.3.2.2.2 Método para obtener formato XML 77
4.3.2.2.3 Método para generar XML de respuesta 78
4.4 Manejador de errores o excepciones 79
4.4.1 Interacción con las entidades relacionadas 79
4.4.2 Definición del funcionamiento de la componente manejadora de errores 80
4.4.2.1 Almacenamiento del error 80
4.4.2.2 Entrega de mensajes a clientes y acciones a seguir 81
4.4.2.3 Envío de mensajes 81
4.4.2.4 Generación de reportes de error 82
4.4.3 Manejo de errores en Visual Basic y VBScript 82
4.4.3.1 Instrucción "On Error" 82
4.4.4 Implementación del Manejador de Errores 84
4.4.4.1 Método que almacena en el error 85
4.4.4.2 Método que rescata el mensaje para los usuarios y las acciones a seguir 86
4.4.4.3 Método que envía mensajes 87
4.4.4.4 Método que genera reportes de error 88
4.4.4.5 Controlador de errores 89
4.5 Sistema log transaccional 90
4.5.1 Interacción con las entidades relacionadas 90
4.5.2 Definición del funcionamiento del sistema de log transaccional 91
4.5.2.1 Almacenamiento de la transacción 92
4.5.2.2 Almacenamiento de la entrada y salida de los servicios de datos 93
4.5.2.3 Lectura de los movimientos realizados por un cliente 93
4.5.3 Implementación del sistema de log transaccional 93
4.5.3.1 Método que almacena una transacción 94
4.5.3.2 Método que almacena la entrada y salida de un servicio de datos 95
4.5.3.3 Método que lee los movimientos realizados por un cliente 96
4.6 Ejemplo integrado 97
4.7 Manejador de fuentes 98
5. ESTUDIO DE PLATAFORMAS 102
5.1 Servidor web 102
5.2 Servidor de aplicación 103
5.2.1 Funcionalidades adicionales de un servidor de aplicación 105
5.2.2 Tecnología para implementar servidores de aplicaciones. 106
5.2.2.1 Alternativa J2EE 106
5.2.2.2 Alternativa No-J2EE 106
5.2.2.3 Alternativa Microsoft 107
5.3 Estandar para servidores de aplicación J2EE 107

5.3.1	Modelo del estandar J2EE	108
5.3.2	Componentes del estandar J2EE	110
5.3.2.1	Java Servlets	110
5.3.2.2	JavaServer Pages	111
5.3.2.3	Enterprise JavaBeans	111
5.3.3	Principales API dentro del estandar J2EE	112
5.3.4	Funcionamiento de la plataforma J2EE	114
5.3.4	Beneficios del estandar J2EE	115
5.4	Microsoft .Net	116
5.4.1	Microsoft .NET Framework	116
5.4.1.1	Lenguaje Común de Ejecución	118
5.4.1.2	Clases de librería Microsoft a.NET Framework	119
5.4.1.2.1	Microsoft ASP .NET	119
5.4.1.2.2	Microsoft ADO .NET	120
5.5	Servidores de Aplicación basados en J2EE	120
5.5.1	BEA WebLogic Server (WLS)	121
5.5.2	SUN iPlanet Aplication Server (iAS)	123
5.5.3	IBM WebSphere Application Server (WAS)	125
6.	**EVALUACIÓN Y SELECCION DE PLATAFORMA**	128
6.1	Microsoft .Net v/s J2EE	128
6.1.1	Analogías y diferencias entre J2EE y .NET	129
6.1.1.1	Sistema de Tiempo de ejecución	129
6.1.1.2	Acceso a bases de datos	130
6.1.1.3	Páginas dinámicas	132
6.1.1.4	Lenguajes de programación	133
6.1.1.5	Herramientas de desarrollo	134
6.1.1.6	Tabla teórica comparativa	135
6.2	Malla de evaluación	136
6.2.1	Definición de los parámetros a cuantificar	137
6.3	Descripción de fuentes investigadas	140
6.3.1	MiddleWare Company	140
6.3.2	Java PetStore y .NET Pet Shop	140
6.3.3	Preparando el ambiente antes de ejecutar el benchmark	141
6.3.4	El benchmark	143
6.3.4.1	Benchmark de aplicaciones web	143
6.3.4.2	Benchmark de transacciones distribuidas durante 24 horas.	149
6.3.4.3	Benchmark de servicios web	150
6.4	Utilizando la malla de evaluación	155
6.4.1	Valores para los parámetros a evaluar	155
6.4.2	Malla final	158
7.	PROPUESTA DE PLAN DE IMPLEMENTACION	162
7.1	Comparación entre ***DNA*** *y* .NET	162
7.2	Plan de implementación	163
8.	**CONCLUSIONES**	**167**
BIBLIOGRAFIA		**169**
APENDICE 1, Abreviaturas		**171**
APENDICE 2, Glosario		**173**
ANEXO N° 1, Lista de Servicios		**175**
ANEXO N° 2, Lista de reportes de error		**177**
ANEXO N° 3, Carta Gantt con plan de implementación		**181**

INDICE DE FIGURAS Y TABLAS

Figura Cap2_1	Topología de los servidores web de BancoEstado	24
Figura Cap2_2	Modelo DNA	28
Figura Cap2_3	Modelo DNA implementado por BECH	29
Figura Cap2_4	Ejemplo de Mensaje de error desplegado a clientes en sitio web de BECH	33
Figura Cap2_5	Segundo ejemplo de mensaje de error desplegado a clientes en sitio web de BECH	33
Figura Cap3_1	Funcionamiento esperado para el motor de presentación	41
FIgura Cap3_2	Funcionamiento del motor de presentación en el sitio web	42
Figura Cap3_3	Motor para la capa de presentación (cuadro inter-lineado)	45
Figura Cap3_4	Ubicación del motor de negocio en un modelo de tres capas	47
Figura Cap3_5	Motor para la capa de negocios (cuadro inter-lineado)	49
Figura Cap3_6	Modelo para el manejador de errores	56
Figura Cap3_7	Sistema de log transaccional	62
Figura Cap5_1	Arquitectura de funcionamiento de un servidor web	103
Figura Cap5_2	Arquitectura de funcionamiento de un servidor de aplicaciones	104
Figura Cap5_3	Modelo estándar J2EE	108
Figura Cap5_4	Relación entre el servidor J2EE y contenedores	115
Figura Cap6_1	Sistema de Tiempo de ejecución para .NET	129
Figura Cap6_2	El sistema de tiempo de ejecución para J2EE	130
Figura Cap6_3	Acceso a base de datos en .NET	131
Figura Cap6_4	Acceso a bases de datos en Java	131
Figura Cap6_5	Arquitectura de ASP.NET	132
Figura Cap6_6	Arquitectura de JSP	132
Figura Cap6_7	Parámetros cuantificables en malla de evaluación	139
Figura Cap6_8	Mejores respuestas sin descarga de imágenes	144
Figura Cap6_9	Carga máxima de usuarios concurrentes sin descarga de imágenes	145
Figura Cap6_10	Curvas de Salida para servidor de aplicaciones con 2 CPU	145
Figura Cap6_11	Curvas de salida para páginas por segundo con servidor de aplicaciones con 2 CPU	146
Figura Cap6_12	Curvas de salida para páginas por segundo con servidor de aplicaciones con 8 CPU	146
Figura Cap6_13	Mejores respuestas con descarga de imágenes	147
Figura Cap6_14	Carga máxima de usuarios concurrentes con descarga de imágenes	147
Figura Cap6_15	Curvas de Salida para páginas por segundo con Servidor de aplicaciones con 2 CPU	148
Figura cap6_16	Curvas de salida para páginas por segundo con servidor de aplicaciones con 8 CPU	148
Figura Cap6_17	Mejores Respuestas para servicios web	151
Figura Cap6_18	Carga máxima de usuarios para serivicios web	152
Figura Cap6_19	Curvas de salida para llamadas a servicios web SOAP con 2 CPU	152
Figura Cap6_20	Curvas de salida para llamadas a servicios web SOAP con 8 CPU	153
Figura Cap6_21	Mejor respuesta de servicios web via proxy	153
Figura Cap6_22	Carga máxima de usuarios para servicios web via proxy	154
Figura Cap6_23	Curva de respuesta para servicios web via proxy con servidor de aplicación con 2 CPU	154
Figura Cap6_24	Curva de respuesta para servicios web via proxy con servidor de aplicación con 8 CPU	155

Tabla Cap1_A	Proyección anual de BECH	12
Tabla Cap2_A	Cantidad de conexiones por servidor en diferentes horarios	25
Tabla Cap4_A	Formato de la instrucción On Error	83
Tabla Cap5_A	Servidores de iAS.	125
Tabla Cap6_A	Comparativa teórica de características J2EE y .NET	135
Tabla Cap6_B	Tabla resumen del benchmark de transacciones realizadas	150
Tabla Cap6_C	Resumen por servidor para páginas por segundo sin descarga de magen	156

Tabla Cap6_D	resumen por servidor paramáximo número de usuarios en un mismo instante sin descarga de imagen	156
Tabla Cap6_E	Resumen por servidor para páginas por segundo con descarga de imagen	157
Tabla Cap6_F	Resumen por servidor paramáximo número de usuarios en un mismo instante con descarga de imagen	157
Tabla Cap6_G	Resumen por servidor para máximo número de transacciones en línea por segundo en un tiempo dado	158
Tabla Cap6_H	Resultado de la malla de evaluación aplicada a los distintos servidores de aplicación	159

RESUMEN.

Este proyecto de tesis nace debido a que BancoEstado tiene la necesidad de construir una nueva versión de su sitio web, ya que la que actualmente tiene se ha vuelto ineficiente al momento de incorporar nuevas funcionalidades e incapaz de soportar la carga de usuarios y transacciones proyectadas por la propia institución.

Realizar un "Análisis y diseño modular de un sitio web para BancoEstado" consiste en construir un diseño teórico formado por componentes fuertemente integrados junto con la plataforma tecnológica más adecuada para una eventual implementación.

Para conseguir estos objetivos primeramente se tiene una etapa de levantamiento de requerimientos, en la cual se determina el estado actual del sitio web. Posteriormente se detectan los problemas principales y se diseña una solución teórica para cada una de ellos, compuesta por módulos separados pero fuertemente dependientes entre sí. Esta solución es implementada en forma teórica con la tecnología actualmente disponible, para ilustrar la forma correcta de implementación de las soluciones encontradas.

A continuación se presenta una investigación, estudio y selección de la plataforma tecnológica más adecuada para la eventual construcción de un sitio web financiero, para finalmente proporcionar un plan de implementación teórico en miras a construir una nueva versión del sitio web con las soluciones propuestas.

SUMMARY.

This thesis project is born because BancoEstado has the necessity to construct a new version of its Web site, since the one that at the moment it has it has become inefficient at the time of incorporating new functionalities and incapable to support the load of users and transactions projected by the this institution.

To make an "Analisis and modular design of a web site for BancoEstado" consists of constructing a theoretical design formed by components strongly integrated along with the most suitable technological platform for a possible implementation.

In order to obtain these objectives firstly there is a stage of rise of requirements. Later, the main problems are detected and a theoretical solution for each one of them is designed, composed by separated modules but strongly employees to each other. This solution is implemented in theoretical form with the technology available at the moment, to illustrate the correct form of implementation of the found solutions.

Next, appears an investigation, study and selection of the technological platform more adapted for the possible construction of a financier web site, finally to provide a theoretical plan of implementation in sights to construct a new version of the web site with the propose solutions.

CAPITULO 1

INTRODUCCION

1. Introducción

La idea de presentar como proyecto de tesis un “Análisis y diseño modular de un sitio web para BancoEstado***”***, (en adelante ***BECH***), nació a raíz de que actualmente el sitio web de ***BECH*** ha cumplido con su objetivo inicial, el cual era posicionar a la institución estatal en el mundo web. Con el logro de este objetivo surge la necesidad de dar un paso más, el que corresponde dentro de los lineamientos de ***BECH*****,** a la construcción de una nueva versión del sitio que cumpla con los factores de calidad[1] deseados (calidad de servicio, servicios propios de negocio, apoyo tecnológico, soporte de imagen corporativa, alto nivel de seguridad, disponibilidad y calidad de la información), con el objeto de asegurar una atención masiva de calidad y eficiencia.

Por lo anterior, fue materia de estudio en este proyecto de tesis el estado del arte actual en el sitio web de ***BECH***, con la idea de presentar una solución teórica para cada una de las problemáticas del sitio, es decir, un diseño modular para el sitio, lo cual implicó realizar las siguientes tareas:

- Diseñar un motor único para la capa de presentación.
- Generar un módulo único para la obtención de datos y para el manejo de la lógica de negocio.
- Crear un estándar para el manejo de errores, tanto en las páginas como en las componentes utilizadas.
- Planteamiento de un sistema único de ***log***, que permita registrar cada una de las transacciones y errores en los que este involucrado el sitio web de ***BECH***.

---

[1] Factores de calidad: se identifican como factores críticos de éxito, de acuerdo a la percepción de los clientes y a la infraestructura o soporte interno.

Además, se estudiaron las plataformas existentes en el ambiente informático capaces de soportar el incremento de usuarios, conexiones y transacciones proyectadas por ***BECH***, con la intención de entregar una evaluación con la propuesta que mejor cumpla con las necesidades del nuevo diseño para el sitio web, utilizándose para ello una malla de evaluación, llevando finalmente las soluciones de los actuales problemas del sitio a esta nueva plataforma, entregándose un plan de implementación hacia la nueva solución.

Como definición para la malla de evaluación se consideraron como base los objetivos requeridos por ***BECH***, tales como; aumento en las conexiones, transacciones y visitas al sitio, además de otros tópicos como; seguridad, tiempo de respuesta, tamaño de las páginas, tiempo de descarga y comportamiento de los componentes. Todo el estudio, análisis y diseño de la malla de evaluación a considerar permitió asegurar y medir el impacto que tiene como la mejor solución obtenida por este proyecto de tesis.

El objetivo general del proyecto de tesis consiste en llevar a cabo las siguientes etapas, a saber: levantamiento de requerimientos de la situación actual, análisis y solución a problemática actual del sitio, investigar plataformas representativas del mercado, implementar solución teórica, evaluar la mejor solución como plataforma y entregar como propuesta un plan de implementación a la mejor plataforma evaluada, la cual debe ser capaz de proveer los mismos servicios que se encuentran actualmente disponibles en el sitio web de ***BECH*** considerando la incorporación de nuevos desarrollos.

Este proyecto de tesis esta organizado en ocho capítulos: El primer capítulo se ha denominado **"**Introducción", y está enmarcado dentro de los antecedentes generales de la organización, modelo del actual sitio web de ***BECH*** con su

arquitectura y funcionalidad, además de la metodología, planificación y objetivos.

El segundo capítulo denominado “Levantamiento de requerimientos”, comienza describiendo brevemente las consideraciones del diseño existente, basándose en las problemáticas actuales del sitio las cuales han permitido conocer las reales necesidades de ***BECH***. Posteriormente se presentan los resultados obtenidos del levantamiento con un estudio que indica la situación actual del portal internet de ***BECH***.

En el tercer capítulo denominado “Análisis y solución teórica” se estudian y solucionan los principales problemas existentes, los cuales establecen un adecuado diseño modular para el sitio y las formas que existen para fortalecerlo teniendo en cuenta que las soluciones entregadas son parte de un protocolo que es uniforme e independiente de tecnologías o plataformas.

El cuarto capítulo denominado “Implementación teórica”, permite representar cada una de las soluciones proporcionadas por el capítulo anterior con las herramientas tecnológicas que actualmente tiene ***BECH***, con el fin de optimizar la actual implemantación.

En el quinto capítulo denominado “Estudio de plataformas”, se desarrolla el tema de investigación y comparación de tecnologías representativas del mercado que permitan cumplir con los factores de calidad esperados para la nueva versión del sitio web de ***BECH***.

En el sexto capítulo denominado “Evaluación y selección de plataforma”, se selecciona la plataforma más adecuada para soportar el sitio web de ***BECH***, se describen los parámetros a evaluar y la importancia de cada uno en relación con los otros, se crea una malla de evaluación que refleja las fortalezas y debilidades de cada plataforma.

El capítulo siete denominado “Propuesta de plan de implementación”,

permite, a modo de compendio, integrar las soluciones teóricas obtenidas en el capítulo tres con la plataforma seleccionada en el capítulo seis.

En el capítulo ocho, denominado “Conclusiones”, se incorporan las conclusiones obtenidas en este proyecto de tesis.

Finalmente se encuentra la bibliografía, referencias electrónicas, apéndices (abreviaturas y glosario) y anexos empleados en este proyecto de tesis. El glosario contiene términos técnicos, propios de los entornos corporativos de informática, los cuales han sido marcados dentro de este documento con una impresión remarcada y en estilo cursivo. Caso similar ocurre con las abreviaturas, que a diferencia del glosario, estan escritas en mayúsculas.

1.1 Antecedentes existentes

BECH ha proyectado que la nueva versión del actual sitio web, cumpla con alcanzar a fines del año 2003, un importante crecimiento, tal como se refleja en la siguiente tabla:

PROYECCION ANUAL DE BANCOESTADO			
	Fines de 2002	Fines de 2003	Aumento
Conexiones	270.000	850.000	314,81%
Transacciones	1.000.000	3.200.000	320%
Visitas Sitio Público	61.000.000	328.000.000	537,7%

*Tabla Cap1_A, Proyección anual de **BECH***

Además se espera que la nueva versión corrija los principales problemas detectados en la actualidad en el portal de internet, los cuales se pueden reflejar de la siguiente forma:

- El diseño de la aplicación actual permite el crecimiento funcional sólo a través de seguir generando páginas de código con ***ASP*** y ***HTML***, junto con componentes aislados e integrados al resto de la aplicación en forma arbitraria, lo cual hace que la aplicación se haga cada vez más compleja dificultando su mantención, es decir, no tiene un diseño modular.
- Los formatos de presentación están insertos en el código de programación (***ASP*** y ***HTML***), por lo cual, cualquier modificación o inserción de formatos implica la intervención directa del código, haciendo más engorrosa su mantención.
- Existen múltiples páginas ***ASP*** con referencias y accesos a datos desde el servidor web, no existiendo un módulo único, estándar y atómico que se utilice para la comunicación con los servicios de negocio ligados a los productos de ***BECH***.
- No existe un estándar para el manejo de la mensajería de error, lo que ha provocado que la corrección de errores se encuentre inserta dentro del código fuente y que cualquier nuevo error detectado se corrija de la misma forma.

BECH ha aprovechado la tecnología internet para ofrecer por este medio sus servicios más tradicionales, entre ellos se cuentan consultas de saldo, transferencia de fondos, el pago de cuentas y servicios. ***BECH*** ha capitalizado sus esfuerzos en montar un sitio web, cuya primera versión salió a fines del año 1999 con un sitio público[2] que servía de vitrina para publicitar servicios, más una parte transaccional que al comienzo soportaba sólo consultas de saldos. Para el

[2] Sitio Público de BancoEstado: Sitio informativo al público en general referente a productos y servicios del banco.

año 2000 ya contaba con transferencias de fondos para personas naturales entre cuentas internas y el 2001 se incorporaba este mismo servicio a las personas jurídicas.

A medida que se han ido agregando funcionalidades, en el transcurso del tiempo se ha detectado un crecimiento sostenido del número de clientes que optan por estos servicios a través de internet, y es por esta razón que esta entidad ha tomado la decisión de cambiar su actual portal web, teniendo como meta un sitio transaccional que pueda soportar un aumento de carga de las magnitudes ya mencionadas en tabla *Cap1_A*, junto con solucionar los problemas actuales ya mencionados.

1.2 Importancia y naturaleza del estudio

La importancia que tiene el presente proyecto radica en los impactos que podría alcanzar:

a) Establecer un diagnóstico certero surgido a través de un análisis sobre el estado actual del sitio web de ***BECH***.

b) Servir como marco teórico al interior de ***BECH*** para la toma de decisiones al momento de contemplar un diseño a implementar.

c) En el caso de implementarse el diseño planteado existiría un cambio en el funcionamiento interno del actual portal internet de ***BECH***, se mejoraría el lanzamiento de nuevos servicios, rendimiento, disponibilidad de servicio y rapidez en la solución de problemas por este canal.

d) Al tener un sitio web más rápido, estable y eficiente se facilitaría la interacción entre el usuario y ***BECH***, aumentando su competitividad frente al resto de las instituciones bancarias que operan en Chile.

La naturaleza del presente estudio es teórica, ya que se basa en la

comprensión, evaluación y estudio de situaciones existentes, con el fin de proponer soluciones en miras de una posible implementación.

1.3 Descripción de la metodología

Hipótesis de Investigación.

*"Es posible obtener un modelo y una plataforma estándar para el sitio web de **BECH**".*

La ingeniería de software puede ser definida como "el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales[3], para completar esta definición se asocian paradigmas o ciclos de vida (cascada, espiral, prototipos, etc.) cuyo fin es la entrega de un software final que solucione una problemática determinada. Ninguno de los paradigmas clásicos se ciñe completamente al tenor de este proyecto de tesis, el cual esta orientado básicamente a tareas de análisis, diseño, investigación y evaluación, todas ellas con un resultado teórico, en las que se excluyen tanto las etapas de codificación como las de pruebas y por ende también se excluye la entrega de un software final (en nuestro caso un sitio web), por tal motivo, no es posible seguir un paradigma al pie de la letra.

Por lo anterior, se propuso trabajar bajo una metodología propia con fases claramente definidas y adecuadas a este proyecto, obtenidas a partir del estudio de los distintos paradigmas existentes, para fundamentar esta decisión es apropiado citar a Roger Pressman[4] : *"No hay necesidad de ser dogmático en la elección de los paradigmas para la ingeniería del software; la naturaleza de la*

---

[3] Roger S. Pressman, 3ra ed. McGraw -Hill 1995, "Ingeniería del Software, Un enfoque práctico". Pág 25.

[4] Roger S. Pressman, 3ra ed. McGraw -Hill 1995, "Ingeniería del Software, Un enfoque práctico". Pág 35.

*aplicación debe dictar el método a elegir. mediante la combinación de paradigmas, el todo puede ser mejor que la suma de las partes”***.**

Para definir la metodología adecuada al proyecto de tesis fue necesario que se contemplara una fase de levantamiento, para poder establecer claramente cuáles son los requerimientos del proyecto, además de contener una etapa de análisis en la cual se estudiaron a cabalidad cada uno de los principales problemas de ***BECH***, considerando también otros que puedan surgir en un levantamiento de la situación actual, dando una solución a cada uno de ellos, también la metodología contempla una etapa de esquematización de las soluciones encontradas a cada uno de las problemáticas detectadas en el levantamiento de requerimientos, una etapa de investigación, en la cual se estudió y analizó cada una de las ***plataformas*** o ***arquitecturas*** existentes en el mercado que cumplan con lo esperado por ***BECH***, una etapa de evaluación y selección de la plataforma más adecuada para dar soporte al nuevo sitio web, y finalmente la etapa en la cual se entrega un plan de implementación de las soluciones a los principales problemas de ***BECH*** bajo la ***plataforma*** seleccionada.

Basado en lo anterior, se detalla el ciclo de vida de este proyecto de tesis:

1. Levantamiento de la situación actual para la toma de requerimientos.
2. Análisis y solución teórica a los problemas principales

   En este punto se hará uso del concepto de reingeniería de software que según Arnold R. S. [1993], puede ser definida como “cualquier actividad que:

   a) Mejore nuestro entendimiento acerca del software
   b) Prepare o mejore el propio software, normalmente para incrementar su facilidad de mantenimiento, reutilización o evolución”.

El proceso de reingeniería sigue un conjunto de pasos que se realizan en el siguiente orden:

- **Análisis de código:** Debido a la experiencia y conocimiento en el área, no fue necesario realizar un uso exhaustivo de análisis dinámico, (proceso en el que se detectan defectos ejecutando el código), ni estático (evaluación que estudia la estructura del código sin ejecutarlo), sino más bien, se utilizaron estas herramientas cuando fue necesario ahondar en el conocimiento de un programa o conjunto de programas en particular.
- **Reestructuración:** *"En este paso el objetivo principal es intentar que el software sea más fácil de entender y de cambiar y menos propenso a errores por cambios futuros"*[5]. Para cumplir con este paso es necesario diseñar un modelo que clarifique el estado actual del sitio web de ***BECH***, en cuanto a reestructuración de datos se debiera identificar si existe sinonimia, (un mismo dato puede nombrarse de distintas formas en un sistema), dentro del sitio web y el que un mismo dato pueda definirse varias veces de forma diferente dentro del sitio, con el objetivo de considerar estos elementos y no repetirlos cuando se realice un nuevo diseño.
- **Ingeniería inversa:** ***"****Partiendo de un nivel de abstracción recrea modelos pertenecientes a niveles superiores, ya sea orientados a procesos o a datos"*[6]. Este concepto se utilizará para determinar a partir de los programas, conocimiento del sitio y de su implementación, una representación actualizada del sitio web.

---

[5] Chikofsky, E. "Reverse engineering and design recovery: a taxonomy", vol. 7, nº 1, Enero 1990.

- **Rediseño:** *"Consiste en consolidar y modificar los modelos obtenidos añadiendo nuevas funciones requeridas por los usuarios"*[7]. Esta etapa será evaluada en el punto cuatro de este ciclo de vida.

3. Implementación teórica de las soluciones a los problemas principales bajo la tecnología actual utilizada por ***BECH***.
4. Investigación de plataformas
5. Evaluación y selección de la mejor plataforma existente, a partir de resultados a obtener según malla de evaluación a especificar en capítulo seis.
6. Propuesta de un plan de implementación de soluciones encontradas para los principales problemas de ***BECH***, bajo la arquitectura mejor evaluada.

1.4 Objetivos

Los objetivos generales que pretende este proyecto son:

a) Establecer el modelo, arquitectura, implementación, funcionalidad, errores y falencias del actual sitio web.

b) Conformar y diseñar una solución para cada uno de los siguientes problemas principales de ***BECH***, con miras a obtener un diseño modular del sitio web, es decir:

- Un motor para la capa de presentación.
- Un módulo único para la obtención de datos
- Un estándar para el manejo de errores tanto en las páginas como para las componentes utilizadas.
- Un sistema de ***log*** que permita registrar cada una de las transacciones y errores que ocurran en el sitio web

---

[6] Mario G. Piattini. "Análisis y diseño detallado de aplicaciones informáticas de gestión", 2000.
[7] Mario G. Piattini. "Análisis y diseño detallado de aplicaciones informáticas de gestión", 2000.

c) Estudiar, evaluar y establecer la mejor arquitectura a usar en el sitio web, propuesto según resultado en malla de evaluación a definir en capítulo seis, que contemple como base, los objetivos generales de ***BECH***.

d) Diseñar la implementación de soluciones obtenidas a los problemas principales de ***BECH*** (punto b) en plataforma a proponer para este proyecto de tesis, entregando un plan de implementación a la nueva solución, obteniendo de esta forma el diseño modular de un nuevo sitio web.

1.5 Inclusiones y exclusiones

Debido a que ***BECH***, es una gran institución en cuanto al número de áreas administrativas que lo conforman, y a su importancia tanto como institución estatal, como económica, además de la cantidad de servicios y productos que ofrece, se hace enmarcar los ámbitos que se incluirán y los que se excluirán en el presente proyecto de tesis.

1.5.1 Inclusiones

En este punto se contemplan todos los aspectos que lleven a concretar los objetivos generales (**Ver punto 1.4**), a excepción de las exclusiones citadas a continuación.

1.5.2 Exclusiones

Quedaron fuera del desarrollo de este proyecto, los siguientes temas:

- Factores económicos. Debido a que en un sitio web como el de ***BECH***, hay

muchas variables en juego y debido al tenor teórico de este proyecto se hace prescindible un estudio técnico-económico ya que las decisiones a tomar descansan en un área gerencial y no exclusivamente en el área de informática, sin embargo se mencionarán precios cuando se estime necesario.

- Area de datos. El área de datos (ambiente ***Host IBM*** y ***SQL)***, en donde se almacena toda la información correspondiente al negocio, es decir, datos tan importantes como productos y servicios que posee cada cliente, puesto que no son de uso exclusivo de internet, sino que también acceden a ella otros canales de ***BECH*** a lo largo del país, tales como; cajeros automáticos, terminales en sucursales, ***WAP***, centros de contacto, entre otros. Además su administración no pertenece al área de informática, sino que al departamento de seguridad de la información de ***BECH***, por tales motivos se excluye de este proyecto de tesis.
- Configuración para Redes. Debido a que un sitio web no depende aplicativamente de la configuración de las máquinas en cuanto a tipo de red, topología u otro, se opta por excluir este ítem del proyecto de tesis.
- Firewall. Debido a que su administración depende del área de seguridad y que su implementación no se relaciona directamente con la funcionalidad de un sitio web, es que se excluye el estudio de un firewall.
- Software. Debido a lo teórico de este proyecto, se excluye la entrega de un software final.

Estos puntos serán mencionados en el proyecto, más no serán incluidos en la nueva versión del sitio web para ***BECH***.

CAPITULO DOS

LEVANTAMIENTO DE REQUERIMIENTOS

2. Levantamiento de requerimientos

El sitio web de ***BECH*** esta compuesto por un sitio público y un sitio privado:

- El sitio público no cuenta con restricciones para su ingreso, es decir, es de acceso universal, su objetivo es permitir que los clientes o potenciales clientes puedan interiorizarse en cuanto a los productos y servicios que ofrece ***BECH*** *(Ver lista de servicios que ofrece BECH en Anexo N°1)*.
- El sitio privado es de uso exclusivo para clientes del banco (personas naturales, empresas e instituciones públicas) e implica la autenticación de éstos. En este lugar el cliente puede realizar las acciones que le sean permitidas según los productos o servicios activos que tenga suscritos.

Tanto el sitio público como el privado conforman lo que se llamará sitio web transaccional de ***BECH***, o simplemente sitio web.

En el sitio web se considera importante el hecho de cultivar la lealtad de los clientes ya que esto tiene una relación directa con la rentabilidad a largo plazo, debido a que mientras mayor sea el tiempo de la relación con ellos mayor será la publicidad que éstos realicen a otras personas y por ende mayor la posibilidad de acercamiento de nuevos clientes. De esta manera, la atención al cliente se convierte en una herramienta competitiva y la ubicuidad de internet junto con su poder en la gestión de relaciones con el cliente es lo que hace que este canal ofrezca enormes oportunidades de aumentar la ventaja frente a la competencia, por esto a la hora de buscar una nueva versión del actual sitio web es necesario cumplir con ciertos factores críticos de éxito, de acuerdo a la percepción de los clientes y a la realidad que presenta el mercado, tales como, los que a continuación se enumeran:

1. Calidad de servicio, en cuanto a que se trate de una solución personalizable, ágil, eficaz, fácil de operar, intuitivo y asesor (proporcionando información útil y la capacidad de guiar al cliente).
2. Solución a través de servicios propios del negocio y otros que permitan marcar la diferencia con respecto a la competencia.
3. Apoyo tecnológico, manejando buenos niveles de disponibilidad (24 x 7), óptimo tiempo de respuesta, escalabilidad y seguridad en todos los niveles.
4. Soportar la imagen corporativa.
5. Alto nivel de seguridad: confianza del cliente en el uso del canal, garantizando privacidad de su información y disponibilidad ante sus requerimientos.
6. Disponibilidad y calidad de la información.

El actual sitio web de ***BECH*** cumple parcialmente con estos factores, por lo cual es necesario realizar un estudio acabado de lo que actualmente se dispone con miras a saber que lineamentos se deben seguir para optimizar la versión actual.

2.1 Estado actual del sitio web de *BECH*.

2.1.1 Hardware y software utilizado.

En la actualidad, el sitio web de ***BECH*** cuenta con ocho servidores encargados de dar servicio a todo el sitio web transaccional del banco. Cada uno de estos servidores cuenta con una configuración básica en cuanto a hardware que puede variar levemente de una máquina a otra y que se detalla a continuación:

- Marca - Modelo: COMPAQ DL 380
- RAM: 1024 Mb.
- Disco: 19000 Mb.
- CPU: 996x2.
- Tarjeta de Red: COMPAQ NC3163 Fast Ethernet.

En lo que a software se refiere, también se encuentra definida una configuración única para cada máquina, la cual se detalla a continuación:

- Windows NT 4.0
- ***IIS*** 4.0
- ***MTS***
- Service Pack 6.0
- ***XML*** 4.0
- ***ADO*** 2.1

Los servidores del sitio web funcionan dentro de una ***DMZ*** con ***balanceo de carga*** por hardware cuya función es distribuir las conexiones entre los diferentes servidores. La ***topología***, en términos generales, se puede esquematizar en el siguiente diagrama:

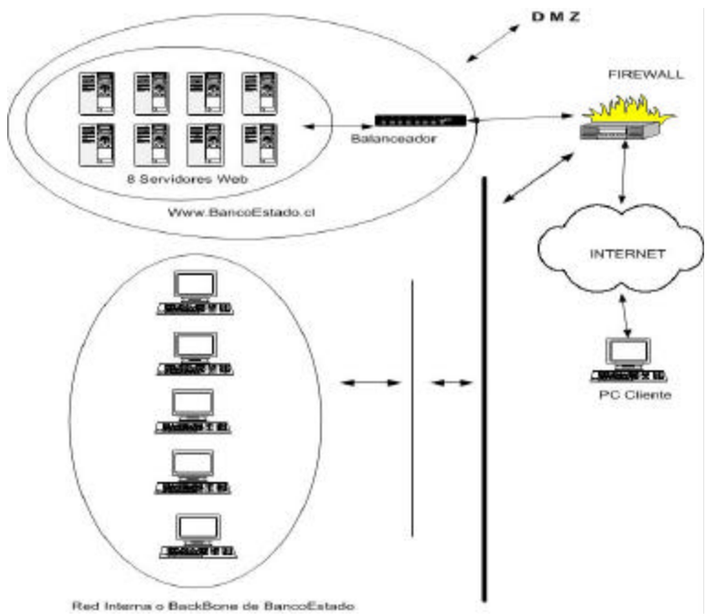


Figura Cap2_1, Topología de los servidores web de BECH

La ***DMZ*** corresponde a la ***zona desmilitarizada*** de ***BECH***, en ella se encuentran los ocho servidores web que dan soporte a internet más el balanceador de carga, la única protección en contra de accesos no permitidos a estos servidores la otorga un ***firewall*****.** La ***DMZ*** se encuentra entre internet y la red interna del banco.

El balanceador de carga se preocupa de distribuir las solicitudes llegadas al sitio web indicando que servidor debe resolver que solicitud según la carga de cada uno para tratar de que ningún servidor este recargado ni que se encuentren servidores ociosos.

2.1.2 Conexiones.

La cantidad de conexiones permitidas por servidor es variable, ya que depende de las capacidades de cada uno y del consumo de recursos en un momento determinado, pero se pueden entregar algunas aproximaciones obtenidas de la práctica, como se puede apreciar en la siguiente tabla:

	Conexiones promedio por servidor.
Horas Normales	70
Horas Peak	200
Máximo soportado	350

Tabla Cap2_A, Cantidad de conexiones por servidor en diferentes horarios

2.1.3 Arquitectura actual de *BECH*.

BECH ha tratado de ceñirse a ***DNA*** como arquitectura para su sitio web, sin conseguirlo completamente debido a la falta de estandarización en el desarrollo de sus aplicaciones.

2.1.3.1 *DNA*

La arquitectura de aplicaciones distribuidas de red (***DNA***) es una plataforma de Microsoft cuya funcionalidad es permitir la creación y distribución de sitios web empresariales. Las aplicaciones implementadas con ***DNA*** pueden mejorar el flujo de información hacia el interior y exterior de las organizaciones, son dinámicas y flexibles a los cambios que dictan los negocios, pudiendo ser fácilmente integradas con los datos y sistemas ya existentes. ***DNA*** suele describirse como un modelo de tres capas porque las aplicaciones web se segmentan en tres capas lógicas de funcionalidad:

- **Capa de presentación:** su objetivo es recibir consultas provenientes de los clientes desde internet y desplegar las respuestas en los navegadores de éstos. En esta capa se encuentran físicamente él o los servidores web, que al tener instalado ***IIS*** son capaces de proveer un sitio web, el cual proporciona el ambiente necesario para poder trabajar con ***HTML***, ***scripts*** y particularmente con ***ASP***.

  Las funcionalidades que ***IIS*** proporciona dentro de ***DNA*** son:

  - Sitio web profesional.
  - Entrega de contenidos confiable.
  - Soporte de múltiples sitios
  - Fuerte integración a la seguridad de Windows NT.
  - Fuerte integración a ***MTS*** y ***COM***.
  - Páginas script (***ASP***).
  - Monitoreo de desempeño.

  De todas las funcionalidades del ***IIS*** quizás la más importante sea que permite ejecutar archivos con extensión ***ASP***, los cuales pueden ser programados en distintos lenguajes, entre ellos ***JScript*** y ***VBScript***,

siendo este último, el lenguaje por defecto. Un archivo ***ASP*** es ejecutado en el servidor al momento de ser instanciado por una aplicación, si es requerido por más de un usuario conectado el ***IIS*** se preocupa de levantar ***hilos de procesos*** para cada solicitud, es decir, crea tantas instancias del archivo ***ASP*** como peticiones de clientes tenga. Este archivo es compilado en tiempo de ejecución, es decir, a medida que lo va ejecutando, el servidor va revisando la sintaxis del código en la página, éste archivo puede interactuar con la capa de negocios, para obtener a través de ella información desde la capa de datos.

- **Capa de negocios:** en esta capa se agrupan las librerías que ocupa la aplicación web para acceder a los datos, procesarlos y proporcionar una respuesta a quien los solicite. Corresponde a la lógica del negocio, y recibe requerimientos de la capa de presentación, concretamente desde un archivo ***ASP*** que instancia a una ***DLL,*** la cual solicita y procesa la respuesta de la capa de datos. Debido a que una misma ***DLL*** puede ser llamada por distintos archivos ***ASP*** en los mismos intervalos de tiempo, se levantan hilos de procesos cuando esto ocurre, en este caso es ***MTS*** el encargado de administrar estas situaciones.

- **Capa de datos:** en esta capa se agrupan los ***repositorios de información***, procedimientos y/o servicios de acceso a datos, indistintamente del tipo de base de datos. Recibe desde la capa de negocio solicitudes de información, las procesa y entrega el resultado a la librería solicitante para su posterior procesamiento.

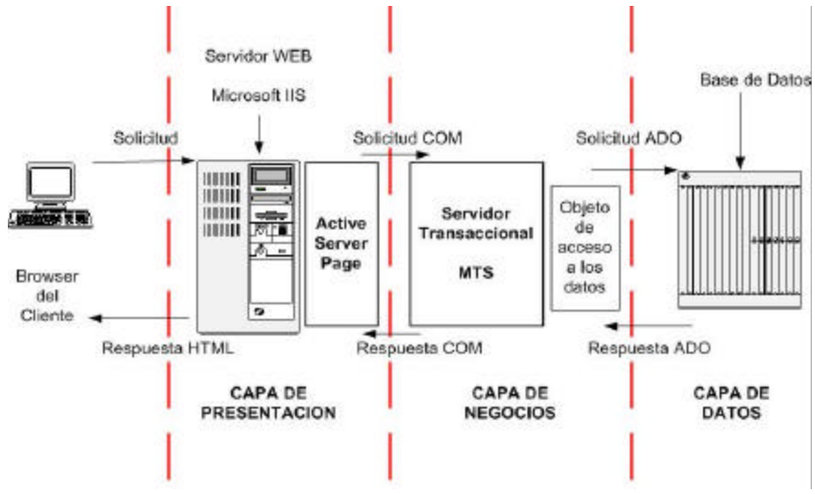


*Figura Cap2_2, Modelo **DNA***

Esta figura puede ser explicada por medio de un caso bastante común y que clarifica la interacción entre las tres capas, imaginemos que un cliente de ***BECH*** a través de su navegador solicita información de una saldo de una cuenta corriente. Esta solicitud es recibida por la capa de presentación del sitio web, específicamente por el ***IIS***, quien comienza a compilar en tiempo de ejecución el archivo ***ASP*** que corresponde a la consulta de saldo solicitada, este archivo instancia a una ***DLL***, perteneciente a la capa de negocio y existente en ***MTS***, indicándole la información que esta siendo requerida, la ***DLL*** interactúa con la capa de datos, para ello se conecta por medio de ***ADO*** al repositorio de datos en el que se encuentra la información solicitada, obtiene dicha información y comienza a preparar la salida en un formato manejable y esperado por el ***ASP***, quien al recibir respuesta a su petición, procesa y ordena la información recibida para enviársela al navegador solicitante para su despliegue en formato ***HTML***.

2.1.3.2 Implementación de *DNA* en *BECH*.

Como ya se ha mencionado previamente, ***BECH*** ha implementado sus aplicaciones web bajo ***DNA*** , objetivo que no se ha logrado completamente por los siguientes inconvenientes:

- Cuando se partió con el sitio web el estándar ***DNA*** aún no estaba masificado, por lo tanto las primeras aplicaciones que formaron el sitio web no incluían todas las buenas prácticas necesarias para una correcta implementación.
- No existían mecanismos o protocolos en ***BECH*** que permitieran fiscalizar la correcta implementación, en modelo de tres capas, de los desarrollos tanto internos (desarrollados por el área de internet) como externos (desarrollados por diferentes empresas outsourcing que prestan servicio al banco).
- La falta de compromiso de las gerencias involucradas en promover un estándar de desarrollo único para las aplicaciones web.

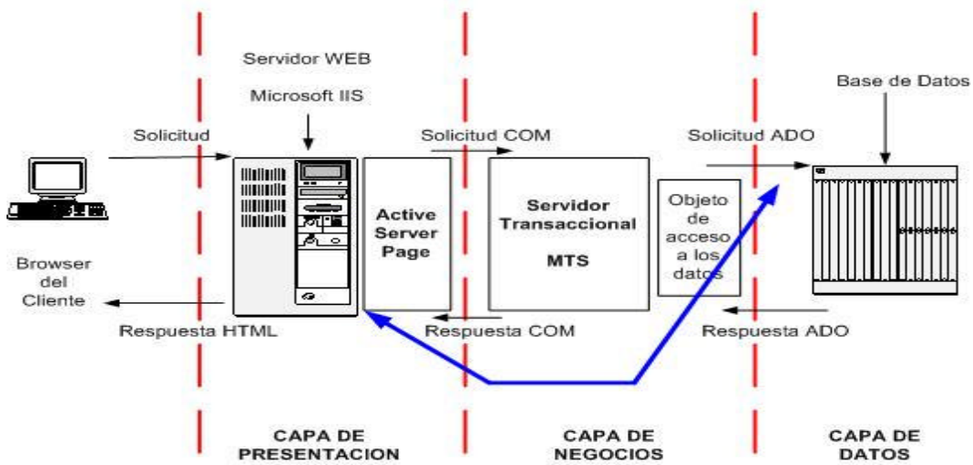


Figura Cap2_3, Modelo DNA implementado por ***BECH***

En la figura se presenta el modelo general del sitio web de ***BECH*** en la forma en que lo ha implementado siguiendo el estándar **DNA**. Es el mismo modelo que se presenta en la figura *Cap2_2*, con la diferencia que se instancia a la capa de datos sin pasar por la capa de negocios, es decir realizando la conexión directamente en el ***ASP***.

2.2 Detección de los principales errores y problemas del sitio web.

En el sitio web de ***BECH***, existen una serie de errores *(Ver anexo Nº2, Lista de reportes de error)* originados por la mala implementación del estándar ***DNA***, que sumado a los errores de desarrollo, instalación y administración de las aplicaciones, hacen de él un lugar con muchas falencias, dentro de las cuales las más importantes y motivo de estudio de este proyecto de tesis son la casi total ausencia de un:

- Motor de presentación
- Motor de negocios
- Manejador de errores
- Sistema de ***log***.

2.2.1 Motor de presentación

En el actual sitio web de ***BECH***, no existe un motor de presentación que permita manejar, en forma independiente del código de programación (***ASP***), los formatos de presentación corporativos del banco. Lo que ocurre es que los formatos de presentación se encuentran insertos en el código de las aplicaciones y cualquier modificación de éstos implica la intervención directa en el código de la aplicación, haciendo más engorrosa su mantención. Ejemplo de esto, fue el cambio de imagen corporativa del banco, que de ser Banco del Estado de Chile, se convirtió en Julio de 2001 en BancoEstado. Este cambio no

fue solo de nombre, además de eso se cambiaron todas las imágenes, logos y colores corporativos del banco, y al no existir un motor de presentación hubo que modificar cada página ***ASP*** *y* ***HTML*** que forma parte del sitio, la duración del proyecto fue superior a seis meses. Además del largo tiempo involucrado en el cambio de imagen, surgieron problemas luego de esto debido a que quedaron algunas páginas sin modificar y esto provocó que luego del cambio de imagen existieran en el sitio web páginas con el antiguo formato. De haber existido un motor de presentación, no hubiesen ocurrido este tipo de problemas y el tiempo de duración del proyecto se hubiese reducido notablemente.

Estos problemas persisten en la actualidad y en el caso que hubiese que realizar un nuevo cambio en el mismo esquema, el resultado actualmente sería muy similar.

2.2.2 Motor de negocio

BECH al no tener completamente normado el desarrollo de sus componentes (***DLL***) incurre en una serie de errores que no le permiten cumplir con los estándares del modelo ***DNA***. Además las falencias de sus desarrollos no estandarizados producen una serie de problemas en el sitio web que afectan directamente a los clientes.

Luego de analizar el estado actual del sitio web de ***BECH*** y revisar la lista de reporte de errores *(Ver anexo Nº2, Lista de reportes de error)* se encuentran las siguientes falencias en su actual motor de negocio:

- Es posible encontrar archivos ***ASP*** que consultan directamente a tablas ***SQL***, saltándose la capa de negocio. Esto implica que los ***ASP*** son más lentos de responder al cliente debido a que al ser compilados en tiempo de ejecución deben conectarse a la base de datos, realizar la consulta, recibir y ordenar los datos, aplicar reglas

del negocio pertinentes para finalmente desplegar en el cliente la respuesta como código ***HTML***.

- Existen archivos ***ASP*** que una vez que instancian a un objeto no lo destruyen, haciendo más lento el desempeño del sitio web. Lo mismo ocurre dentro del código de las ***DLL*** que al no destruir las instancias hacen más lenta la comunicación con y desde el sitio web.

2.2.3 Manejo de errores.

El manejo de los errores ocurridos en un sitio web es algo imprescindible a la hora de considerar un buen servicio internet, más aun si se trata de un sitio transaccional bancario. Es común encontrar en el portal de ***BECH*** algunos mensajes de errores como “En este momento no lo podemos atender, favor intente más tarde”, o “ha ocurrido un error”, lo que deja a los usuarios con la incertidumbre sobre qué es efectivamente lo que se encuentra mal, incluso a veces ni siquiera se muestra este tipo de mensaje, simplemente es el servidor web el que devuelve a cliente el código y la descripción del error ocurrido en la ejecución de alguna instrucción, por lo general son errores asociados a páginas dinámicas (***ASP***).

BECH presenta una serie de problemas con respecto al manejo de los errores que ocurren en sus páginas ***ASP***, en sus componentes (***DLLs***) y en sus servicios de datos tanto ***HOST*** como ***SQL***. Lo habitual es que no se maneje este tipo de excepciones y que se desplieguen mensajes a los clientes que no explican el motivo del problema, algunos ejemplos de estos mensajes se ilustran en las siguientes imágenes tomadas desde los servidores web que posee ***BECH***:

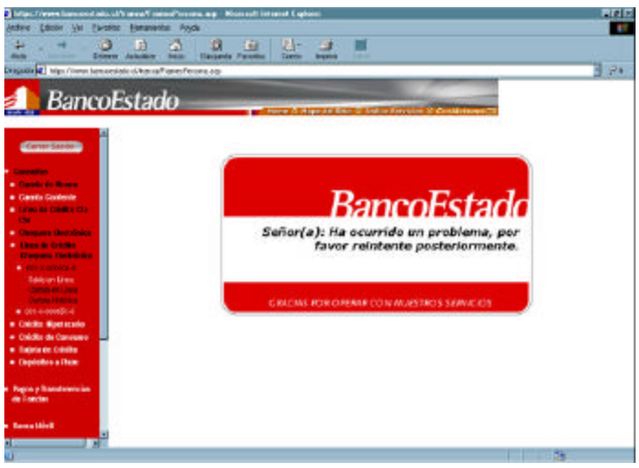


*Figura Cap2_4. Ejemplo de Mensaje de error desplegado a clientes en sitio web de **BECH**.*

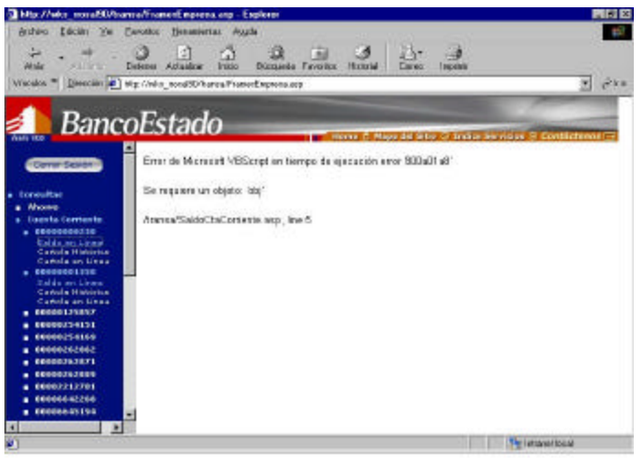


*Figura Cap2_5, Segundo ejemplo de mensaje de error desplegado a clientes en sitio web de **BECH**.*

Al no existir un manejo adecuado de errores, muchos reportes llegan directamente al área de internet de ***BECH***, lo que implica que esta área debe revisar el código fuente de la página, componente, servicio ***HOST*** o procedimiento almacenado en ***SQL*** para detectar la causa del problema y corregirlo.

Actualmente los usuarios y clientes del banco en internet al ver un mensaje de error que poco explica la causa y a la vez no indica ninguna acción

a realizar para salir de la situación, reportan el error telefónicamente a la mesa de ayuda del sitio web de ***BECH*** que opera las 24 horas del día los 365 días del año. Luego de recibir el reporte de error por parte del cliente, esta área entrega una plantilla con los errores reportados al área de informática para que sea esta la que inicie una investigación de la causa del problema, lo corrija y entregue un informe con todos los errores que se corrigieron y cuales fueron los motivos de que éstos se hayan producido, delegando en el área de internet la solución de todos los problemas que tienen que ver tanto con la misma área como también con el área de datos, seguridad y otras áreas**.**

En resumen, no existe un plan de contingencia, ni un manejador de errores que permita tanto a los desarrolladores, encargados y usuarios del sitio web, saber cuál es la causa del problema y qué acciones debieran realizar para solucionarlo.

2.2.4 Sistema *log*.

En los sitios web transaccionales y en especial aquellos que son bancarios es indispensable contar con un sistema de ***log*** que permita registrar cada uno de los movimientos de sus clientes, esto con el objetivo de salvaguardar los intereses del banco y el de los propios clientes. En este registro debe quedar la fecha y hora del movimiento, que fue específicamente lo que se realizó, las cuentas y montos involucrados además de los errores que pudieran ocurrir, con el fin de solucionar estos problemas con posterioridad. Un sistema de ***log*** permite realizar un estudio estadístico acerca de las horas con mayor número de conexiones, cantidad de usuarios que utilizan los servicios, productos más visitados, errores más frecuentes y una infinidad de información útil para la gerencia del banco.

Específicamente en ***BECH*** no se cuenta con un sistema integro y propietario de ***log***, actualmente solo existen registros de algunas transferencias para el caso del web transaccional, a parte de esto, el único ***log*** que utiliza el banco es el que le proporciona el ***IIS***.

Un buen sistema de ***log*** debiera contar con las siguientes características:

1. Un repositorio de datos que almacene los errores ocurridos en el sitio web.
2. Registrar cada transacción realizada por los clientes, para lo cual los siguientes datos son necesarios:
   - Identificación del cliente
   - Transacción que se invocó (servicio ***IBM HOST***, o procedimiento almacenado en ***SQL***)
   - Entrada y salida para dicha transacción .
   - Archivos que invocaron la transacción.
   - Datos de fecha y hora.
3. Un repositorio de datos con información sobre acciones a seguir para reporte de errores comunes

El contar con un completo sistema de ***log***, más allá de lo que ofrece hoy en día el ***IIS***, permitiría:

- Realizar una mejor gestión a la gerencia de operaciones de ***BECH***
- *A*gilizar el trabajo del área de internet
- Ser un respaldo para el banco y el cliente
- Monitorear los errores producidos en el sitio.

2.3 Otras falencias del sitio web.

Además de los errores principales, existen otras falencias en el sitio web, que al no depender directamente del área de informática o ser parte de las exclusiones de este proyecto de tesis, no fueron estudiados en forma exhaustiva.

El corregir estos errores o indicar pautas para su corrección, ayuda a robustecer aun más la nueva versión del sitio web que se pretende para ***BECH***. Los errores detectados tras el análisis del actual sitio web junto con el estudio de los reportes de errores entregados por la mesa de ayuda de esta institución son los siguientes:

- Administración deficiente de los códigos fuentes de las aplicaciones instaladas en el sitio web.
- No homogolización del sitio web en las máquinas de producción, ya que existen distintas versiones de las aplicaciones instaladas en ese ambiente.
- Diseños mal estructucturados de las bases de datos ***SQL*** y problemas con los servicios de datos ***IBM HOST***.

2.3.1 Control de código fuente

Uno de los grandes problemas que existe en el sitio web de ***BECH***, es no poder mantener un correcto control acerca de las versiones de sus códigos fuentes, tanto de páginas, componentes y servicios de datos. Habitualmente ocurre que en un grupo de desarrollo los integrantes de éste coincidan en el desarrollo de aplicaciones que usan código fuente en común, lo que hace que cada uno vaya desarrollando según su necesidad sin considerar que la misma fuente puede estar siendo modificada por otra persona, esto implica que una aplicación puede terminar bien en desmedro de la otra.

Otro resultado de la ausencia de un adecuado control de fuentes es que pueden existir en un momento dado distintas versiones de un mismo código, lo que dificulta a los desarrolladores al momento de querer hacer alguna modificación, debido a que algunas versiones pueden tener cambios que no pueden ser puestos en los servidores de producción al formar parte de pruebas en desarrollo, y a la vez contienen una modificación vital para el sitio de producción, esta ambigüedad solamente se soluciona manteniendo un adecuado control de las versiones existentes de los códigos utilizados en el sitio web.

2.3.2 Unificación de ambientes.

Al existir un sitio web que posea varios servidores dando soporte a los usuarios y clientes que lo acceden, es común que ocurran problemas relacionados con la existencia de diferentes versiones de las aplicaciones en los distintos servidores, esto se debe a que no existe un control de las versiones instaladas en producción, es decir, los encargados de administrar los servidores web no replican en todos los servidores las mismas versiones de las aplicaciones del sitio.

Un gran porcentaje de los errores detectados y corregidos en el sitio web de ***BECH*** corresponden a que en el servidor donde ocurrió el problema no se encontraba la versión adecuada de la aplicación que presentó el inconveniente. Si existiera una unificación en el ambiente de producción, es decir, que se controle que cada aplicación sea instalada correctamente en cada servidor (las mismas versiones de ésta), no ocurrirían este tipo de errores, esto mismo se debiera aplicar para los ambientes de desarrollo y pre-producción.

2.3.3 Problemas con las bases de datos

BECH presenta una serie de problemas relacionados con sus bases de datos y los servicios utilizados para acceder a ellas, estos problemas se presentan tanto en sus bases ***SQL*** como en las de ***IBM HOST***.

En el primer caso existen bases de datos que no están bien normadas, no son consistentes con la información que poseen y sus servicios no realizan sus tareas correctamente ya que no poseen manejador de errores y existe un gran número de ellos que realizan actividades parecidas (accediendo a las mismas bases y tablas).

En el caso de las bases ***IBM HOST*** el problema principal radica en un gran número de servicios que poseen para realizar las mismas funciones, ejemplo de ello es que por cada consulta de saldo de cada producto se utiliza un servicio distinto y lo mismo ocurre para el caso de las cartolas en línea.

La sobrepoblación de servicios de acceso a datos, su nulo manejo de errores, el que realicen tareas muy parecidas, sumado a la malformación y la poca consistencia de las bases de datos, hacen que el sitio web de ***BECH*** no sea muy eficiente en el manejo de datos.

CAPITULO TRES

ANALISIS Y SOLUCION TEORICA

3. Análisis y solución teórica

Cada uno de los principales problemas de ***BECH***, vistos en el apartado 2.2, son resueltos en forma teórica en este capítulo. La idea básica es modelar cada solución considerando que son en parte un protocolo para diseñar y mantener el sitio web, independiente de la tecnología o plataforma en la que descansen dichas soluciones.

3.1 Motor para la capa de presentación

3.1.1. Que se entiende por motor para la capa de presentación

Se entiende por motor para la capa de presentación a una interfaz única para el despliegue de datos en el cliente. Este motor consiste en una componente y bases de datos que unidas en un mismo modelo cumplen dos funciones básicas:

1. Solicitar y recibir información desde la capa de negocios.
2. Aplicar formatos y diseños a la información obtenida de la capa de negocio y posterior envío de ésta al navegador del cliente.

La justificación para tener un motor para la capa de presentación radica en que de esta forma es más simple comprender, mantener y administrar todos los formatos de la capa de presentación.

3.1.2. Objetivos del Motor de Presentación

Los objetivos que debe cumplir el motor para la capa de presentación son :

- **Escalabilidad.** Debe ser flexible y capaz de crecer en el tiempo, permitiendo que se agreguen, eliminen o modifiquen funcionalidades en forma paramétrica.

- **Navegación y diseño coherente del sitio.** Debe permitir una navegación uniforme, en el sentido de mantener el mismo esquema a lo largo del sitio, permitir la personalización de los mensajes, ofertas de productos o servicios. En cuanto a diseño, debe mantener el tipo de fuente, colores y textos en todo el sitio, así como también desplegar las imágenes correspondientes.
- **Modularizar los cambios estéticos.** Permitir cambios estéticos según tipo de producto, tipo de cliente u otro tópico, con tan solo cambiar ciertos parámetros.

3.1.3. Entidades relacionadas con el motor de presentación

El motor de presentación desde un punto de vista general se comunica a través del servidor web con el componente de la capa de negocios para obtener la información deseada, posteriormente aplica diseños y da formato a esta información para generar la respuesta al cliente. El motor de presentación esta físicamente dentro de la capa de negocio y la capa de datos, un esquema general de lo que se espera como modelo para el motor se presenta en la siguiente figura:

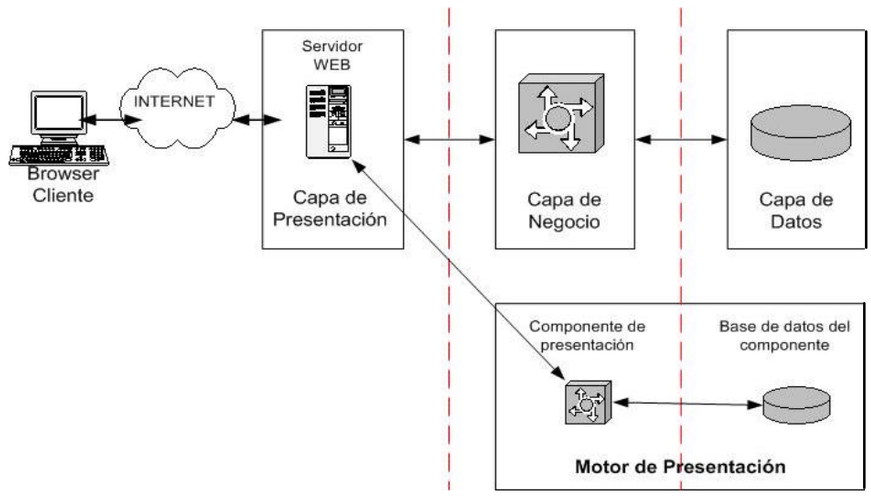


Figura Cap3_1, funcionamiento esperado para el motor de presentación.

En esta figura se modela el motor de presentación como una entidad que necesita del servidor web para su funcionamiento, pero que es independiente en cuanto a la lógica de negocio, ya que en este motor se define que diseño utilizar para desplegar la información solicitada y como trabajar los datos obtenidos de la capa de negocio antes de devolver la información al navegador del cliente solicitante.

El motor para la capa de presentación contempla el despliegue de la información tanto del sitio privado como del sitio publico bajo un mismo esquema genérico.

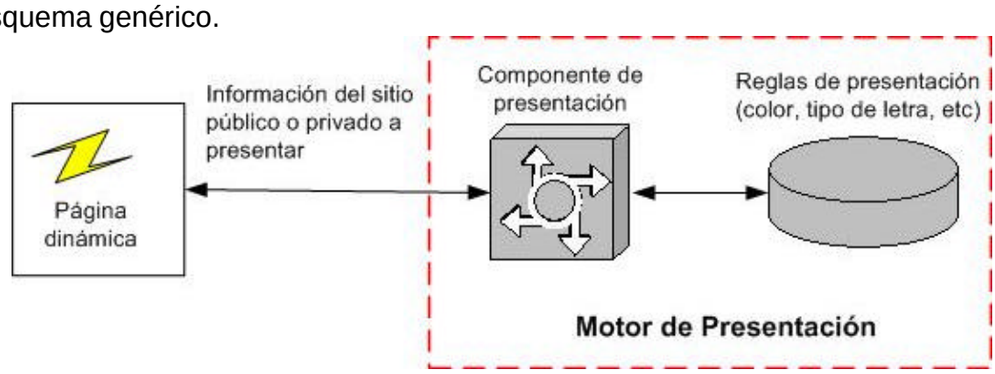


Figura Cap3_2, Funcionamiento del motor de presentación en el sitio web.

En la figura *Cap3_2,* se muestra el funcionamiento del motor de presentación, tanto para el sitio privado como público, para el primer caso los parámetros de entrada al motor son los productos y servicios asociados, y para el segundo pueden ser noticias, despliegue de datos financieros, simuladores, entre otros.

3.1.3.1. Funcionamiento del motor de presentación

El motor de la capa de presentación solicita información a la capa de negocios, la cuál por medio de componentes llama a los servicios de datos necesarios que se encuentran en la capa de datos y devuelve la información al motor de presentación, el cual da diseño y formato a la información requerida antes de devolverla al cliente que la solicitó.

3.1.3.2. Funcionamiento en el sitio privado

En el sitio privado existen dos conceptos básicos a considerar: los productos y los servicios. Se entiende por productos a aquellas entidades tales como cuentas corrientes, cuentas de ahorro, créditos hipotecarios, etc. y por servicios a aquellos que (por lo general) se aplican a un producto, es decir saldos, cartolas, cupones de pago, etc, para clarificar el concepto se puede hacer una analogía entre objetos y atributos, donde los productos serían los objetos y los servicios los atributos. Debido a la existencia de estas entidades, es que se debe considerar un modelo según funcionalidad, con el motivo de generar ***plantillas*** estándares en cuanto al contenido que deben desplegar como respuesta al cliente, por ejemplo no vale la pena tener varias páginas dinámicas para desplegar los saldos de cada uno de los distintos productos con que cuenta un sitio web financiero como el de ***BECH***, sino que contar con una sola que despliegue el saldo de cualquiera de los productos ofrecidos por el sitio web, siendo el motor de presentación el encargado de presentar en diferentes formatos la información dependiendo del producto y servicio deseado.

Por lo mismo debe existir un tipo de ***página dinámica***, para cada tipo de servicio, la cual debe aplicar una ***plantilla*** distinta según el producto con el que sé este trabajando en determinado momento.

3.1.3.3. Funcionamiento en el sitio público

Como la función básica de un sitio público es desplegar noticias e información general, el contenido a desplegar debe tener diseño y tiempos de vida configurables (una noticia puede estar publicada una semana en cambio el valor de la U.F. varia día a día)

El motor de presentación considerará la configuración de distintos tipos de diseños para el sitio web (sin ***frames***, con dos o más ***frames***), junto con otros parámetros tales como: imágenes de fondo, textos, colores, etc. según el contexto

3.1.4. Entradas y salidas del motor de presentación

A continuación se definen las relaciones entre las entidades que interactúan con el motor de presentación, cabe recordar que toda relación pasa directamente por el servidor web, quien hace el papel de comunicador entre las entidades y el motor de presentación.

3.1.4.1. Motor de presentación – Páginas dinámicas

Las páginas dinámicas contienen el código que el servidor web compila, dentro de este se encuentra el nombre del producto y servicio que se esta solicitando, con estos datos el motor de presentación debe comunicarse con la capa de negocio para obtener la información deseada del área de datos, además debe saber interpretar la información recibida, y obtener desde el repositorio de ***plantillas*** la correspondiente para presentar la información solicitada al cliente.

3.1.4.2. Motor de presentación – Capa de negocio

El motor de presentación debe indicar a la capa de negocio que servicio debe instanciar, además debe entregar todos los parámetros de entrada requerido por ese servicio, y una vez que reciba la información, aplicar la ***plantilla*** asociada al tipo de producto y servicio, obtenida del repositorio correspondiente.

3.1.5. Modelo Final

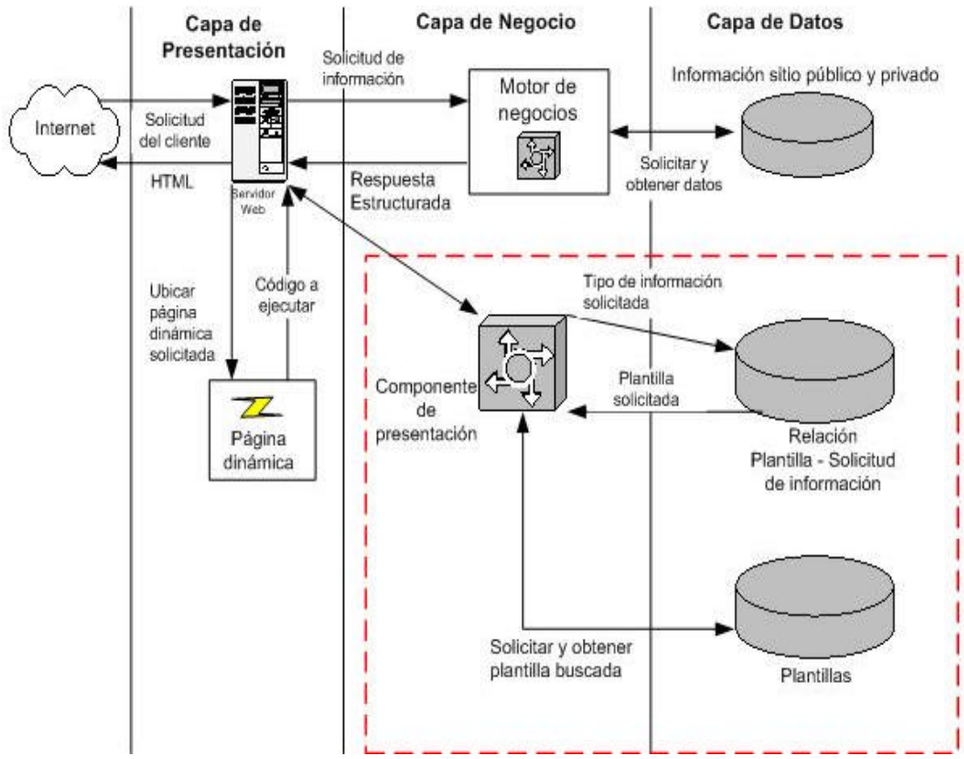


Figura Cap3_3, Motor para la capa de presentación (cuadro inter-lineado)

En la figura es posible ver el funcionamiento del motor de presentación:

- **Componente para motor de presentación:** debe tener la lógica para acceder a los repositorios de datos y obtener la relación entre el producto ó contenido a desplegar y la correspondiente ***plantilla*** con el diseño y formato a procesar.
- **Repositorio de plantillas:** Contiene información de la ubicación física del archivo con formato y diseño a aplicar sobre la información obtenida por el motor de negocio.
- **Relación Plantilla – Información solicitada:** contiene básicamente la asociación entre que ***plantilla*** aplicar, según el tipo de información que se este solicitando.

3.2 Motor de negocios

3.2.1 Que se entiende por motor de negocios

Se entiende por motor de negocios a una interfaz única para la obtención de la información que maneja el sitio web. Este motor esta formado por un ***componente*** y un grupo de bases de datos que unidas en un mismo modelo cumplen dos funciones básicas:

- Consultar a la capa de datos por información de negocio según requerimiento del cliente llegado por medio del servidor web.
- Aplicar lógica a los datos obtenidos, generando una ***respuesta estructurada***, que por medio del servidor web será entregada al motor de presentación.

La importancia de contar con un motor de negocios es básicamente poder modificar la lógica del sitio web sin tener que realizar mayores modificaciones en la capa de presentación y en su motor, mientras mejor sea el motor de negocios menor será la lógica a implementar en las ***páginas dinámicas***.

3.2.2 Objetivos del motor de negocios

Los objetivos del motor de negocios son:

- **Unificar todas las conexiones a servicios de datos.** Es decir que exista un componente capaz de conectarse y realizar consultas a las bases de datos independientemente del formato en el que almacenen la información.
- **Unificar la lógica de negocio.** Con esto se pretende crear un modelo formado por una o más componentes que sea capaz de

obtener la información solicitada, procesarla y darle un formato estructurado.

- **Escalabilidad**. Debe contar con la capacidad de permitir un crecimiento en el tiempo que no implique agregar mayor lógica en las ***páginas dinámicas***.
- **Mantención.** Debe ser fácil de mantener, sin que esto implique realizar cambios mayores en la capa de presentación.

3.2.3 Entidades relacionadas con el motor de negocio

Básicamente el motor de negocios descansa entre :

- La capa de negocios (donde están las componentes)
- La capa de datos (donde están los repositorios de datos que permiten el funcionamiento del motor)

A continuación se presenta un modelo que gráfica el lugar ocupado por el motor de negocios dentro del modelo de tres capas.

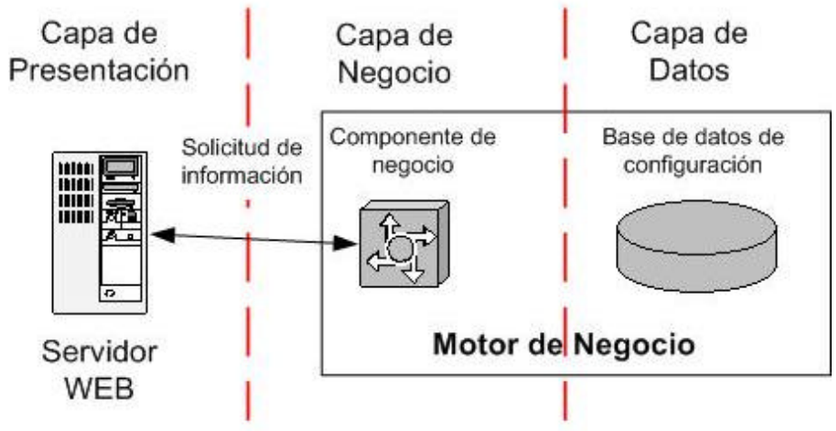


Figura Cap3_4, Ubicación del motor de negocio en un modelo de tres capas

El componente debe realizar:

- Conexión y ejecución de consultas a las bases de datos con la información de negocio.
- Incorporar la lógica de negocio a los datos obtenidos.
- Conexión y ejecución de consultas a las bases de datos de configuración.

Los repositorios deben poseer datos parametrizables que permitan la asociación entre información solicitada y servicio de datos además de la configuración para poder ordenar los datos en un formato de ***respuesta estructurada***. Entre los datos que almacenan los repositorios se encuentran:

- Código de producto-servicio y el nombre del servicio de datos asociado.
- Formato a aplicar para generar una respuesta estructurada según tipo de producto-servicio determinado.

3.2.4 Entradas y salidas del motor de negocio

La relación entre las entidades que interactúan con motor de negocio se pueden definir de la siguiente forma:

3.2.4.1 Motor de negocio – Página dinámica

El motor de negocio cuenta con una componente cuya función es recibir desde la página dinámica una solicitud y contestar con una respuesta estructurada, esto último por medio de los siguientes pasos:

- Determinar que tipo de servicio de datos a utilizar, ya sea que se trate del sitio público o del privado.

- Obtener la información solicitada por medio del servicio de datos ya obtenido en el paso anterior, utilizando una componente para obtención de datos.
- Dar un formato de respuesta estructurada a la respuesta proporcionada por el servicio de datos utilizado.

3.2.4.2 Motor de negocio – Capa de datos

El motor de negocio cuenta con una componente encargada de llamar a los servicios de datos, estos pueden corresponder al sitio privado o al público, en cada caso la finalidad es ocupar esta componente como único interlocutor válido entre la capa de negocio y la capa de datos.

3.2.5 Modelo Final

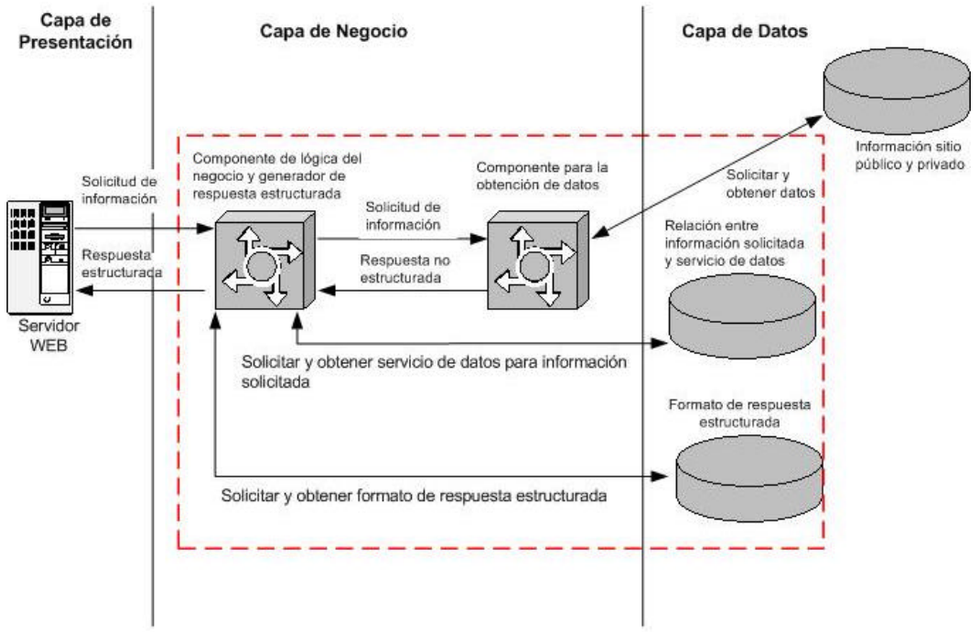


Figura Cap3_5, Motor para la capa de negocios (cuadro inter-lineado)

El funcionamiento del motor de negocios, se puede explicar de la siguiente forma: cuando llega una solicitud por parte del servidor web al componente de lógica del negocio, éste debe ir al repositorio de relación entre

información solicitada y servicio de datos, obtener de dicho repositorio el servicio de datos correspondiente a la información solicitada, luego utilizando el componente para la obtención de datos acceder al repositorio de información correspondiente en la capa de datos, obteniendo así la información deseada. Una vez obtenida la información solicitada, el componente para la obtención de datos devuelve la información al componente de lógica del negocio en un formato no estructurado, para estructurar dicha información se debe acceder al repositorio de formato de respuesta estructurada y obtener el formato asociado a la información solicitada, con este formato el componente de lógica del negocio estructura la información y la devuelve al servidor web.

3.3 Manejador de errores o excepciones

A menudo ocurren errores en las aplicaciones web, éstos errores pueden ser producto de un mal manejo de información, una base de datos cerrada, la inexistencia de algún componente, un mal cálculo numérico, una respuesta inesperada que entregue un servicio de datos o un ***string de conexión*** errado hacia una base, entre otras causas. Estos errores producidos en la ejecución de un programa son producto de excepciones que se generan en tiempo de ejecución. De éstos errores, como de la forma de manejarlos dentro de las aplicaciones y de la generación de un estándar para su manejo dentro de ***BECH***, tratarán los siguientes puntos.

3.3.1. Qué es una excepción

Una excepción es un evento que ocurre durante la ejecución de un programa y que interrumpe el flujo normal de ejecución. Es decir, es cualquier problema que se presenta durante la ejecución del código de una aplicación que obliga al compilador de ésta a detener abruptamente su ejecución.

3.3.1.1. Qué se entiende por manejador de errores

Un manejador de errores o excepciones es el encargado de evitar que cualquier error que se produzca dentro de cualquier aplicación que se ejecute en el servidor, termine con la entrega de información incomprensible para el usuario que la solicita, así como también para los administradores del sitio web que deben corregir el problema presentado en la aplicación. Su misión es devolver un mensaje entendible al usuario que solicita la información, dando aviso a éste del error y que a la vez sea comprensible para los administradores del sitio.

El manejador de errores va ligado completamente al lenguaje de programación con el cual se construyó la aplicación, es necesario que cada una de las partes que componen el sitio web (páginas, componentes y servicios de datos) contenga un manejador de errores, para así poder determinar claramente donde se produjo el error y realizar los pasos correspondientes para solucionarlo. Al crear un mecanismo de manejo de excepciones se deben establecer algunos pasos esenciales:

- Activación del controlador de errores.
- Escribir una rutina de tratamiento del error
- Crear un escape para la rutina

Además de esto, el manejador de excepciones debe ser simple de usar y entender, tener una separación del código para el manejo de las excepciones del código normal y tener un tratamiento uniforme de las excepciones.

3.3.2. Objetivo del manejador de errores

El objetivo de tener un manejador de errores dentro de un sitio web es:

- Evitar que las aplicaciones interrumpan abruptamente su desarrollo

- Evitar que el usuario que solicita información reciba mensajes poco entendibles y que en nada ayudan a resolver el percance
- Permitir a los administradores del sitio poder identificar rápida y exactamente el motivo del error, para que sea corregido lo antes posible.

Es necesario que el mensaje que se entregue al cliente le indique a éste los pasos a seguir para solucionar el problema o bien a quien recurrir en caso de que alguna aplicación no funcione, además es necesario que sea el mismo manejador de excepciones el encargado de registrar un mensaje real del error ocurrido en un repositorio, para que sean los administradores del sitio los encargados de resolver el problema según el reporte generado por el manejador y si la causa del error lo amerita, enviar una notificación al área encargada de la administración o supervisión de la causa del problema, como por ejemplo los encargados de las bases ***SQL*** y de los servicios de datos, entre otros.

La información que el manejador de errores entrega al cliente debe estar almacenada y definida claramente en algún repositorio, en él se debe tener un mensaje asociado a cada tipo de error, los pasos a seguir para resolverlo, si amerita el envío de una notificación a los administradores o responsables de la aplicación y los nombres de éstos.

3.3.3. Entidades relacionadas con el manejador de errores

El manejador de errores debe ser único para todas las partes involucradas dentro del sitio web, es por ello que debe interactuar y dar soporte a cada una de ellas, es decir, debe ser el encargado de manejar los errores que ocurran en las componentes y páginas dinámicas.

El manejador de excepciones del sitio web debe resolver los errores presentados en dos entidades (componentes y páginas) manejando todas sus excepciones en el mismo módulo de manejo de errores, además el módulo debe interactuar directamente con repositorios de datos, en uno de ellos almacenar todos los errores ocurridos en cualquiera de las dos entidades ya mencionadas y en el otro tener la lista de mensajes a entregar a los clientes, lista de administradores y responsables, acciones a seguir y si corresponde el envío de alguna notificación. También debe hacer uso de una aplicación para el envío de mensajería, con esto, es el mismo manejador de errores el encargado de iniciar la solución de los problemas, ya que se encarga de enviar la información del error a los administradores del sitio o a los encargados de las áreas que presentan problemas.

3.3.4. Entradas y salidas del manejador de errores

La relación entre las dos entidades que interactúan con el manejador de errores se puede definir de la siguiente forma:

3.3.4.1. Manejador de errores – Páginas dinámicas

Es probable que al manipular información dentro de una página dinámica ocurra alguna excepción, como por ejemplo que se ingresen datos con formato errado (texto en vez de números), que falte información, que se produzcan errores de cálculos o inclusive errores en las validaciones hechas en ***scripts***, etc. Estos errores pueden detener la ejecución de la página y enviar al cliente un mensaje poco entendible y que en nada ayuda a su solución. Es aquí donde debe intervenir el manejador de errores, y lo debe hacer de la siguiente forma: la página debe tener dentro de su codificación estipulada que es el manejador de errores el encargado de recibir cada excepción que ocurra dentro de la

ejecución del programa, así cada vez que ocurra alguna excepción de cualquier tipo sea el manejador de errores el encargado de interpretar dicha excepción, dejar registro de ella en un repositorio, enviar a pantalla un mensaje claro para el usuario que explique fácilmente la razón del problema y los pasos a seguir, para ello debe consultar dentro de una lista el mensaje a desplegar según el tipo de error y la fuente de éste, es decir, que tipo de error ocurrió y el lugar donde se originó (página o componente), también el manejador debe verificar en la misma lista si corresponde o no enviar una notificación a los administradores y/o encargados de las áreas que presentan problemas así como también debe generar un reporte de errores el cual debe ser despachado al área de informática para su posterior resolución. Con esto se lograría que el primer reporte de errores provenga del propio manejador y no de los reportados por los clientes a la administración del sitio, obteniendo con ello un sitio web mucho más proactivo.

3.3.4.2. Manejador de errores – Componentes

La mayoría de los errores tienden a ocurrir dentro de los componentes de datos o negocio que posee un sitio web, principalmente debido a que estos componentes son en si módulos separados que son creados por diferentes personas y que deben interactuar entre ellos, esto se refleja en la condición de que el autor de un componente puede detectar errores durante la ejecución pero no siempre sabe como corregirlos de la mejor manera, y el usuario de una componente posiblemente sabe como corregir el error pero difícilmente puede detectarlo.

La idea de centralizar el manejo de todos los errores que ocurran en cualquier componente, ya sea de negocio o de datos, sin importar la persona que la haya codificado es lo ideal si se trata de un sitio web, para que esto

ocurra cada una de las componentes debe tener indicado que es el manejador de errores el encargado de recibir cada una de las excepciones que ocurran en la ejecución de ellas, con esto se logra que sea el manejador el encargado de procesar el error, generar el reporte correspondiente, entregar un mensaje claro para quien solicita información de la componente, almacenar en un repositorio toda la información correspondiente al error y enviar un mensaje a los encargados del sitio y/o de la aplicación si la causa del error lo requiere.

3.3.5. Modelo Final

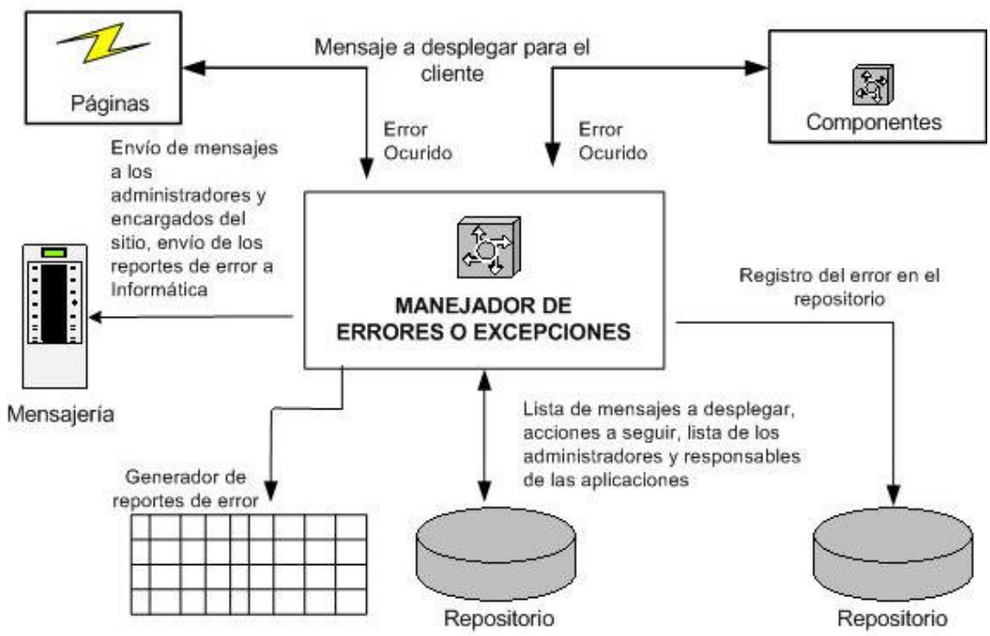


Figura Cap3_6, Modelo para el manejador de errores

En la figura *Cap3_6* se aprecia la interacción del manejador de errores con las dos entidades que lo utilizan directamente, componentes y páginas, ambas entidades envían al manejador la información de la excepción ocurrida durante su ejecución, esta información es recepcionada por el manejador quien inmediatamente la deja grabada en un repositorio de datos, luego de ello con la información recibida de la entidad que lo instancia revisa en otro repositorio cual

es el mensaje que debe enviar a la entidad y las acciones a seguir, dependiendo de estas acciones puede o no enviar un mensaje a los administradores del sitio y/o a los encargados de las aplicaciones, también debe generar un reporte de errores en un repositorio temporal y dependiendo de la frecuencia programada, enviar una lista con todos los errores reportados al área de informática para que se gestione su solución. Al ser el manejador de errores el primero en iniciar el proceso de solución de éstos, se obtiene un sitio web más proactivo.

Es necesario que exista una aplicación que administre al manejador de errores, la funcionalidad de esta aplicación sería la de agregar y/o modificar los mensajes entregado a los clientes, las acciones a seguir, los nombres de los encargados y si corresponde el envío de mensajes a estos últimos. Desde esta misma aplicación se deben ver la totalidad de los errores generados, con esto los encargados del sitio o de la aplicación con problemas podrán enterarse más técnicamente del origen del error.

3.4 Sistema *log* transaccional

Todos los servidores web generan archivos ***log***, en los cuales almacenan toda la información generada por un usuario que accede al sitio web, es decir, almacena todos los movimientos generales que realiza un cliente, como lo son la fecha y hora de conexión, la dirección ***IP*** del cliente, la página accesada, las imágenes y archivos descargadas, los bytes transferidos desde el servidor al cliente y la dirección en la que se encontraba el cliente antes de realizar la petición entre otros datos. Toda esta información generada por el servidor en archivos ***log*** no es relevante si hablamos de un sitio web transaccional, debido a que, lo realmente importante de almacenar en un caso como este es toda la información relacionada con las transacciones que realizan los clientes, esto

como medida de seguridad para ambas partes y además como sistema de gestión para los administradores del sitio. Teniendo un sistema de ***log*** transaccional se puede contar con una herramienta que da seguridad a los clientes ya que cada una de sus transacciones quedará registrada y podrá consultarla en cualquier momento, además los administradores del sitio pueden identificar a través de este sistema los productos más solicitados, las transacciones realizadas con mayor frecuencia, los horarios de máxima conexión, entre otros datos relevantes.

3.4.1. Que se entiende por sistema *log* tradicional

Un sistema de ***log*** tradicional es aquel que genera el propio servidor web, el cual por cada "golpe" (*hit*) que se produce en el sitio genera una línea en un archivo ***log***, es decir, cada vez que un cliente realiza una acción en el servidor web (acceder a una página, descargar una imagen, etc.) se añade una línea al archivo ***log***. Cada una de estas líneas contiene cierta información que varía según el formato del archivo ***log*** pero que normalmente suele contener datos como la dirección ***IP*** del cliente, la hora y fecha en las que se produce la acción, la operación que realiza el cliente, el archivo sobre el cual se realiza la operación, los bytes que se transfieren en la operación, etc.

Existen diversos tipos de archivos ***log***, entre ellos están:

- **Extended Common Log File Format :** Este formato es utilizado por bastantes servidores web, como por ejemplo el Apache. A diferencia de los otros dos formatos que utilizan la primera línea para indicar el tipo y versión del formato, así como los campos que tiene cada línea. En este formato, en cambio, la primera línea ya pertenece a los datos de la primera acción del archivo. Este tipo de formato consta de nueve campos separados entre si por un espacio.

- **Microsoft Internet Information Services (IIS):** Este formato en la primera línea indica el tipo de archivo ***log*** (***IIS***), la segunda la versión de este tipo de archivo, la tercera indica la fecha y hora en la que se ha creado el archivo y en la cuarta línea se aparece el nombre de los distintos campos que posee cada línea del archivo. En este tipo de archivos los campos van separados por tabulados y no por espacios como en el caso anterior.
- **Microsoft Internet Security and Acceleration Server 2000 (ISA):** Al igual que el ***IIS***, este formato también presenta un encabezado con el tipo de archivo, versión, fecha y campos de cada línea.

3.4.2. Que se entiende por sistema *log* transaccional

Un sistema de ***log*** transaccional es aquél que permite registrar cada uno de los movimientos que realiza un cliente en un sitio web financiero o comercial (compra y venta). A diferencia del sistema de ***log*** tradicional, el transaccional se encarga de almacenar en un repositorio todas las transacciones realizadas por el cliente:

- Un pago de servicio, transferencia de fondos, consulta de saldos, etc. en el caso de un sitio web financiero
- los datos asociados a una compra o venta (cuenta de cargo, monto de compra, fecha y hora, etc.) en el caso de un sitio web comercial.

Es absolutamente necesario contar con un sistema de ***log*** transaccional en un sitio web de una institución financiera, debido a que es necesario registrar cada una de las operaciones que realice el cliente en el sitio. Al realizar este registro en un repositorio se brinda seguridad a ambas partes, ya que debe haber un respaldo electrónico de cada transacción realizada sobre el sitio web. Este historial de transacciones realizadas debe ser accequible para el cliente en

el momento que lo estime conveniente. Además el tener toda esta información almacenada en un repositorio permite a los administradores del sitio poder gestionar la información recopilada, obteniendo datos como cuales son los productos mas y menos utilizados, horarios máxima conexión, transacciones realizadas con mayor frecuencia, errores ocurridos durante las transacciones, entre otras.

3.4.3. Objetivo del sistema de *log* transaccional

El objetivo principal de un sistema de ***log*** es almacenar en un repositorio de datos todos los movimientos que realicen los clientes en un sitio web transaccional.

La información almacenada en el repositorio de datos debe estar disponible para el cliente en cualquier momento que éste lo solicite, además, el mantener un registro con todos los movimientos permite a los administradores del sitio proveer de información histórica de las operaciones realizadas a sus clientes, gestionar estos datos obteniendo información relevante para sus productos, información tal como productos más visitados, operaciones realizadas con mayor frecuencia, horarios y fechas de mayor conexión, además de poder almacenar todas las entradas y salidas a los distintos servicios de datos que se utilizan en el sitio, teniendo de este modo detalles de posibles errores ocurridos en estos servicios pertenecientes a otras áreas.

Es necesario almacenar las entradas y salidas de los servicios de datos en otro repositorio, en el cual también se almacenen los posibles errores ocurridos en su ejecución, para así evitar mezclar la información de las operaciones realizadas satisfactoriamente por los clientes con las entradas, salidas y posibles errores de los servicios de datos.

Deben ser las mismas páginas dinámicas las que se encarguen de instanciar al sistema de ***log*** para que registre una operación cuando ésta se realiza, independiente del éxito, debe almacenar todos los datos de la operación para su posterior despliegue como una opción de las transacciones realizadas, y además deben ser las componentes las encargadas de instanciar el sistema ***log*** para que almacene en el repositorio correspondiente la entrada y salida al servicio de datos y el error, si es que ocurre.

3.4.4. Entidades relacionadas con el sistema de *log*

El sistema de ***log*** es el encargado de almacenar cada una de las transacciones u operaciones que realice un cliente en el sitio web transaccional, como también el encargado de almacenar las entradas, salidas y posibles errores de los servicios de datos y debe ser el único que realice estas tareas dentro del sitio web. Para ello debe recibir toda la información de una transacción realizada por parte de la página dinámica y registrarla en el repositorio destinado para ello, la información almacenada debe estar indexada y debe estar asociada a cada cliente, para poder acceder a ella posteriormente cuando el cliente desee ver todas las operaciones realizadas por él.

La otra labor del sistema de ***log*** es almacenar cada entrada, salida y posible error que ocurra al usar un servicio de datos, para ello debe recibir toda esta información por parte de la componente que interactúa con el servicio de datos y almacenar esa información en un repositorio distinto al de las transacciones realizadas, este repositorio será revisado por los administradores del sitio para su posterior corrección o deliberación del problema al área responsable de éste.

3.4.5. Entradas y salidas del sistema de *log*

La relación entre las dos entidades que interactúan con el sistema de ***log*** transaccional se puede definir de la siguiente forma:

3.4.5.1. Sistema de *log* – Páginas dinámicas

Las páginas dinámicas deben tener estipulado dentro de su codificación que cada vez que se realice una transacción u operación financiera dentro del sitio web, debe ser el sistema de ***log*** transaccional el encargado de registrar esa información en el repositorio destinado para ello, de esta forma, es la página dinámica la encargada de entregar todos los datos necesarios para registrar el movimiento, como lo son el rut del cliente, la acción realizada, las cuentas involucradas, los montos transferidos, los productos utilizados, la fecha y hora en que se realiza el movimiento y otros datos que los administradores del sitio consideren necesario almacenar.

También debe existir una página dinámica que obtenga, a través de un componente, información del sistema de ***log***, esto debido a que el cliente del sitio puede solicitar información de las operaciones ya realizadas por él, lo que sería una especie de cartola de movimientos o historial de operaciones realizadas.

3.4.5.2. Sistema de *log* – Componentes

Las componentes que utilizan servicios de datos deben tener instanciada dentro de su codificación que el sistema de ***log*** almacenará las entradas, las salidas y los posibles errores que puedan ocurrir al llamar a un servicio de datos. Con esto se pueden corregir todo tipo de problemas que ocurren en la llamada a estos servicios que no dependen de los administradores del sitio web,

ya que quedaría un registro de cada movimiento realizado con los distintos servicios que operan en un sitio web.

La componente debe utilizar al sistema de ***log*** para que registre todo lo que ocurra con los servicios de datos antes de llamar al manejador de errores, si es que es necesario utilizarlo, ya que la función del sistema de ***log*** es solo almacenar la entrada a un servicio de datos, el nombre del servicio, la fecha y hora de la llamada al servicio y el retorno o salida que este entrega, y de ocurrir un error es el manejador de errores el encargado de recepcionarlo.

Cuando una componente reciba la petición de una página dinámica para obtener los movimientos o transacciones de un cliente, debe realizar una solicitud al sistema de ***log*** para que éste entregue la información que se encuentra en su repositorio de datos y que corresponde a la del cliente solicitado.

3.4.6. Modelo Final

El modelo final del sistema de LOG transaccional se representa en la siguiente figura:

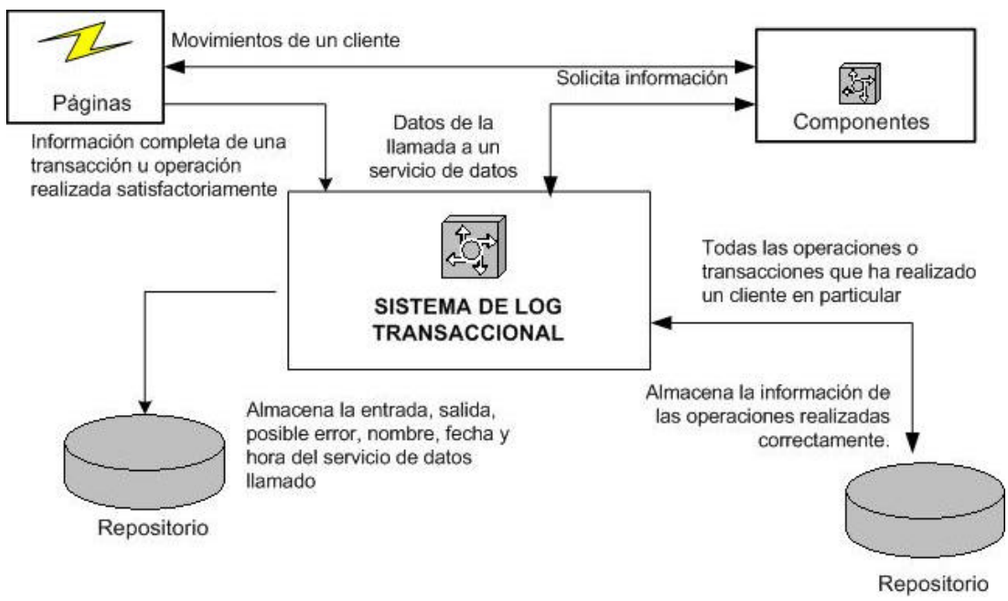


*Figura Cap3_7, Sistema de **log** transaccional*

En la figura se aprecia la interacción del sistema de ***log*** transaccional y las dos entidades que se relacionan con él, las páginas dinámicas y componentes. Las primeras son las encargadas de proporcionar al sistema de ***log*** la información completa de una transacción u operación que se realizó dentro del sitio, para que sea este el que la registre dentro del repositorio de datos destinado para ello. Las componentes son las encargadas de entregar al sistema de ***log*** toda la información involucrada en la llamada a un servicio de datos para que este lo almacene en un repositorio en particular, la información almacenada corresponde a la entrada, salida, posible error y nombre del servicio de datos llamado, además de la fecha y hora de su llamada.

Cuando una página dinámica solicita información histórica de los movimientos de un cliente debe hacerlo a través del motor de negocio, para que sea éste el que haga el requerimiento al sistema de ***log*** de la información necesitada por la página, el sistema de ***log*** debe acudir a su repositorio de datos y entregar la información solicitada al motor, el cual se encarga de procesarla y devolverla a la página solicitante.

3.5 Otras falencias del sitio web de *BECH*

Corresponde a las otras falencias detectadas en el estudio del actual sitio web de ***BECH***

3.5.1 Falencias que no serán resueltas

Dentro de las falencias detectadas en el actual sitio web de BECH, varias de ellas dependen de áreas ajenas a la de informática y otras forman parte de las exclusiones de este proyecto de tesis, por tal motivo no serán solucionadas íntegramente.

3.5.1.1 Problemas con los servicios de datos

Los problemas presentados con los servicios de datos ya sea estos ***SQL*** o ***CICS*** se refieren a menudo a su casi nula capacidad de manejar errores o a el exceso de servicios existentes en ambos ambientes. Como prueba de ello es que existe un servicio ***CICS*** para el saldo de cuenta corriente, otro para el saldo de cuenta de ahorro, otro para el de línea de crédito, otro para el de cuenta vista y así sucesivamente para cada uno de los productos que ofrece el banco, en vez de existir un solo servicio de saldos, que sea parametrizable y entregue el saldo del producto solicitado en la invocación del servicio. La misma situación ocurre en el ambiente de ***SQL***, puesto que existen muchos servicios realizando las distintas operaciones sobre las bases de datos, existiendo muchos que realizan la misma función, pero como se crearon para aplicaciones distintas, son diferentes servicios, es decir, no existe una reutilización de los servicios existentes.

3.5.1.2 Ambientes de desarrollo, test y producción

Es necesario que en un sitio web existan y estén claramente definidos los distintos ambientes; el de desarrollo, test, pre-producción y producción. Esto se debe cumplir con mayor razón cuando se trata de un sito web transaccional, debido a que las aplicaciones desarrolladas intervienen directamente en las transacciones realizadas por los clientes del sitio. Al existir claramente diferenciados los distintos ambientes se pueden desarrollar aplicaciones en desarrollo, luego ser probadas en el ambiente de test, si la prueba es satisfactoria llevar la aplicación al ambiente de pre-producción (que es similar a producción) y si el comportamiento es el esperado pasar la aplicación al ambiente de producción.

También es necesario que en los distintos ambientes existan todos los mecanismos necesarios para realizar un completo plan de pruebas y no pasar las aplicaciones a otro ambiente, sino hasta que se hayan completado y certificados todas las pruebas correspondientes. La no existencia o la existencia a medias de un ambiente puede provocar errores en producción por instalar ahí aplicaciones que no han sido certificadas correctamente o que no pudieron ser certificadas en los ambientes previos debido a falta de mecanismos de pruebas o pruebas de estrés, entre otras.

3.5.1.3 Pasos a producción

El área encargada de pasar las aplicaciones a producción a menudo comete errores, tales como no replicar correctamente la aplicación en todos los servidores de producción, instalar versiones distintas a las solicitadas, una mala compilación de los componentes y una instalación errónea de éstos en los servidores. Para controlar estos problemas además de identificar claramente cada paso a producción, es necesario crear un sistema de pasos a producción, en el cual se identifique claramente la persona que solicita el paso, el motivo de este, las páginas y componentes involucradas, un comentario con indicaciones para realizar el paso y una identificación clara de lo que el paso involucra. Además es necesario que se pueda seguir el curso de una solicitud, para saber en que esta se encuentra y si ha habido complicaciones con su instalación.

3.5.2. Solución a otros problemas del sitio web

De las otras falencias detectadas en sitio actual de ***BECH*****,** se presenta una solución teórica al siguiente problema:

3.5.2.1. Manejador de fuentes

Es habitual que en un grupo de desarrollo en un momento dado puedan existir dos personas modificando la misma aplicación y al momento de pasar la aplicación a producción el cambio realizado por un desarrollador elimine los realizados por el otro. Además del problema presentado al desarrollar, existe la problemática de que es difícil identificar las modificaciones hechas a una aplicación, ya que todas las modificaciones se van realizando sobre la misma fuente. Para evitar estos problemas es necesario contar con un control o manejador de fuentes.

Un manejador de fuentes debe permitir que solo un desarrollador pueda tomar una fuente para ser modificada, así evitará que un segundo desarrollador modifique la misma fuente y elimine el trabajo hecho por el primer desarrollador, además el manejador debe exigir que al modificar una fuente el desarrollador identifique el cambio realizado bajo un nombre creando así una versión de la fuente, de este modo se irán generando versiones del código fuente tantas veces como modificaciones sufra la aplicación, así será más fácil identificar la fuente asociada a un cambio específico. El manejador de fuentes debe contener todas las páginas del sitio como también la de todos los componentes utilizados. El manejador debe estar asociado a los distintos ambientes que se utilicen en el sitio web, para así identificar la fuente que se utiliza en el ambiente de desarrollo, test, pre-producción y producción, así como también debe tener asociado permisos o privilegios para que solo los usuarios autorizados puedan tomar una fuente, modificarla y cambiarla de ambiente.

CAPITULO CUATRO

IMPLEMENTACION TEORICA

4. Implementacion teórica

Actualmente ***BECH*** cuenta con tecnologías y hardware que no pueden ser rechazados a priori como alternativa posible de implementación para el nuevo sitio debido al costo y esfuerzo ya invertidos. La tecnología básicamente corresponde al estándar ***DNA***, es por eso que en este capítulo se implementan teóricamente las soluciones dadas en el capítulo tres, la implementación debe utilizar el estándar utilizado por ***BECH***.

4.1. Definición de conceptos

En el capítulo tres se utilizaron conceptos genéricos independientes a la tecnología a implementar. Debido a que en este capítulo se habla del estándar ***DNA***, es necesario describir particularmente cada tecnología a utilizar y su correspondencia con los conceptos ya mencionados, estos conceptos son:

- **XML:** Corresponde a archivos de texto que contienen datos estructurados, esto es, datos que tienen relaciones internas bien definidas y restricciones que proporcionan un contexto para los datos. En ***BECH*** se utiliza la versión 4.0.
- **DLL:** Es la expresión práctica del concepto ya utilizado en capítulos anteriores para referirse a componentes de negocio. Corresponde a código compilado que contiene rutinas reutilizables. En ***BECH*** estos archivos están programados con ***VB* 6.0**.
- **XSL:** Es la expresión práctica del concepto utilizado para referirse a plantillas, es una tecnología ***XML*** de hojas de estilos que sirve para darles formato de presentación a documentos ***XML***, es decir, es un documento que describe un conjunto de reglas

encaminadas a la presentación de este tipo de documentos en formato ***HTML***.

- **ASP:** Es la expresión práctica del concepto ya utilizado en capítulos anteriores para referirse a páginas dinámicas. Corresponde a código compilado en tiempo de ejecución que generalmente instancian a ***DLL***s para utilizar sus rutinas.

4.1.1 Especificación de seudocódigo

El seudocódigo a utilizar para las implementaciones teóricas de las soluciones dadas a los principales problemas de ***BECH***, corresponde a código ***VB* 6.0** y a código ***ASP***. Las especificaciones a utilizar en todos los ejemplos teóricos corresponden a:

- Instanciar una ***DLL*** dentro del código ***ASP***

```
Dim MiObjeto
Set MiObjeto=Server.CreateObject("Dll.Clase")
```

- Instanciar una ***DLL*** dentro del código ***VB***

```
Dim MiObjeto as Dll.Clase
Set MiObjeto= New Dll.Clase
```

Ambas instancias a ***DLL***s hacen que el servidor web cree una referencia a la clase "Dll.Clase" para su uso tanto en el ***ASP*** como en el código ***VB***.

- Establecer conexión con una base de datos ***SQL***

```
'Se utiliza algún componente para leer archivos de configuración
'con el que se rescata el string de conexión a la base de datos
'en el caso del BECH se usa una componente llamada Confini.dll

Dim objIni As New CONFINILib.RWIni
Dim strConnectionString
Set objIni = New CONFINILib.RWIni
objIni.ConfFile = "conexion.ini"
strConnectionString = objIni.GetValue("DATABASE", "strCnx")

'se establece la conexión con la base de datos
Conn.ConnectionString = strConnectionString
Conn.Open
```

- Establecer conexión con bases de datos ***IBM HOST.*** Este tipo de conexiones se utiliza para llegar a los datos de negocio propiamente tales (archivos de clientes, de cuentas corrientes, etc.), para ello se utiliza un componente propietario, llamado ***CucBech***, el que funciona como una caja negra para el canal Internet. Al establecer una conexión de este tipo, se le debe proporcionar al componente el nombre del servicio de datos a invocar, y la entrada respectiva en forma de una cadena de texto, este componente propietario solicita la ejecución de dicho servicio de datos y recibe la respuesta, la cual es devuelta en una cadena de texto a quien lo haya instanciado. Desde ***VB*** el seudocódigo para llegar a estas bases es el siguiente:

  Dim Conec_Datacom as CUCBECHLib.CICS
  *Set Conec_D*atacom = *new ("CUCbech.CICS")*
  Respuesta = C*onec_D*atacom.*llamaCics(nombre_Servicio, CadenaEntrada)*

  Donde Respuesta recibe en cadena de texto la salida que retorna el servicio, la cual debe ser interpretada.

4.2 Motor para la capa de presentación

El "motor para la capa de presentación", tiene sub-partes en la capa de negocio y en la de datos, todas ellas unidas con el objetivo de dar formato y diseño a las respuestas obtenidas a partir de la solicitud del cliente Internet.

La presencia en las dos capas se puede describir de la siguiente manera:

- **Capa de Negocio:** ***DLL*** cuyo fin es acceder a los datos propios del motor para la capa de presentación, en el cuál se define que ***XSL*** utilizar.
- **Capa de Datos:** Repositorios configurables que almacenen la relación entre información solicitada y ***XSL*** a utilizar para determinada consulta.

El formato y diseño a aplicar dependen de la consulta solicitada por el cliente, diferenciando si corresponde a información del sitio privado o del sitio público. La distinción que hace el "motor para la capa de presentación" entre estos dos sitios, radica en la existencia de una relación entre el ***XSL*** a aplicar y el tipo de solicitud.

Para el sitio público existen distintas formas de esquemas para el ***home*** que son configurables para permitir cambios no costosos en la presentación final a los clientes.

En cambio, para el sitio privado se utiliza básicamente una serie finita de esquemas que mantienen cierta continuidad para desplegar las respuestas solicitadas según formato y diseño correspondiente, ya sea esto un saldo, cartola u otro, con el fin de no saturar al cliente con saldos de un tamaño y cartolas de otro.

Resumiendo, si el "motor para la capa de presentación" recibe una consulta por medio del servidor web, se gatilla por medio del "motor para la capa de negocios" un servicio de datos que manipula información referente a dicho requerimiento, aplicando posteriormente el formato y diseño según sitio público o privado

4.2.1 Implementación del motor para la capa de presentación.

En la figura Cap3_3, se ilustra el modelo final del motor para la capa de presentación, la implementación se hace en base a ese modelo y es por medio de código ***VB***, la ***DLL*** tiene una sola clase. El nombre de la ***DLL*** es *"Motor_Presentacion"* y la clase, en la cual se agregarán todos los métodos creados, se llama *"Cls_Presentacion"*.

4.2.1.1 Relación entre archivo XSL e información solicitada

El nombre de este método es *"Obtiene_Archivo_XSL"* y tiene como parámetros de entrada los siguientes campos: tipo de producto y tipo de servicio. Para leer el repositorio "Relación XSL – Información" existe un servicio de datos de nombre *"Svc_nombre_XSL"*, que tiene los mismos parámetros de entrada que este método y que retorna el nombre y dirección física del archivo ***XSL*** a utilizar. El siguiente seudocódigo ilustra este método:

```
Private Function Obtiene_Archivo_XSL (Byval Tipo_Producto as String,_
                                      Byval Tipo_Servicio  as String
                                      ) as string

    'Se establece la conexión con la base de datos SQL

    'se declaran variables del tipo ADO para trabajar con la base de datos
    Dim Conn As ADODB.Connection
    Dim Obj As ADODB.Command
    Dim rs As ADODB.Recordset

    Set Conn = New ADODB.Connection
    Set Obj = New ADODB.Command
    Set rs = New ADODB.Recordset

    'se definen los 2  parámetros necesarios
    Dim parametro1  As ADODB.Parameter
    Dim parametro2  As ADODB.Parameter

    'se indica que se ejecutará un procedimiento almacenado
    Obj.ActiveConnection = Conn
    Obj.CommandType = adCmdStoredProc
    Obj.CommandText = "Svc_Nombre_XSL"

    'se le asignan los valores a los 2 parámetros
     Set parametro1 = Obj.CreateParameter("tip_prod", adVarchar, adParamInput,
    10, Tipo_Producto)
    Set parametro2 = Obj.CreateParameter("tip_srvc", adVarchar, adParamInput,
    10, Tipo_Servicio)

    'se agregan los 2 parámetros a la llamada del servicio
    Obj.Parameters.Append parametro1
    Obj.Parameters.Append parametro2

    'se ejecuta el procedimiento almacenado
    Set rs = Obj.Execute

    'se asigna la salida separada por un asterisco, puesto que son dos
    'los parámetros retornados por el método

    Obtiene_nombre_XSL= rs(0) & "*"  & rs(1)

End function
```

4.2.1.2 Aplicar una plantilla XSL a un XML

El nombre de este método es *"Aplica_XML_XSL"* y tiene como parámetros de entrada un XML con información y la dirección física del archivo ***XSL*** a utilizar, aplica el ***XSL*** sobre el ***XML*** y retorna como salida código ***HTML***. El siguiente seudocódigo ilustra este método:

```
Private Function Aplica_XML_XSL (Byval cadena_XML as string,_
                                 Byval direccion_XSL as string
                                 ) as string

'Se definen variables de tipo XML
Dim ObjRaizXml As MSXML2.FreeThreadedDOMDocument40

'se define variable que permite leer archivo del disco duro
Dim filexsl as Scripting.FileSystemObject
Dim cadena as string

'se carga la cadena en formato XML para trabajarla como archivo XML
ObjRaizXml.async = True
ObjRaizXml.loadXML (cadena_Xml)

'se obtiene el archivo desde el disco duro del servidor
cadena=""
Set ArchivoXsl = filexsl.OpenTextFile(direccion_XSL, 1, 0)
Do While Not (ArchivoXsl .AtEndOfStream)
        cadena = cadena  & fs.readline
Loop

 'se selecciona el nombre de la etiqueta para comenzar a leer el archivo
Set ObjHeader = ObjRaizXml.selectNodes("ETIQUETA")

'se define una variable temporal donde almacenar el HTML generado
salida=""

'se escribe el encabezado HTML
salida = salida & "<html><title>BancoEstado</title>"
salida = salida & "<body>"

'se comienza a leer el archivo y a obtener los datos contenidos en el
For Each Objhijo In ObjHeader
        Set child = Objhijo.getElementsByTagName("TIPODATO")
        Tipodato=child(0).Text
        Set child = Objhijo.getElementsByTagName("VALORDATO")
        Valordato=child(0).Text
        'se aplica el template correspondiente de la plantilla XSL
        Resultado=AplicaPlantila(cadena,TipoDato,ValorDato)
        'se agrega a la salida el resultado de la aplicación de la plantilla
        'sobre el XML
        salida = salida & Resultado
Next
salida = salida & "</body></html>"

Aplica_XML_XSL= salida

End function
```

4.2.1.3 Generador de Presentación

Una vez creado los métodos necesarios para el funcionamiento del motor de presentación, falta uno que se encargue de sincronizar y realizar todas las actividades para poder presentar en el navegador del cliente la información que éste solicita y en el formato que ***BECH*** tenga estipulado. El nombre de este método es *"Presenta_Informacion"* y recibe como parámetros de entrada el tipo de solicitud, el tipo de producto y los XML con información. La salida es una cadena de texto que contendrá un ***HTML*** que el servidor web debe enviar al navegador del cliente, el siguiente seudocódigo ilustra este método:

```
Public Function Presenta_Informacion (Byval tipo_sol as string,_
                                      Byval tipo_prod as string, _
                                      Byval cadena_entrada as string
                                      ) as string

Dim InfoRecibidaCapaNegocio as String

'se obtiene la dirección física de la plantilla a utilizar
dirfisica=Obtiene_Archivo_XSL (tipo_prod,tipo_sol)

'se aplica la plantilla XSL al la información en formato XML
salida= Aplica_XML_XSL (cadena_entrada, dirfisica)

'se envía la respuesta
Presenta_Informacion=salida

End function
```

4.3 Motor para la capa de negocios

El "motor para la capa de negocios" tiene presencia en dos de las tres capas ***DNA***, su objetivo es acceder a los servicios de datos y aplicar cierta lógica de negocio a las respuestas obtenidas.

La presencia en estas dos capas se describe a continuación:

- **Capa de Negocios:** ***DLL***s que básicamente cumplen dos funciones:
  - Obtener respuestas de los servicios de datos que se soliciten.
  - Aplicar la lógica de negocios necesaria a estas respuestas para retornar al motor de presentación una respuesta ***XML***.

- **Capa de Datos:** Repositorios configurables que almacenen la siguiente información:
  - Relación entre información solicitada y servicio de datos.
  - Formato del ***XML*** a generar como respuesta.

En el punto 3.2 se presentó un modelo teórico de cómo debiera ser el motor para la capa de negocios, en este punto se tomará ese modelo y se indicará la forma de implementarlo bajo la plataforma ***DNA***.

4.3.1 Modelo motor para la capa de negocios

En la figura Cap3_5, se ilustro el modelo final del motor para la capa de negocios. La funcionalidad de la ***DLL*** de lógica del negocio y generador de ***XML*** consiste en determinar que tipo de servicio de datos se utiliza (independiente de si se trata del sitio público o del privado), obtener la información solicitada por medio del servicio de datos ya obtenido en el paso anterior, utilizando a la ***DLL*** para obtención de datos para crear el ***XML*** de salida.

4.3.2 Implementación del motor para la capa de negocios

Esta implementación cuenta con dos ***DLL***s, una para la obtención de datos y otra para la lógica del negocio propiamente tal.

4.3.2.1 DLL única para la obtención de datos.

El nombre de la DLL es *"Motor_Negocios_Datos"* y la clase en la cual se agregan los métodos se llama *"Cls_datos"*.

4.3.2.1.1 Obtener tipo de base y servicio de datos a utilizar.

El nombre de este método es *"Obtener_Base_Servicio_Datos"* y tiene como parámetros de entrada el código consulta a utilizar, retornando el tipo de base de datos y el nombre del servicio de datos a llamar. Tiene asociado un servicio de datos que lee del repositorio "Relación entre información solicitada y servicio de datos", que tiene la misma entrada y se llama *"svc_Base_Servicio_Datos*". El siguiente seudocódigo ilustra este método:

```
Private Function Obtener_Base_Servicio_Datos (codigo_consulta as string _
                                                ) as as ADODB.Recordset

'Se establece conexión con la base de datos SQL
'se definen variables de tipo ADO
Dim Conn As ADODB.Connection
Dim Obj As ADODB.Command
Dim rs As ADODB.Recordset

'se indica que se ejecutará un procedimiento almacenado
Obj.ActiveConnection = Conn
Obj.CommandType = adCmdStoredProc
Obj.CommandText = "svc_Base_Servicio_Datos"

Set parametro1 = Obj.CreateParameter("Cod_Cons", adInteger, adParamInput, 10,
Cint(codigo_consulta))

Obj.Parameters.Append parametro1

'se ejecuta el procedimiento almacenado
Set Obtener_Base_Servicio_Datos = Obj.Execute
End function
```

4.3.2.1.2 Método para ejecución de servicio de datos

El nombre de este método es *Ejecuta_Serv_Datos"* y tiene como parámetro de entrada el tipo de base de datos, el nombre del servicio de datos y su entrada, retornando una cadena de texto sin formato con la respuesta. El siguiente seudocódigo ilustra este método:

```
Public Function Ejecuta_Serv_Datos (tipobase  as string, _
                                    Nomb_serv_datos as string, _
                                   Entrada as string) as string

'se establece la conexión con el tipo de base
if tipobase=SQL then
        'se estable comunicación con base SQL
        'se prepara ejecución del  SP
        'se ejecuta el Procedimiento Almacenado con los
        'parámetros de entrada correspondiente
```

```
else
        'se estable comunicación con base HOST IBM
        'se prepara ejecución del servicio CICS
        'se ejecuta el servicio de datos con la cadena de entrada
end if
'se asigna el resultado de la ejecución del servicio de datos
Ejecuta_Serv_Datos = salida

End function
```

4.3.2.2 DLL única para lógica de negocios

Esta ***DLL*** es la encargada de consultar a través de la "***DLL*** única para la obtención de datos" por los servicios de datos necesarios y les aplica lógica de negocio a la información obtenida, retornando una estructura ***XML*** al servidor web su despliegue final. El nombre de la ***DLL*** es *"Motor_Negocios_Logica"* y la clase en la cual se agregan los métodos se llama *"Cls_Logica"*, que cuenta con los siguientes métodos:

4.3.2.2.1 Método para llamar a la *DLL* única para la obtención de datos

El nombre de este método es *"LLamar_DLL_Datos"* y tiene como parámetros de entrada el código de consulta a utilizar y la cadena de entrada para dicha consulta, recibe una cadena de texto sin formato como respuesta. El siguiente seudocódigo ilustra este método:

```
Private Function LLamar_DLL_Datos (codigo as string, _
                                   entrada as string _
                                   ) as string

'Se establece la conexión a la base de datos
'se instancia a la componente única para la obtención de datos

dim comp as motor_negocios_datos.cls_datos
set comp = New motor_negocios_datos.cls_datos
set LLamar_DLL_Datos = comp.Ejecucion_Servicio_Datos(codigo,entrada )

End function
```

4.3.2.2.2 Método para obtener formato XML.

El nombre de este método es *"Obtener_formato_XML"* y tiene como parámetros de entrada el código consulta a utilizar, retornando el esquema del

XML a generar, sin información tan solo como plantilla a llenar con datos. Tendrá asociado un servicio de datos que lee del repositorio "Formato de ***XML***", este servicio de datos tiene la misma entrada que el método y se llama *"svc_Obtener_formato_XML*". El siguiente seudocódigo ilustra este método:

```
Private Function Obtener_formato_XML (codigo_consulta as string _
                                                    ) as string

'Se establece conexión con la base de datos SQL
'se definen variables de tipo ADO
Dim Conn As ADODB.Connection
Dim Obj As ADODB.Command
Dim rs As ADODB.Recordset

'se indica que se ejecutará un procedimiento almacenado
Obj.ActiveConnection = Conn
Obj.CommandType = adCmdStoredProc
Obj.CommandText = "svc_Obtener_formato_XML"

Set parametro1 = Obj.CreateParameter("Cod_Cons", adInteger, adParamInput, 10,
Cint(codigo_consulta))

Obj.Parameters.Append parametro1
'se ejecuta el procedimiento almacenado
Set  rs = Obj.Execute
Obtener_formato_XML= rs(0)
End function
```

4.3.2.2.3 Método para generar XML de respuesta

El nombre de este método es *"Generar_XML_respuesta"* y tiene como parámetros de entrada la cadena de texto sin formato obtenida desde la ***DLL*** única para la obtención de datos y el formato del XML a generar. El siguiente seudocódigo ilustra este método:

```
Public Function Generar_XML_respuesta (cadena_respuesta  as string, _
                                                    XML_a_utilizar as XML
                                                    ) as XML
'se definen variables a utilizar
Dim strXml As String
Dim contador as integer
Dim i as integer
I=0

'Se definen variables de tipo XML
Dim ObjRaizXml As MSXML2.FreeThreadedDOMDocument40
'se carga el formato XMLa utilizar para trabajarlo como archivo XML
ObjRaizXml.async = True
ObjRaizXml.loadXML (XML_a_utilizar)
 'se selecciona el nombre de la etiqueta para comenzar a leer el archivo
Set ObjHeader = ObjRaizXml.selectNodes("ETIQUETA")
 'se escribe el encabezado XML
strXml = "<?xml version=""1.0"" ?><SALIDA>"
```

```
For Each Objhijo In ObjHeader
        Set child = Objhijo.getElementsByTagName("NOMBRECAMPO")
        nombcampo=child(0).Text
        Set child = Objhijo.getElementsByTagName("LARGOCAMPO")
        largocampo=child(0).Text
        Set child = Objhijo.getElementsByTagName("FORMATOCAMPO)
        formatocampo=child(0).Text
        Set child = Objhijo.getElementsByTagName("ITERACIONESDELCAMPO)
        iteraciones=child(0).Text
        for i=0 to iteraciones
                valorcampo=mid(cadena_respuesta,i, largocampo)
                i= i + largocampo + 1
                'el formato puede indicar la cantidad de decimales de un
                'valor, el formato de una cuenta corriente, etc.
                Campoformateado=aplicaformato(valorcampo,formatocampo)
                'se agrega a la cadena de salida
                strXml = strXml & "<" & nombcampo & ">" & Campoformateado
                strXml = strXml & "</" & nombcampo & ">"

        next
Next
strXml = strXml & "</SALIDA></xml>"
Generar_XML_respuesta = strXml
End Function
```

4.4 Manejador de errores o excepciones

Como se definió en el punto 3.3 el manejador de errores debe interpretar cada error ocurrido tanto en una página dinámica como en un componente, y enviar al cliente que solicita información, un mensaje entendible que a la vez permita generar un reporte para los administradores del sitio, enviando si corresponde, una notificación a los encargados de las aplicaciones.

Como el manejador de errores debe poder ser instanciado desde una componente o de una página web dinámica, se requiere que éste sea en si una componente, es decir, una ***DLL***.

4.4.1 Interacción con las entidades relacionadas

En el caso de las páginas ***ASP***s, éstas deben tener instanciado el manejador de errores dentro de su codificación , para ello lo instancian como si fuera un objeto, accediendo de este modo a las clases y métodos que posee el

manejador, y en el caso de las ***DLL***s, éstas deben tener referenciado e instanciado el manejador de errores que es en sí otra ***DLL***.

Una vez instanciada la **DLL** en cualquier objeto definido para ello se puede comenzar a ocupar cada uno de los métodos que posea la clase instanciada.

4.4.2 Definición del funcionamiento de la componente manejadora de errores

Esta componente debe recibir como parámetros de entrada todos los datos necesarios para almacenar el error dentro de un repositorio de datos, basándose en el tipo de error ocurrido y el lugar en donde ocurrió, leer otro repositorio de datos y rescatar el mensaje a entregar al cliente junto con las acciones a seguir, según estas acciones interactuar con otro componente encargado de enviar mensajería, generarando a la vez un reporte de errores que posteriormente son enviados a los administradores del sitio web del banco. Los repositorios a utilizar por esta componente son bases de datos ***SQL* 7.0**.

Cada una de las acciones a realizar por esta componente debe ser definida como un método dentro de una clase.

4.4.2.1 Almacenamiento del error

Para almacenar el error dentro de una tabla ***SQL*** destinada para ello debe hacerse vía un procedimiento almacenado para no acceder directamente a la base de datos desde la componente. El ***SP*** construido debe recibir como parámetros los siguientes campos: número del error, descripción del error, origen del error (página o componente), nombre del origen (nombre de la página asp o del método, clase y componente), fecha y hora.

Una vez que el ***SP*** reciba los parámetros proporcionados por la componente manejadora de errores, los debe registrar en la tabla de la base ***SQL*** destinada para ello y retornar el Número o ID con el cual fue grabado en la base dicho error.

4.4.2.2 Entrega de mensajes a clientes y acciones a seguir

Para poder acceder a los mensajes que se deben entregar a los clientes y las acciones a seguir se debe contar con otro ***SP*** que acceda a la tabla ***SQL*** que contiene esta información. Para ello el método que realiza esta acción debe entregar como parámetro el número del error y la fuente de éste (página o componente), con éstos datos el ***SP*** debe entregar el mensaje a desplegar al cliente, las acciones a seguir y si corresponde o no enviar un mensaje a los administradores o encargados de la aplicación que esta originando los problemas además de los nombres y direcciones de correo de éstos.

El mensaje a desplegar como las acciones a seguir deben ser retornadas por el componente manejador de errores a quien lo solicite y si corresponde enviar un mensaje debe entregar esta información al método encargado de enviar el mensaje.

4.4.2.3 Envío de mensajes

La misma componente manejadora de errores cuando invoque al método que rescata si corresponde o no enviar un mensaje, debe pasar a este nuevo método los parámetros necesarios para enviar el mensaje a los administradores o responsables de la aplicación que esta originando problemas, todo esto a través de la componente que realiza la mensajería dentro de las aplicaciones del banco. Este método debe generar un correo electrónico indicando el origen del problema, dónde se presentó y la fecha y hora de ocurrencia, una vez

generado este mensaje lo debe enviar a las casillas que corresponda utilizando al componente encargada de realizar efectivamente el envío del correo electrónico.

4.4.2.4 Generación de reportes de error

Cada vez que ocurra un error de funcionamiento de alguna aplicación o algún otro error que haya sido definido como reportable por los administradores del sitio, se debe almacenar toda esta información en una tabla ***SQL*** temporal, la cual será vaciada posteriormente a una plantilla excel que será enviada al área de desarrollo internet como un reporte de errores generado por la componente manejadora de errores que existe en el sitio web. La generación de esta plantilla excel será realizada vía una aplicación, que utilizarán los administradores o encargados del sitio y que accederá a través de la componente manejadora de errores directamente a las bases ***SQL*** que almacenan esta información.

4.4.3 Manejo de errores en Visual Basic y VBScript

Las opciones disponibles para el manejo de errores difiere entre Visual Basic y ***VBScript***, este último utilizado por las páginas ***ASP***. En Visual Basic hay varias instrucciones de manejo de errores disponibles, como lo son: *On Error Go To <etiqueta>, On Error Resume* y *On Error Resume Next.* En VBScript, la única construcción de manejo de errores es *On Error Resume Next*

4.4.3.1 Instrucción “On Error”

Activa una rutina de control de errores y especifica la ubicación de la misma en un procedimiento; también puede utilizarse para desactivar una rutina

de control de errores. La sintaxis es : **On Error GoTo** *línea,* **On Error Resume Next** y **On Error GoTo 0**

La sintaxis de la instrucción **On Error** puede tener cualquiera de los formatos siguientes:

Instrucción	**Descripción**
On Error GoTo *etiqueta*	Activa la rutina de control de errores que comienza en la línea especificada en el argumento necesario *"etiqueta".* El argumento *"etiqueta"* es cualquier etiqueta de línea o número de línea. Si se produce un error en tiempo de ejecución, el control pasa a *"etiqueta"*, activando el controlador de errores. La línea especificada en el argumento *"etiqueta"* debe encontrarse en el mismo procedimiento que la instrucción *On Error*; o de lo contrario, se producirá un error en tiempo de compilación.
On Error Resume Next	Especifica que, en caso de que se produzca un error en tiempo de ejecución, el control pase a la instrucción que sigue inmediatamente a aquélla en la que se ha producido el error, donde continúa la ejecución.
On Error GoTo 0	Desactiva cualquier controlador de errores del procedimiento actual.

Tabla Cap4_A, Formato de la instrucción On Error

Si no se utiliza una instrucción *On Error*, cualquier error en tiempo de ejecución que se produzca será fatal; es decir, aparece un mensaje de error y la ejecución se detiene.

Un controlador de errores "activado" es aquél que se ha habilitado mediante una instrucción *On Error* y un controlador de errores “activo" es un controlador activado que se encuentra en el proceso de tratar un error.

Si se produce un error mientras un controlador está activo (entre la aparición del error y una instrucción *Resume, Exit Sub, Exit Function* o *Exit Property*), el controlador de errores del procedimiento actual no puede tratarlo. El control vuelve al procedimiento que hace la llamada, y si éste cuenta con un controlador de errores activado, se utiliza ese controlador para tratar el error. Si también está activo el controlador de error del procedimiento que hace la llamada, el control vuelve hacia los procedimientos llamadores anteriores hasta llegar a un controlador de error activado, pero no activo. Si no se encuentra un controlador de errores activado e inactivo, el error es fatal en el punto en el que se produjo.

Cada vez que el controlador de errores devuelve el control al procedimiento que hace la llamada, éste se convierte en el procedimiento actual. Una vez tratado un error con un controlador en cualquier procedimiento, la ejecución continúa en el procedimiento actual en el punto designado por la instrucción *Resume*.

Una rutina de control de errores no es un procedimiento *Sub* ni *Function.* Es una sección de código marcada con una etiqueta o un número de línea.

4.4.4 Implementación del Manejador de Errores

El manejador de errores tiene una sola clase en la cual se agregan los métodos a medida de que éstos sean necesarios. El nombre de la componente es *“Controlador”* y la clase en la cual se agregan los métodos se llama *“Cls_Errores”*.

Para controlar los errores que ocurran dentro de la componente manejadora de errores, se utiliza una rutina simple que solo devuelve a quien lo solicita un mensaje indicando que no se pudo ejecutar correctamente el controlador de errores.

4.4.4.1 Método que almacena en el error

El nombre de este método es *"AlmacenaError"* y tiene como parámetros de entrada los siguientes campos: número del error, descripción del error, origen del error (página o componente), nombre del origen (nombre de la página ***ASP*** o del método, clase y componente), fecha y hora. Para almacenar en la base de datos ***SQL*** el error se us un ***SP*** de nombre *"Svc_Alm_Err"* , que tiene los mismos parámetros de entrada que el método de la ***DLL*** y retorna el número ID con el cual fue grabado el error. El siguiente seudocódigo ilustra este método:

```
Private Function AlmacenaError (Byval Num_Err as double, _
                                Byval Desc_Err as String, _
                                Byval Nom_Ori_Err as String, _
                                Byval Fec_Err as datetime, _
                                Byval Hor_Err as datetime) as double

'se establece conexión con la base de datos SQL
 'se declaran variables del tipo ADO para trabajar con la base de datos
Dim Conn As ADODB.Connection
Dim Obj As ADODB.Command
Dim rs As ADODB.Recordset
Set Conn = New ADODB.Connection
Set Obj = New ADODB.Command
Set rs = New ADODB.Recordset

'se definen los 5 parámetros necesarios
Dim parametro1 As ADODB.Parameter
...
Dim parametro5 As ADODB.Parameter

Obj.ActiveConnection = Conn
Obj.CommandType = adCmdStoredProc
Obj.CommandText = "Svc_Alm_Err"

'se le asignan los valores a los 5 parámetros
 Set parametro1 = Obj.CreateParameter("num_err", adDouble, adParamInput, 10, Cdbl(Num_Err))
...
```

```
Set parametro5 = Obj.CreateParameter("hor_err", adDBTime, adParamInput, 20, Hor_Err)

'se agregan los 5 parámetros a la llamada del servicio
Obj.Parameters.Append parametro1
...
Obj.Parameters.Append parametro5

'se ejecuta el procedimiento almacenado
Set rs = Obj.Execute

'se le asigna el valor retornado por el SP a la salida del método
AlmacenaError=rs(0)

End Function
```

4.4.4.2 Método que rescata el mensaje para los usuarios y las acciones a seguir

El nombre de este método es *"TraeMensaje"* y tiene como parámetros de entrada los siguientes campos: número del error, origen del error (página o componente) y nombre del origen (nombre de la página ***ASP*** o del método, clase y componente). Para acceder a la información almacenada en la base de datos ***SQL*** se usa un ***SP*** de nombre *"Svc_Lee_Msj_Err"*, que tiene los mismos parámetros de entrada que el método de la ***DLL*** y retorna un registro con información que contiene el mensaje a desplegar para el usuario, las acciones a seguir, si corresponde enviar o no un mensaje, el texto del mensaje a enviar y las direcciones de correo electrónico de los encargados de la aplicación que presentó problemas. El siguiente código ilustra este método:

```
Private Function TraeMensaje (Byval Num_Err as double, _
                              Byval Org_Err as String, _
                              Byval Nom_Ori_Err as String)
                              as ADODB.Recorsdset

    'se establece la conexión con la base de datos SQL
    'se definen variables de tipo ADO

    Dim Conn As ADODB.Connection
    Dim Obj As ADODB.Command
    Dim rs As ADODB.Recordset
    Set Conn = New ADODB.Connection
    Set Obj = New ADODB.Command
    Set rs = New ADODB.Recordset
     'se definen los 3  parámetros necesarios
```

```
Dim parametro1  As ADODB.Parameter
Dim parametro2  As ADODB.Parameter
Dim parametro3  As ADODB.Parameter

‘se determina el SP a ejecutar

Obj.ActiveConnection = Conn
Obj.CommandType = adCmdStoredProc
Obj.CommandText = “Svc_Lee_Msj_Err”

‘se le asignan los valores a los 3 parámetros

 Set parametro1 = Obj.CreateParameter("num_err", adDouble, adParamInput,
10, Cdbl(Num_Err))
…

‘se agregan los 3 parámetros a la llamada del servicio
Obj.Parameters.Append parametro1
…

‘se ejecuta el procedimiento almacenado
Set rs = Obj.Execute

‘se le asigna el valor retornado por el SP a la salida del método
Set TraeMensaje =rs

End Function
```

4.4.4.3 Método que envía mensajes

El nombre de este método es *“EnviaMensaje”* y tiene como parámetros de entrada los siguientes campos: Dirección de correo de los administradores del sitio, cuerpo del mail a enviar, titulo del mensaje a enviar y dirección física del archivo a enviar (si es que se envía, sino se ingresa en blanco). Para poder enviar el correo electrónico a los destinatarios seleccionados se utiliza una componente propia del banco que realiza el envío de mensajería (la componente se llama Internet y la clase Cls_Mail). El siguiente seudocódigo ilustra este método:

```
Private Function EnviaMensaje (Byval Dir_Eml as String, _
                               Byval Cpo_Eml as String, _
                               Byval Tit_Eml as String, _
                               Byval Pth_Arc_Env as String) as Boolean

Dim Objmail as Internet.Cls_Mail
Dim salida as boolean
Set Objmail = New Internet.Cls_Mail
‘se debe procesar la información recibida y crear un correo electrónico
‘con el formato esperado por los administradores
```

```
salida  = Objmail.SendMail(dir_mail,titulo,cuerpo,atach)

‘se le asigna el valor retornado por el SP a la salida del método
EnviaMensaje = salida
End Function
```

4.4.4.4 Método que genera reportes de error

El nombre de este método es *“GeneraReporte”* y tiene como parámetros de entrada los siguientes campos: número del error, descripción del error, origen del error (página o componente), nombre del origen (nombre de la página ***ASP*** o del método, clase y componente), fecha y hora. Para almacenar en la base de datos ***SQL*** el error se usa un ***SP*** de nombre *“Svc_Alm_Err_Rpt”*, que tiene los mismos parámetros de entrada que el método y cuya función será grabar en la tabla temporal destinada para esta función. El siguiente seudocódigo lo ilustra:

```
Public Function GeneraReporte (Byval Num_Err as double, _
                               Byval Desc_Err as String, _
                               Byval Nom_Ori_Err as String, _
                               Byval Fec_Err as datetime, _
                               Byval Hor_Err as datetime) as double

    ‘se establece conexión con la base de datos SQL
    ‘se definen variables de tipo ADO

    Dim Conn As ADODB.Connection
    Dim Obj As ADODB.Command
    Dim rs As ADODB.Recordset
    Set Conn = New ADODB.Connection
    Set Obj = New ADODB.Command
    Set rs = New ADODB.Recordset

    ‘se definen los 5  parámetros necesarios
    Dim parametro1  As ADODB.Parameter
    …
    Dim parametro5  As ADODB.Parameter

    Obj.ActiveConnection = Conn
    Obj.CommandType = adCmdStoredProc
    Obj.CommandText = “Svc_Alm_Err_Rpt”

    ‘se le asignan los valores a los 5 parámetros
     Set parametro1 = Obj.CreateParameter("num_err", adDouble, adParamInput,
    10, Cdbl(Num_Err))
    …
     Set parametro5 = Obj.CreateParameter("hor_err", adDBTime, adParamInput,
    20, Hor_Err)

    ‘se agregan los 5 parámetros a la llamada del servicio
    Obj.Parameters.Append parametro1
    …
```

Obj.Parameters.Append parametro5

‘se ejecuta el procedimiento almacenado
Set rs = Obj.Execute

‘se le asigna el valor retornado por el SP a la salida del método
GeneraReporte=rs(0)

End Function

4.4.4.5 Controlador de errores

Una vez creado todos los métodos que realizan una acción en particular, falta crear un método que se encargue de coordinar todas las acciones necesarias para el correcto funcionamiento del manejador de errores, este método recibirá todos los parámetros proporcionados por la página ***ASP*** o por la ***DLL*** y será el encargado de almacenar el error a través del método *“AlmacenaError”*, luego ingresar este error en la tabla temporal que alimenta al reporte de errores, utilizando para ello el método *“GeneraReporte”*, luego debe entregar un mensaje entendible al cliente además de las acciones a seguir y para ello utiliza el método *“TraeMensaje”* y si corresponde enviar un mensaje a los administradores o encargados de la aplicación con problemas lo hará utilizando el método *“EnviaMensaje”*. Este método es en si, la base del sistema manejador de errores y el siguiente seudocódigo ilustra su funcionamiento.

Public Function ManejaError (Byval Num_Err as double, _
Byval Desc_Err as String, _
Byval Ori_Err as String, _
Byval Nom_Ori_Err as String, _
Byval Fec_Err as datetime, _
Byval Hor_Err as datetime) as String

Dim *NumRegistro* as Double
Dim NumReporte as Double
Dim *Enviomail* as Boolean
Dim rs As ADODB.Recordset
Set rs = New ADODB.Recordset

‘primero debe almacenar el error
NumRegistro=AlamacenaError(Num_Err,Desc_Err,Nom_Ori_Err,Fec_Err,Hor_Err)

‘luego rescato el mensaje para el cliente y las acciones a seguir
Set rs= TraeMensaje (Num_Err ,Ori_Err, Nom_Ori_Err)

‘se pregunta si corresponde o no almacenar este error para generar reporte
IF rs(“reporte”)=”SI” then

```
        NumReporte=GeneraReporte(Num_Err,Desc_Err,Nom_Ori_Err,Fec_Err,Hor_Err)
End if

‘se pregunta si corresponde o no enviar un mensaje
IF rs(“mensaje”)=”SI” then
        Enviomail=EnviaMensaje(rs(“Dirmail”) ,rs(“Cuerpo”),rs(“titulo”),””)
End IF

‘Se devuelve el mensaje al cliente
ManejaError=rs(“mensaje”)

End Function
```

4.5 Sistema *log* transaccional

El sistema de ***log*** transaccional es el encargado de registrar cada uno de los movimientos que realice un cliente en el sitio web transaccional de ***BECH***. Es decir, cada vez que el cliente realice una operación en el sitio web transaccional, ésta quedará registrada en un repositorio de datos que para efectos del banco se tratará de una base de datos ***SQL***, este mismo repositorio será la base de las cuales se extraerán los movimientos históricos de los clientes que estarán a su disposición cada vez que ellos lo estimen conveniente y a la vez permite a los administradores del sitio poder gestionar la información recopilada.

Como el sistema de ***log*** transaccional además de grabar en una base ***SQL*** las entradas y salidas de cada servicio de datos que se llame con el fin de identificar los errores ocurridos en esa instancia, debe también poder ser instanciado desde una componente o de una página web dinámica, por lo tanto se requiere que éste sea en si una componente.

4.5.1 Interacción con las entidades relacionadas

Como se definió en el capitulo anterior el sistema de ***log*** transaccional debe interactuar con las páginas dinámicas y componentes.

Las páginas ***ASP*** deben entregar al sistema de ***log*** transaccional toda la información correspondiente a una correcta operación realizada por algún cliente del banco, para que el sistema lo almacene en la base ***SQL*** destinada para ello. Los parámetros que debe entregar la página son:

- Rut del cliente
- Acción realizada (consulta de saldo, pago, transferencia, etc.)
- Productos involucrados
- Cuentas involucradas
- Montos transferidos o consultados
- Datos relevantes del producto
- Fecha y hora de la consulta o transferencia

Las componentes que utilicen servicios de datos deben proporcionar al sistema de ***log*** transaccional la entrada a estos servicios y las salidas que éstos retornen, para que sea el sistema quien almacene en la tabla ***SQL*** creada para este objeto, al tener esta información se podrá saber de posibles errores ocurridos en los servicios de datos. Los parámetros que le debe pasar la componente al sistema de ***log*** son los siguientes:

- Nombre del servicio
- Cadena de entrada al servicio
- Salida del servicio
- Método, clase y ***DLL*** de la cual se llamó al servicio
- Fecha y hora de la llamada al servicio

4.5.2 Definición del funcionamiento del sistema de *log* transaccional

El sistema de ***log*** transaccional debe ser una ***DLL***. Esta componente recibe como parámetros de entrada todos los datos necesarios para almacenar

la transacción realizada por un cliente en particular, como también los necesarios para almacenar las entradas y salidas de los servicios de datos utilizados en el sitio web de ***BECH*** y también debe entregar los movimientos realizados por un cliente en particular en un periodo de tiempo dado. Los repositorios de datos en los cuales se almacenaran los movimientos de los clientes junto con las entradas y salidas de los servicios de datos serán bases de datos ***SQL***. Cada una de las acciones a realizar por esta componente debe ser definida como un método dentro de una o varias clases.

4.5.2.1 Almacenamiento de la transacción

Para almacenar la transacción dentro de una tabla ***SQL*** destinada para ello debe hacerse vía un ***SP*** el cual debe recibir como parámetros los siguientes campos:

- rut del cliente
- acción realizada (consulta de saldo, pago, transferencia, etc.)
- productos involucrados
- cuentas involucradas
- montos transferidos o consultados
- Datos relevantes del producto
- Fecha y hora de la consulta o transferencia.

Una vez que el ***SP*** reciba los parámetros proporcionados por la componente de ***log*** transaccional deben ser registrados en la tabla de la base ***SQL*** destinada para ello y retornar el Número o ID con el cual fue grabada la transacción.

4.5.2.2 Almacenamiento de la entrada y salida de los servicios de datos

Para almacenar las entradas y salidas de un servicio de datos dentro de una tabla ***SQL*** se debe utilizar un ***SP*** que reciba como parámetros los siguientes campos: nombre del servicio, cadena de entrada al servicio, salida del servicio, método, clase y ***DLL*** de la cual se al servicio y la fecha y hora de la llamada.

Una vez que el ***SP*** reciba los parámetros proporcionados por la componente de ***log*** transaccional los debe registrar en la tabla de la base ***SQL*** y retornar el Número o ID con el cual fue grabado.

4.5.2.3 Lectura de los movimientos realizados por un cliente

Para poder acceder a las transacciones realizadas por un cliente en el sitio web transaccional de ***BECH*** se debe utilizar un ***SP*** que reciba como parámetros los siguientes campos: rut del cliente, fecha de Inicio para la búsqueda y fecha de termino para la búsqueda, y con éstos datos acceda a la tabla ***SQL*** en donde se almacenan los movimientos de todos los clientes.

Una vez que el ***SP*** reciba los parámetros proporcionados por la componente de ***log*** transaccional, debe realizar la búsqueda en la tabla ***SQL*** según los criterios ingresados, y si hay coincidencia de datos retornar todas las transacciones realizadas por el cliente, para ello deberá retornar la acción realizada (consulta de saldo, pago, transferencia, etc.), productos involucrados, cuentas involucradas, montos transferidos o consultados, datos relevantes del producto y fecha y hora de la consulta o transferencia.

4.5.3 Implementación del sistema de *log* transaccional

Una vez detallada la funcionalidad del sistema de ***log*** transaccional se puede iniciar su implementación. El sistema de ***log*** a implementar tiene una

sola clase en la cual se irán agregando los métodos en la medida de que sean necesarios. El nombre del Sistema ***log*** es *"LOG"* y la clase se llama *"Cls_Transacciones"*.

4.5.3.1 Método que almacena una transacción

El nombre de este método es *"AlmacenaTransaccion"* y tiene como parámetros de entrada los siguientes campos: rut del cliente, acción realizada (consulta de saldo, pago, transferencia, etc.), productos involucrados, cuentas involucradas, montos transferidos o consultados, datos relevantes del producto y fecha y hora de la consulta o transferencia. Para almacenar en la base de datos ***SQL*** la transacción se usará un ***SP*** de nombre *"Svc_Alm_Tra_Log"*, que tendrá los mismos parámetros de entrada que el método de la ***DLL*** y retornará el número ID con el cual fue grabada la transacción. El siguiente seudocódigo ilustra este método:

```
Public Function AlmacenaTransaccion (Byval Rut_Cli as String, _
                                     Byval Acc_Rlz as String, _
                                     Byval Nom_Prod_Err as String, _
                                     Byval Cuenta as string, _
                                     Byval Monto Sting) as double

    'se establece conexión con la base de datos SQL
    'se declaran variables del tipo ADO
    Dim Conn As ADODB.Connection
    Dim Obj As ADODB.Command
    Dim rs As ADODB.Recordset
    Set Conn = New ADODB.Connection
    Set Obj = New ADODB.Command
    Set rs = New ADODB.Recordset

    'se definen los 5 parámetros necesarios
    Dim parametro1 As ADODB.Parameter
    ...
    Dim parametro5 As ADODB.Parameter

    Obj.ActiveConnection = Conn
    Obj.CommandType = adCmdStoredProc
    Obj.CommandText = "Svc_Alm_Tra_Log"
    'se le asignan los valores a los 5 parámetros
    Set parametro1 = Obj.CreateParameter("Rut_cli", advarchar, adParamInput, 10,
    Rut_cli)
    ...
    Set parametro5 = Obj.CreateParameter("Mto_inv", adDouble, adParamInput,
    20, Monto)
```

```
        'se agregan los 5 parámetros a la llamada del servicio
        Obj.Parameters.Append parametro1
        ...
        Obj.Parameters.Append parametro5

        'se ejecuta el procedimiento almacenado
        Set rs = Obj.Execute

        'se le asigna el valor retornado por el SP a la salida del método
        AlmacenaTransaccion=rs(0)
End Function
```

4.5.3.2 Método que almacena la entrada y salida de un servicio de datos

El nombre de este método es *"AlmacenaServicioDato"* y tiene como parámetros de entrada los siguientes campos: nombre del servicio, cadena de entrada al servicio, salida del servicio, método, clase y ***DLL*** de la cual se llamo al servicio y la fecha y hora de la llamada al servicio. Para almacenar en la base de datos ***SQL*** la entrada y salida del servicio de datos se usará un ***SP*** de nombre *"Svc_Alm_Srv_Dat"*, que tiene los mismos parámetros de entrada que el método de la ***DLL*** y retorna el número ID con el cual fue grabada la información correspondiente al servicio de datos. El siguiente seudocódigo ilustra este método:

```
Public Function AlmacenaServicioDato (Byval Nom_Srv as String, _
                                      Byval Entrada as String, _
                                      Byval Salida as String, _
                                      Byval Nom_Dll as string, _
                                      Byval Clase_Metodo Sting) as double

        'se establece conexión con la base de datos SQL
        'se declaran variables del tipo ADO
        Dim Conn As ADODB.Connection
        Dim Obj As ADODB.Command
        Dim rs As ADODB.Recordset

        Set Conn = New ADODB.Connection
        Set Obj = New ADODB.Command
        Set rs = New ADODB.Recordset

        'se definen los 5 parámetros necesarios
        Dim parametro1 As ADODB.Parameter
        ...
        Dim parametro5 As ADODB.Parameter
        Obj.ActiveConnection = Conn
        Obj.CommandType = adCmdStoredProc
        Obj.CommandText = "Svc_Alm_Srv_Dat"

        'se le asignan los valores a los 5 parámetros
        Set parametro1 = Obj.CreateParameter("Nom_Srv", advarchar, adParamInput,
        15, Nom_Srv)
```

```
…
Set parametro5 = Obj.CreateParameter("Cls_Mtd", advarchar, adParamInput,
50, Clase_Metodo)

‘se agregan los 5 parámetros a la llamada del servicio
Obj.Parameters.Append parametro1
…
Obj.Parameters.Append parametro5

‘se ejecuta el procedimiento almacenado
Set rs = Obj.Execute

‘se le asigna el valor retornado por el SP a la salida del método
AlmacenaServicioDato=rs(0)
End Function
```

4.5.3.3 Método que lee los movimientos realizados por un cliente

El nombre de este método es *“TraeMovimientos”* y tiene como parámetros de entrada los siguientes campos: rut del cliente, fecha de Inicio para la búsqueda y fecha de termino para la búsqueda. Para leer en la base de datos ***SQL*** los movimientos realizados por un cliente en un periodo de tiempo dado se usa un ***SP*** de nombre *“Svc_Lee_Mov_Rlz”*, que tiene los mismos parámetros de entrada que el método de la ***DLL*** y retornar todas las transacciones realizadas por el cliente, para ello debe entregar la acción realizada (consulta de saldo, pago, transferencia, etc.), productos involucrados, cuentas involucradas, montos transferidos o consultados, datos relevantes del producto y fecha y hora de la consulta o transferencia. El siguiente seudocódigo ilustra este método:

```
Public Function TraeMovimientos    (Byval Rut_Cli as String, _
                                    Byval Dv_Rut_Cli as String,)
                                     as ADODB.Recordset

    ‘se establece conexión con la base de datos SQL
    ‘se declaran variables del tipo ADO
    Dim Conn As ADODB.Connection
    Dim Obj As ADODB.Command
    Dim rs As ADODB.Recordset
    Set Conn = New ADODB.Connection
    Set Obj = New ADODB.Command
    Set rs = New ADODB.Recordset

    ‘se definen los 2  parámetros necesarios
    Dim parametro1  As ADODB.Parameter
    Dim parametro2  As ADODB.Parameter
```

```
        Obj.ActiveConnection = Conn
        Obj.CommandType = adCmdStoredProc
        Obj.CommandText = "Svc_Lee_Mov_Rlz"

        'se le asignan los valores a los 2 parámetros
        Set parametro1 = Obj.CreateParameter("num_rut_cli", addouble,
        adParamInput, 8, Rut_Cli)
        Set parametro2 = Obj.CreateParameter("dv_rut_cli", advarchar, adParamInput,
        1, Dv_Rut_Cli)

        'se agregan los 2 parámetros a la llamada del servicio
        Obj.Parameters.Append parametro1
        Obj.Parameters.Append parametro2

        'se ejecuta el procedimiento almacenado
        Set rs = Obj.Execute

        'se le asigna el valor retornado por el SP a la salida del método
        Set TraeMovimientos=rs(0)
End Function
```

4.6 Ejemplo integrado

Una vez implementadas teóricamente las soluciones a los cuatro problemas principales del actual sitio web de ***BECH***, es decir, una vez que se han creado el motor de presentación, motor de negocio, manejador de errores y sistema ***log***, hay que ver el comportamiento de estas cuatro soluciones en la práctica, para ello se ejemplificará con un caso común para los clientes de ***BECH***, la solicitud en línea de una cartola de cuenta corriente, y para ello se ilustrará el seudocódigo de la páginas ***ASP*** encargada de realizar esta acción.

El seudocódigo en VBScript que realiza esta acción utiliza a las cuatro componentes ya creadas, y se ilustra a continuación:

```
<%
'se instancias los objetos a utilizar

Set presentacion=Server.CreateObject("Motor_Presentacion.Cls_Presentacion")
Set negocio=Server.CreateObject("Motor_Negocios_Logica.Cls_Logica")
Set datos=Server.CreateObject(("Motor_Negocios_Datos.Cls_Datos")
Set errores=Server.CreateObject("Controlador.Cls_Errores")
Set registro=Server.CreateObject("LOG.Cls_Transacciones")

'se rescatan los parámetros de entrada

tipoprod=request("tipo_prod")
tiposerv=request("tipo_serv")
Numprod=request(num_producto")
RutCliente=request("rut")
```

```
DvCliente=request("dv")

' se genera cadena de entrada para saldo cuenta corriente
cadena=RutCliente & DvCLiente & Numprod

'se ejecuta el servicio de datos para obtener el saldo, el tipo de base es CICS y
'el nombre del servicio es SAIE020
salida=datos.Ejecuta_Serv_Datos("CICS","SAIE020",cadena)

'Se definen variable de tipo XML
Dim ObjRaizXml As MSXML2.FreeThreadedDOMDocument40

'se obtiene el formato del XML a utilizar, donde
'SaldoCCT indica saldo de cuenta corriente
StrXml=negocio.Obtener_formato_XML("SaldoCCT")

 'se genera el XML con los datos obtenidos del servicio
cadena_Xml=negocio.Generar_XML_respuesta(salida, StrXml)

 'se carga la cadena de salida  en formato XML para trabajarla como archivo XML
ObjRaizXml.async = True
ObjRaizXml.loadXML (cadena_Xml)

'se verifica si hubo error o no
StrError = ObjRaizXml.getElementsByTagName("CODERROR").text

If StrError<>0 then  'hubo error
        DescError = ObjRaizXml.getElementsByTagName("DESCERROR").text
        Msjcliente=errores.ManejaError(StrError, DescError,"ASP","saldoCCT",date,time)
        'se muestra el mensaje al cliente
        Response.write MuestraMensajes(Msjcliente)
        'se detiene la ejecucion del programa
        Response.end
End if

'se selecciona el nombre de la etiqueta de salida para leer el archivo XML
Set ObjHeader = ObjRaizXml.selectNodes("SALIDA")

'se almacena esta transaccion en el sistema de log
numregistro=registro.AlmacenaTransaccion(RutCliente,"Saldo","CCT",numprod,"0")

'se obtiene la cadena con formato HTML para presentarla al cliente
cadenaHTML=presentacion.Presenta_Informacion("Saldo","CCT", cadena_Xml)

'se escribe el HTML
response.write cadenaHTML

%>
```

4.7 Manejador de fuentes

Como se especificó en el punto 3.6.3.1, ***BECH*** tiene muchos problemas en sus desarrollos de aplicaciones web, producto de la ausencia de un sistema manejador de códigos fuentes.

Dentro de las alternativas de sistemas manejadores de códigos fuentes existentes en el mercado, y que pueden ser compatibles con las herramientas de desarrollo que utiliza ***BECH***, se tienen:

- **Visual Source Safe:** Es el sistema de administración de código fuente y control de versiones más eficiente y productivo para equipos de desarrollo que utilizan .NET. Los desarrolladores pueden administrar el código fuente, contenido Web y cualquier otro tipo de archivo de manera amigable y segura. Este producto forma parte del paquete de Visual Studio 7.0 (también llamado Visual Studio .NET), aunque su potencia es óptima con las herramientas de desarrollo de .NET, también da soporte a las versiones anteriores de Microsoft y a otras herramientas no pertenecientes a esta compañía, aunque no logra su mayor rendimiento
- **Rational ClearCase:** Es una herramienta confiable, de control de versiones, posee varias versiones, partiendo por la básica, diseñada para pequeños grupos de trabajo, hasta llegar a la versión empresarial diseñada para grandes grupos de desarrollo en empresas globalmente distribuidas. Provee soporte excepcional para el desarrollo en paralelo, permitiendo que múltiples desarrolladores diseñen, codifiquen, prueben y optimicen software eficientemente a partir de una base común de código.
- **AllFusion Harvest Change Manager :** Este producto conocido antes como "CCC/Harvest" permite que el manejo, sincronización y los controles de cambios en sistemas cliente-servidor y aplicaciones web, se hagan bajo las más estrictas normas de calidad, controles de seguridad y de forma automática. Es una de

las soluciones más completas para la gestión de la configuración y el cambio de entornos distribuidos.

Cualquiera de las tres alternativas propuestas pueden ser utilizadas por el área de desarrollo de aplicaciones web de ***BECH*** para acabar con los problemas presentados por el nulo manejo de los códigos fuentes de sus aplicaciones web, tanto páginas dinámicas como componentes. Las tres alternativas utilizan bases de datos para guardar la información correspondiente a las distintas versiones de las aplicaciones desarrolladas, todas permiten diferenciar a los usuarios hacen uso de ellas, con el fin de poder asignar privilegios a ciertos usuarios (administradores, jefes de proyecto, etc.) solo la interfaz y la lógica del software difiere un poco, siendo la opción “AllFusion Harvest Change Manager” la de menor complejidad para los usuarios y con una interfaz bastante más amigable.

CAPITULO CINCO

ESTUDIO DE PLATAFORMAS

5. Estudio de plataformas.

En este capítulo se utiliza el concepto de plataforma entendido como un número de reglas o estándar sobre los cuales se puede desarrollar un sistema informático. A partir de esta definición fueron objeto de estudio las plataformas con mayor participación en el mercado que pudieran dar soporte a un sitio web transaccional como el que se espera.

Se investigaron los dos tipos de servidores a considerar al momento de diseñar un sitio web, enumerados a continuación:

- Servidores Web, encargado exclusivamente de la presentación en el cliente.
- Servidores de aplicación, encargados de la conexión entre capa de datos, negocio y eventualmente presentación.

Dentro de la categoría de los servidores de aplicación el mercado actualmente presenta dos grandes opciones:

- Alternativa J2EE(*Java 2 Enterprise Edition*) de Sun.
- Alternativa Microsoft .NET

J2EE es un estándar de código abierto promocionado por Sun, sobre el cuál existen servidores de aplicación implementados por terceros. En cambio Microsoft .NET es un producto propietario que trabaja fuertemente ligado con tecnologías Microsoft.

Estas alternativas fueron el objeto principal de estudio a lo largo de este capítulo, ya que en los servidores de aplicación descansa la tecnología para implementar un sito web como el que espera ***BECH***.

5.1 Servidor web.

Un servidor web es, esencialmente, un programa que responde páginas ***HTML*** a partir de una solicitud realizada por un cliente. Cuando se detecta una

petición el servidor espera que la petición involucrada este bajo un formato adecuado. El recurso pedido hace referencia a un archivo en el disco duro que el servidor es capaz de recuperar y enviar al cliente. Tanto la petición como la respuesta se encapsulan siguiendo el protocolo ***HTTP***. La arquitectura representada por la figura *Cap5_1*, se basa en el modelo cliente-servidor y cada parte está típicamente en una máquina distinta de la red, aunque no hay incoveniente si residen en la misma máquina.

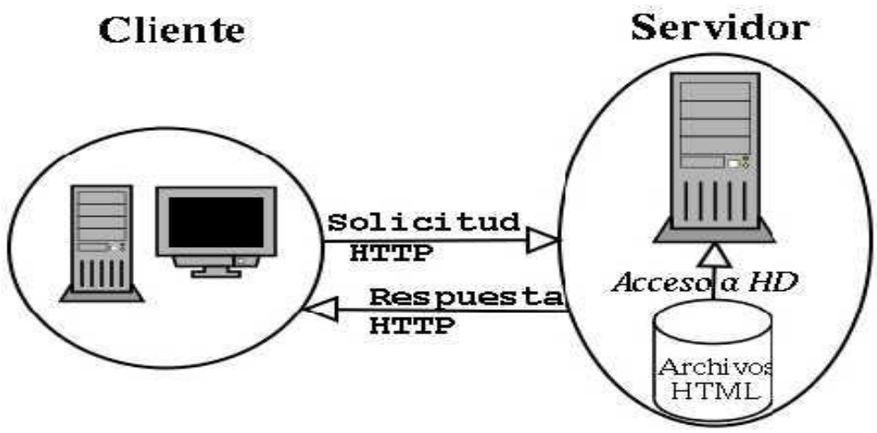


Figura Cap5_1, Arquitectura de funcionamiento de un servidor web

5.2 Servidor de aplicación.

Un servidor de aplicación ejecuta la lógica existente entre el cliente final y los servidores de bases de datos. Por ejemplo, cuando un cliente accede a un sitio web por medio de un navegador es un servidor web el que envía dicha solicitud al servidor de aplicación que ejecuta la lógica, recupera y actualiza los datos del cliente desde las fuentes finales. El servidor de aplicación ejecuta los programas de negocio en lugar del cliente, del servidor web o de otros sistemas.

Físicamente un servidor de aplicación separa la lógica de negocio de los datos, dentro de una arquitectura multi-capa. Es esta característica la que permite desarrollar y desplegar aplicaciones en forma rápida y fácil sin

necesidad de mayor programación, debido a que no es necesario modificar ninguna otra entidad, solo el servidor de aplicación.

Los servidores de aplicaciones se desarrollan bajo el objetivo de tener aplicaciones de misión crítica constantemente disponibles para un número creciente de clientes. Adicionalmente estas aplicaciones necesitan ser seguras y fiables sin importar el número de personas que acceden al sistema o a la fuente de datos, el servidor de aplicaciones siempre debiera estar activo. Anteriormente a los servidores de aplicaciones, las aplicaciones web se ejecutaban frecuentemente sobre servidores web que estaban realmente diseñados para servir páginas web por lo que ejecutar y desarrollar aplicaciones se hacía lento y complejo.

Un servidor de aplicaciones clásico, como el representado en la figura *Cap5_2*, se apoya en un modelo cliente-servidor de tres capas:

1. **Presentación:** una interfaz, generalmente gráfica que reside en los clientes, típicamente como un un navegador.
2. **Lógica de negocio:** donde reside el servidor de aplicaciones y el conjunto de programas a los que da soporte.
3. **Almacenamiento:** generalmente una base de datos.

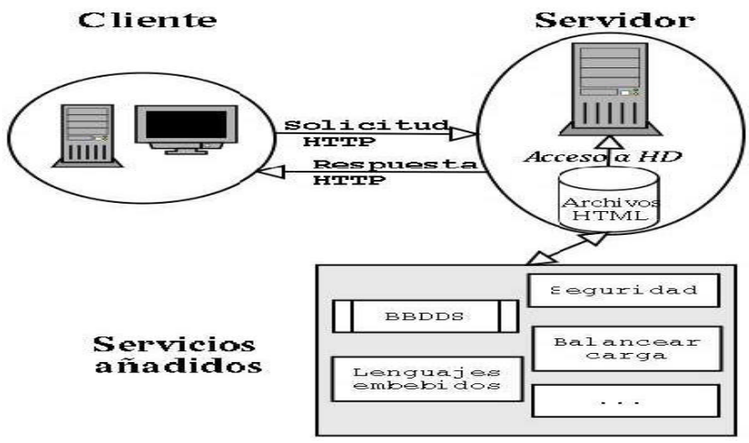


Figura Cap5_2, Arquitectura de funcionamiento de un servidor de aplicaciones

5.2.1 Funcionalidades adicionales de un servidor de aplicación.

En general se puede afirmar que cuantos más puntos de la siguiente lista sean implementados por un servidor de aplicaciones mejor será la funcionalidad del mismo.

- **Generación de HTML:** debe incorporar generación dinámica de contenido, en formatos ***HTML***, ***XML***, etc, para enviar al cliente.
- **Trabajo con bases de datos:** debieran existir objetos que faciliten el acceso a bases de datos, ocupándose de gestionar las conexiones y proporcionando un acceso uniforme.
- **Funcionamiento multiproceso o multihilo:** el servidor es el responsable de tener funcionando un número de hilos o procesos que atiendan a distintas peticiones.
- **Sesiones:** debido a que ***HTTP*** es un protocolo sin estados, un servidor de aplicaciones provee de persistencia a los datos del usuario mediante objetos de sesión, esto elimina la necesidad de incluir código en las aplicaciones para diferenciar las peticiones de distintos usuarios.
- **Lógica de negocio:** la lógica de negocio propia de cada aplicación debe poder ser encapsulada en componentes. A cada uno de ellos se le podrán asignar mecanismos propios de seguridad y gestión de transacciones.
- **Seguridad:** debe poseer características de seguridad que den soporte a aplicaciones seguras. Los clientes deben autenticarse contra al servidor, y es éste el responsable de darles acceso a sus diferentes componentes, como puede ser una base de datos.
- **Balanceo de carga:** trabajando sobre un *cluster* de servidores, puede enviar las peticiones a diferentes equipos en función de la carga y la disponibilidad. Este balanceo es la base para implementar sistemas

tolerantes a fallos o herramientas para la monitorización centralizada de todos los equipos del *cluster*.

5.2.2 Tecnología para implementar servidores de aplicaciones.

Al momento de implementar un servidor de aplicaciones las tecnologías que lideran el mercado son aquellas que cumplen completamente con el estándar J2EE, que lo cumplen parcialmente (no J2EE) o que usen tecnología Microsoft .NET.

5.2.2.1 Alternativa J2EE.

La estrategia comercial de Sun con respecto a Java y a su nueva plataforma *J2EE* está teniendo mucho éxito en este sector. De hecho, a veces se utiliza el término: *servidores de aplicaciones Java* para referirse a aquellos servidores de aplicaciones que implementan adecuadamente las soluciones propuestas por J2EE.

J2EE es una especificación que propone un estándar para servidores de aplicaciones. Define diferentes tecnologías e indica como deben trabajar juntas. Todos los servidores de aplicaciones que quieran ser etiquetados como servidores de aplicaciones *J2EE* deben pasar un test de compatibilidad, que garantiza la correcta implementación de las tecnologías Java.

5.2.2.2 Alternativa No-J2EE.

Actualmente el mercado se divide entre las opciones J2EE y Microsfot .NET, pero antes de llegar a este estado existían otras opciones de servidores de aplicaciones. Dos son los ejemplos más claros, por un lado PHP y por otro Coldfusion de Allaire-Macromedia.

- PHP (Hypertext Preprocessor ó Preprocesador de Hipertexto) es un lenguaje que permite crear aplicaciones web con código abierto, en el lado del servidor, ocupando lenguaje scripting para ello
- Allaire-Macromedia Coldfusion es un programa comercial basado en un lenguaje propietario de etiquetas diseñado para realizar consultas sobre bases de datos en forma dinámica. Hasta la versión 4 era independiente de J2EE, pero la última versión ya integra dicha tecnología

5.2.2.3 Alternativa Microsoft.

Microsoft va por camino separado, si se opta por su servidor de aplicaciones, se está obligado a utilizar la plataforma Microsoft completa.

Las primeras soluciones que ofreció esta empresa se basaban en el servidor web ***IIS***, el lenguaje de script ***ASP*** y la tecnología de objetos distribuidos ***COM,*** en lo que se denominó estándar ***DNA***. La nueva propuesta es Microsoft *.NET* que permite ser progrmado en distintos lenguajes como C#, VB, etc. Todas estas soluciones siguen la política habitual de Microsoft que tiende a apoyarse en las entrañas de Windows y obviar estándares abiertos, pero hay alguna excepción, por ejemplo en la plataforma .NET se incluye soporte a SOAP.

5.3 Estándar para servidores de aplicación J2EE.

J2EE es un estándar para el desarrollo de aplicaciones multi-capa que sean escalables, de alta disponibilidad, seguras y eficientes.

Esta plataforma permite a los desarrolladores enfocarse en la lógica de negocio mientras que J2EE maneja los detalles de bajo nivel. Con J2EE, los

servicios son fácilmente mejorables y rápidamente desarrollados, permitiendo a los negocios reaccionar rápidamente ante los cambios competitivos.

5.3.1 Modelo del estándar J2EE.

El modelo de aplicaciones J2EE proporciona un acercamiento para el desarrollo de aplicaciones de alta disponibilidad y alta escalabilidad. Una característica importante de este modelo es que las aplicaciones que se pueden desarrollar sobre él están libres de complejidades de bajo nivel como administración de transacciones, protocolos de seguridad, etc. para poder enfocarse en la lógica de negocio y en las interfaces a presentar al usuario.

En la figura *Cap5_3*, se puede distinguir la presentación en el cliente, la presentación en el servidor, la lógica de negocios en el servidor y el área de datos, el servidor de aplicaciones corresponde en la figura a la plataforma J2EE.

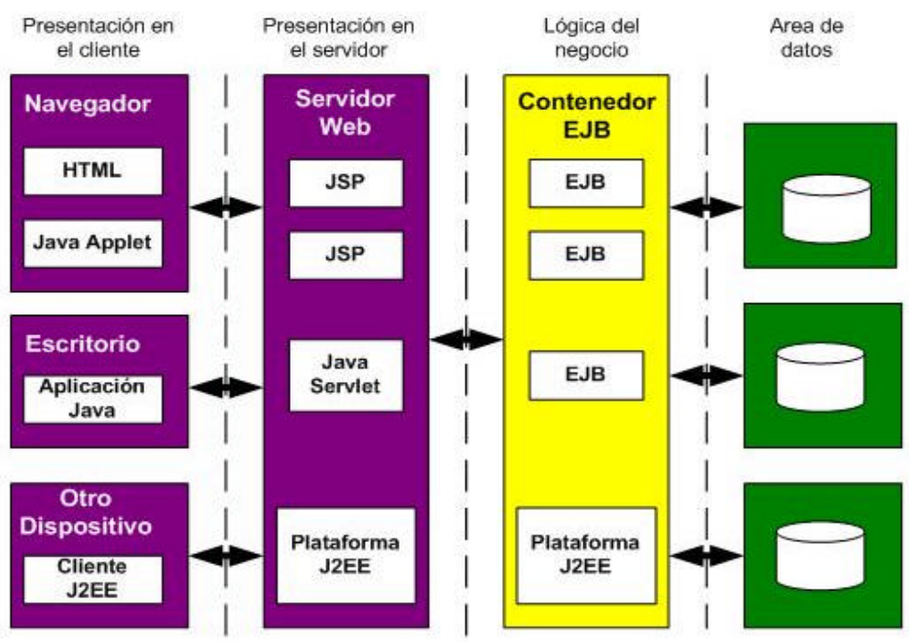


Figura Cap5_3, Modelo estándar J2EE

- **Presentación en el cliente:** corresponde a la interacción con los clientes por medio de un browser en cuyo caso, se da a través de: páginas web con ***HTML*** plano, páginas web con applets Java ó por medio de una aplicación de escritorio en cuyo caso se trata de aplicaciones Java solitarias.
- **Presentación en el servidor:** corresponde al servidor web, dentro del cuál se cuenta con ***JSP*** y Java Servlets para hacer de puente entre el cliente y la información solicitada.
- **Lógica de negocios en el servidor:** por medio de los componentes ***EJB*** se realizan consultas a los sistemas de información de la empresa, procesando las respuestas obtenidas para entregarlas al componente de presentación en el servidor que solicitó la información.
- **Area de datos:** corresponde a las bases de datos y sistemas propios de la empresa que concentran la información de negocio.

El modelo J2EE corresponde a un modelo multi-capa, la primera capa es el cliente que normalmente es un navegador web o una aplicación Java solitaria. Este invoca a la lógica del negocio de una o más capas medias, que a su vez acceden a los datos desde el sistemas de información de la empresa, que corresponde a los datos del negocio, en la tercera capa.

Desarrollar un servicio multi-capa requiere aplicaciones cliente, lógicas de negocio y de presentación junto con código de infraestructura (framework en inglés). La infraestructura corresponde a componentes de bajo nivel del sistema que acceden a variadas bases de datos, recursos del sistema y que proporcionan seguridad. Los detalles de la infraestructura son manejados por J2EE de tal forma que el desarrollador sólo necesite desarrollar las lógicas de negocio y presentación.

En la capa media, la lógica de negocio se implementa como componentes ***EJB***, mientras que la lógica de presentación se implementa como

JSP y Servlets. Los Servlets y los ***JSP*** permiten la separación del procesamiento de la solicitud de su lógica de presentación. La capa de presentación del modelo permite fácilmente acceder a las funciones de negocio de la capa media. La tecnología ***JSP*** permite a los desarrolladores presentar páginas web creadas dinámicamente. Los servlets permiten a los desarrolladores crear presentaciones dinámicas para los usuarios completamente en lenguaje Java.

5.3.2 Componentes del estándar J2EE.

Las aplicaciones J2EE están hechas de componentes. Un componente J2EE es una unidad funcional autónoma que esta montada dentro de una aplicación J2EE junto con sus librerías y archivos asociadas que a la vez se relaciona con otros componentes. El estándar J2EE define los siguientes componentes:

- Aplicaciones cliente y *applets,* son componentes que se ejecutan en el cliente.
- Java Servlet y ***JSP***, son componentes que se ejecutan en el servidor web.
- Enterprise JavaBeans, son componentes de negocio que se ejecutan en el servidor

5.3.2.1 Java Servlets.

Los Servlets son programas que manejan la lógica de presentación de una aplicación actuando como un despachador central para aplicaciones, procesando la entrada, llamando a componentes de la lógica de negocio, accediendo a ***EJB***, y dando formato a la salida. Los servlets controlan el flujo de la aplicación a partir de una interacción directa con el usuario.

Los servlets son como los ***applets*** excepto en que se ejecutan en el servidor en vez de en el cliente.

5.3.2.2 JavaServer Pages.

Los ***JSP*** son programas que permiten insertar código servlet directamente en un documento de texto, a la vez una página ***JSP*** es un documento de texto que contiene dos tipos de texto: plantillas de datos estáticos, los cuales pueden ser expresados en cualquier formato de texto, tales como ***HTML***, ***XML*** y elementos ***JSP***, los cuales determinan como la página construye en forma dinámica el contenido.

Pueden realizar procesos complejos, salidas condicionales, y comunicarse con otros objetos de una aplicación.

Los ***JSP*** se compilan en servlets, cuando son instaladas o cuando son llamadas por primera vez. Esto pone a los ***JSP*** a disposición del entorno de aplicación como objetos estándares y les permite ser llamadas desde un cliente usando una ***URL***.

Se puede pensar en los Servlets y las ***JSP***s como las caras opuestas de la misma moneda: cada una puede realizar las tareas del otro. Sin embargo, como las ***JSP***s están escritas en ficheros ***HTML***, con código Java embebido, son la mejor elección para las tareas de dibujo y presentación. Los servlets son la mejor elección como despachadores centrales para solicitudes de ***JSP*** entrantes.

5.3.2.3 Enterprise JavaBeans.

Un componente ***EJB*** es un programa que contiene campos y métodos para implementar la lógica de negocio, pero sin proporcionar presentación

visible para el usuario. Pueden ser usados por si solos o en conjunto con otros ***EJB*** para ejecutar la lógica de negocio en el servidor J2EE

Los ***EJB***s permiten dividir la lógica de negocio, la reglas, y los objetos en unidades discretas, modulares y escalables. Cada ***EJB*** encapsula una o más tareas de la aplicación, incluyendo estructuras de datos y métodos que operan sobre ellas. Típicamente, los ***EJB***s también reciben parámetros y envían valores de retorno. Los ***EJB***s siempre funcionan dentro del contexto de un "contenedor", que sirve como un enlace entre los ***EJB***s y los servidores que los hospedan.

Hay tres tipos de ***EJB***: los de sesión, los de entidad y los de mensajería.

Los de entidad corresponden a programas que una vez creados continúan dando servicio hasta que son borrados en forma explícita, es decir son permanentes. A la vez están basados en red, es decir pueden ser ocupados por cualquier programa de la red interna en forma remota.

Los de sesión representan servicios asociados a un cliente en particular, que mantiene automáticamente el estado entre múltiples llamadas a métodos del cliente.

Finalmente los ***EJB*** de mensajería permiten que las aplicaciones J2EE procesen mensajes en forma asíncrona, en conjunto con ***JMS***, es decir no trabajan en línea, lo que permite comunicarse con servicios hospedados en otras insituciones.

5.3.3 Principales *API* dentro del estándar J2EE.

Una ***API*** corresponde a un conjunto de rutinas, protocolos y herramientas para la construcción de software, que están ligadas al sistema operativo ó plataforma al cuál pertenecen. El estándar J2EE proporciona un conjuto de ***API***s, las principales se detallan a continuación:

- **JDBC :** permite invocar comandos ***SQL*** desde programas Java. Se usa en un ***EJB*** cuando se sobrepasa la carácterística de persistencia ó se tiene un bean de sesión accesando a una base de datos. Se puede usar desde un servlet o una página ***JSP*** para acceder directamente a una base de datos sin pasar por un ***EJB***.

  Cuenta de dos partes: una interfaz de nivel aplicativo usada por los componentes de aplicación para acceder a la base de datos, y una una interfaz proveedora de servicios para adjuntar un manejador ***JDBC*** dentro de la plataforma J2EE.

- **JMS :** permite que las aplicaciones creen, envien, reciban y lean mensajes. Define un conjuto de interfaces en común y les otorga la semántica necesaria que permite a los programas escritos en comunicarse con otras implementaciones de mensajería.

  JMS minimiza el conjunto de conceptos que un programador debe aprender para usar productos de mensajería, sin embargo provee suficientes características para que se puedan implementar aplicaciones de mensajería sofisticadas. Permite comunicaciones que son:

  - Asíncronas: un proveedor ***JMS*** puede entregar mensajes a un cliente mientras los recibe; un cliente no necesita solicitar los mensajes en orden para recibirlos.
  - Confiables: asegura que un mensaje es entregado solo una vez, aunque permite menores niveles de confiabilidad para aplicaciones que puedan permitirse la perdida de un mensaje o de recibirlos en forma duplicada.

- **JNDI** (Java Naming and Directory Interface, Interfaz Java para Nombres y Directorios) : proporciona funcionalidades para el manejo de nombres y directorios, permite almacenar y recuperar cualquier tipo de objeto Java.

- **JTA** (Java Transaction API, API para Transacciones Java) proprciona una interfaz estándar para el manejo de transacciones, provee un control por defecto para manejar los envíos y restauraciones de las transacciones. Esto significa que cualquier aplicación que este accediendo a una base de datos, es capaz de acceder a los datos acutalizados despues de cada operación de lectura o escritura. En el caso de consultas a dos bases de datos en forma separada, pero que dependen entre si, se hace necesario el uso de ***JTA***, para delimitar donde la transacción comienza, donde realiza los rollbak y los commit.
- **RMI**, abreviatura de Remote Method Invocation (Invocación de métodos remotos), un conjunto de protocolos que capacita objetos Java para que se comuniquen remotamente con otros objetos Java.

5.3.4 Funcionamiento de la plataforma J2EE.

Los componentes ya descritos no se preocupan de codificación de bajo nivel, sino que dejan esta responsabilidad a los contenedores, para cada tipo de componente existe un determinado contenedor. Los contenedores son la interfaz entre un componente y las funcionalidades especificas de bajo nivel de la plataforma J2EE (***API***), antes que un componente cualquiera sea ejecutado pasa por el control de su contenedor.

Existen tres tipos de contenedores: contendor de aplicaciones cliente, contenedor Web y contenedor EJB. Todos ellos administran la ejecución de sus respectivos componentes.

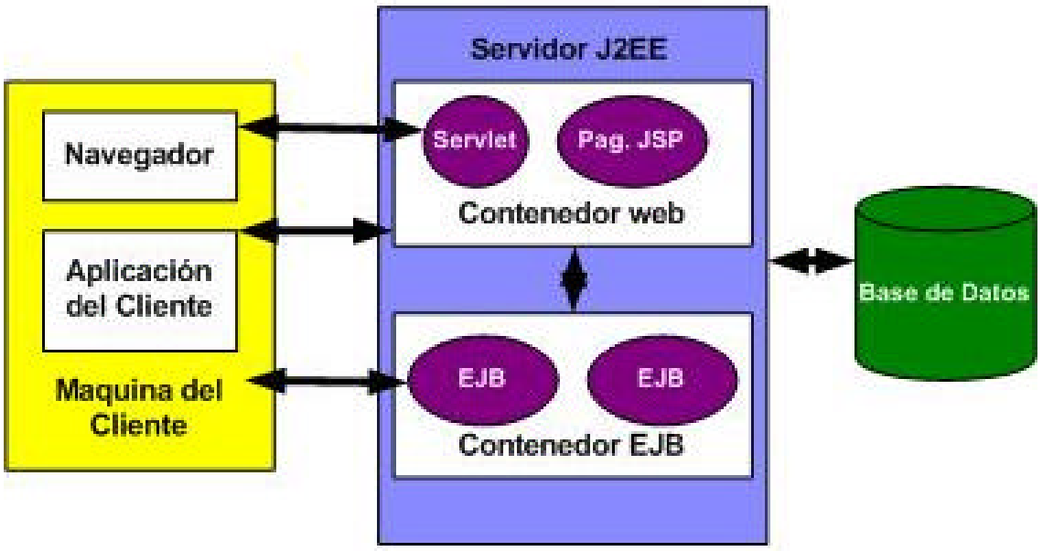


Figura Cap5_4, Relación entre el servidor J2EE y contenedores

5.3.5 Beneficios del estándar J2EE.

A continuación se enumeran algunos de los beneficios principales que proporciona el estándar J2EE.

1. Soluciones rápidas de liberar al mercado. Los desarrollos se concentran en la lógica de negocio en vez de estar preocupados de la infraestructura de la empresa.
2. Libertad de opción. La tecnología J2EE es un conjunto de estándares que muchos vendedores pueden implementar. Los vendedores son libres para manipular la implementación pero no los estándares o ***API***S. Sun proporciona una extensiva herramienta para evaluación de compatibilidad con el estándar J2EE, la llamada J2EE Compability Test Suite (CTS), para licenciarse en J2EE, esta herramienta ayuda a asegurar compatibilidad entre las aplicaciones que los vendedores ofrecen lo cual permite asegurar portabilidad para las aplicaciones y componentes escritos para J2EE.
3. J2EE proporciona soluciones Java entre todas las capas.

4. Conjunto de estándares abiertos— EJB, JSP, Servlets, JDBC, JNDI, entre otros.
5. Portabilidad, la característica Java WORA (Write Once, Run Anywhere, Escribe una vez, ejecuta en cualquier parte) está presente en el estándar debido al lenguaje de programación Java.

5.4 Microsoft .Net

Para satisfacer la demanda del mercado por soluciones integrales para las aplicaciones web bajo el esquema de servidores de aplicación es que Microsoft extiende su estándar ***DNA*** agrupando un conjunto de nuevas tecnologías basadas en su sistema operativo Windows bajo el nombre de .NET.

Microsoft.NET es un conjunto de tecnologías para conectar información, personas, sistemas y dispositivos. Proporciona un alto nivel de integración de software a través del uso de servicios web, que son pequeñas aplicaciones que se conectan las unas con las otras e incluso con aplicaciones más complejas a través de Internet

5.4.1 Microsoft .NET Framework.

.NET FrameWork: es un componente integral de Windows que permite la construcción y ejecución de servicios web más todo el entorno multicapa ya proporcionado anteriormente por ***DNA***. Incluye tecnologías para servicios web, aplicaciones web, acceso a bases de datos y otros.

Los elementos básicos de .NET son:

- **Software para cliente inteligente:** Software para ejecutar una variedad de clientes inteligente, desde computadores personales a pequeños dispositivos como PDAs, o a dispositivos más sofisticados

- **Servicios web *XML*:** Permite que las aplicaciones compartan datos y consuman servicios proporcionados por otras aplicaciones independiente de como fueron construidas dichas aplicaciones, en que plataforma corren, o que dispositivos las accedan, .Net incluye implementaciones de los últimos estándares para servicios web, tales como el estándar ***XML***.
- **Servidores empresariales:** Son aquellos productos que soportan aplicaciones empresariales resolviendo distintas partes de la solución integral.
- **Herramientas de desarrollo y ambiente de ejecución:** Incluye Visual Studio.Net, un ambiente de desarrollo integrado para la construcción de aplicaciones .Net, junto con.Net Framework, la cual tiene dentro el Lenguaje de Ejecución Común (Common Language Runtime, CLR), ***ASP***.Net (un nuevo ambiente para desarrollo web para la construir de aplicaciones .NET), y un completo conjunto de librerías.

El Framework de .NET es la infraestructura para la plataforma .NET, esta diseñado para simplificar el desarrollo de aplicaciones en ambientes altamente distribuidos, particularmente internet, su primera versión fue fue editada en enero de 2002 en su versión 1.0

Algunos beneficios de .NET Framework son:

- Soporta sobre 20 lenguages de programación diferentes, entre ellos: Pascal, C#, Perl, Java, Visual Basic. Debido esta capacidad multi-lenguajes, .NET Framework capacita a los desarrolladores en el uso de los lenguajes de programación más apropiados para una tarea determinada y permite combinar los lenguajes dentro de una misma aplicación. Los componentes escritos en distintos lenguajes pueden consumir funcionalidades los unos de

los otros en forma transparente, sin trabajo adicional por parte del desarrollador.

- Administra la mayoría de las complejidades involucradas en el desarrollo de software, permitiendo que los desarrolladores se enfoquen en codificar en la lógica de negocio.
- Simplifica la construcción, despliegue y administración de aplicaciones más robustas, seguras y eficientes.
- El diseño de .NET Framework basado en componentes, minimiza la cantidad del código que los desarrolladores deben escribir y maximiza el potencial para re utilización de código.
- Cuenta con dos componentes principales:
    - CLR, Lenguaje Común de Ejecución
    - Un conjunto de librerías de clases del Framework de .NET.

5.4.1.1 Lenguaje Común de Ejecución.

CLR es la base de Framework .NET, es una especie de agente que administra código en tiempo de ejecución, integración de lenguajes, proporciona servicios principales tales como administración de memoria, administración multi-hilos, administración remota, mientras fortifica la precisión y seguridad del código. El concepto de administración de código es un principio fundamental de CLR.

Además, el CLR también realiza algunas operaciones en tiempo de desarrollo cuando se trata de características tales como administración del cilco de vida, manejo de excepciones entre lenguajes, y reducción de la cantidad de código que un desarrollador debe escribir para llevar la lógica de negocio hacía un componente reutilizable.

5.4.1.2 Clases de librería Microsoft a .NET Framework

Las librerías de clases del Framework .NET son un conjunto de clases reutilizables orientadas al objeto y que ayudan al desarrollo de las aplicaciones

Las clases básicas proporcionan una funcionalidad estándar tales como manipulación de cadenas de texto, administración de seguridad, comunicaciones por red, administración de hilos de procesos, administración de textos, y características de disñeo para la interfaz del usuario.

5.4.1.2.1 Microsoft ASP .NET

ASP.NET es una estructura de programación construída en CLR que puede ser usado en un servidor para construir aplicaciones web robustas. ***ASP***.NET ofrece varios e importantes ventajas sobre los modelos previos de desarrollo:

- **Rendimiento:** ***ASP***.NET es código compilado en CLR en en el servidor. En forma distinta a sus precedesores, puede tomar ventaja de este tipo de compilación, optimización de código nativo, y el uso de mantener en memoria determinados servicios de uso frecuente. Esto radica en mejoras de ambiente
- **Poder y flexibilidad.** Debido a que ***ASP***.NET esta basado en CLR, el poder y flexibilidad de la plataforma entera esta disponible para desarrolladores de aplicaciones web. NET es a la vez independiente del lenguaje, así que se puede seleccionar el que mejor se acomoda a cada desarrollo.
- **Simplicidad.** Permite la creación de interfaces del usuario que separan en forma clara la lógica de negocio del código de presentación.
- **Flexibilidad.** emplea un sistema basado en archivos planos de texto, los cuales simplifican la configuración de los servidores y de las aplicaciones web. Debido a que la información de configuración es almacenada como

texto plano, la instalación de una aplicación ***ASP***.NET radica en en la copia de los archivos involucrados. No es necesario reiniciar los servidores, ni siquiera para reemplazar código que este siendo ejecutado.

- **Escalabilidad y Disponibilidad.** Los procesos son monitoreados por ***ASP***.NET, así que al momento de que alguno de ellos falle, un nuevo proceso puede ser creado en su lugar lo que permite que las aplicaciones esten constantemente disponibles para manejar las solicitudes.

5.4.1.2.2 Microsoft ADO .NET

La clase ***ADO***.NET permite que los desarrolladores interactuen con datos accesados en la forma de XML a traves de OLE DB, ODBC, Oracle, and ***SQL*** server. Las clases ***XML*** permiten manipulación de ***XML***, busqueda y transformaciones.

ADO.NET es capaz de utilizar ***XML*** para proporcionar aceso desconectado a los datos.

5.5 Servidores de Aplicación basados en J2EE.

Existen numerosos servidores de aplicación basados en el estándar J2EE, pero solo algunos de ellos son certificados en J2EE, esto significa que cumplen con el estándar cabalmente, lo que permite que las aplicaciones realizadas con ellos tengan un conjunto de características compatibles entre sí.

Los conceptos que son comunes a todo servidor de aplicación certificado J2EE son:

- Su ubicación dentro del modelo multicapa, como puede ser visto en *figura Cap5_3.*

- Los componentes EJB, JSP y servlets.
- El conjunto de APIs provistas por J2EE.

Dentro de las diferentes alternativas que ofrece el mercado informático para servidores de aplicación que cumplan totalmente con el estándar de J2EE se cuentan principalmente, por distribución de mercado, los siguientes:

- ***BEA*** WebLogic Server (WLS)
- ***SUN*** iPlanet Aplication Server (iAS).
- ***IBM*** WebSphere Application Server (WAS).

5.5.1 BEA WebLogic Server (WLS).

Proporciona un conjunto completo de servicios basados en J2EE y maneja automáticamente muchos detalles del comportamiento de las aplicaciones, sin requerir programación extra.

Presenta las siguientes características:

- Soporte comprensivo de J2EE, lo que facilita la implementación y despliegue de las aplicaciones. WLS es el primer servidor de aplicaciones independiente en conseguir la certificación J2EE.
- Soporta a los siguientes clientes: navegadores web y otros clientes que usen **HTTP** junto con dispositivos móviles que usan ***WAP***.
- Los recursos críticos se usan eficientemente y la alta disponibilidad está asegurada a través del uso de los componentes J2EE y mecanismos como el ***clustering*** de WLS para las páginas web dinámicas.
- Ofrece una consola de administración basada en ***web*** para configurar y monitorizar los servicios de WLS.
- Proporciona soporte de ***SSL*** para encriptar datos transmitidos a través de WLS, los clientes y los otros servidores. La seguridad de

WLS permite la autenticación y autorización del usuario para todos sus servicios.

- Proporciona una estrecha integración y soporte con las bases de datos más importantes, herramientas de desarrollo y otros entornos.

Cuenta con la opción ***cluster*** que permite que se distribuyan peticiones de cliente y servicios de negocio entre varios servidores WLS. Los programas en la capa del cliente acceden al ***cluster*** como si fuera un solo servidor WLS. Cuando la carga de trabajo aumenta, se pueden agregar otros servidores al ***cluster*** para compartir el trabajo. El cluster utiliza un algoritmo de balance de carga para elegir el servidor WLS que sea capaz de manejar la petición, cuando una petición falla, otro servidor que proporcione el servicio solicitado puede asumir el control. Por ejemplo, el estado de la sesión de un servlet se puede replicar en un servidor secundario WebLogic de modo que si el servidor WebLogic que está manejando una petición falla, la sesión del cliente se pueda reanudar de forma ininterrumpida desde el servidor secundario. Todos los servicios de WebLogic, EJB, JMS, JDBC, y RMI están implementados con capacidades de ***clustering***.

WLS cuenta con un almacén de conexiones JDBC, definido en el servidor WebLogic, abre un número predefinido de conexiones a la base de datos. Una vez que estén abiertas, las conexiones a la base de datos son compartidas por todas las aplicaciones del servidor WebLogic que necesiten acceder a esa base de datos. Sólo se incurre una sola vez en la costosa sobrecarga asociada con el establecimiento de conexiones para cada conexión del almacén, por cada petición de cliente. El servidor WebLogic vigila las conexiones a la base de datos, refrescándolas cuando es necesario y asegurándose de la fiabilidad de los servicios de la base de datos para las aplicaciones.

El servidor WebLogic escucha peticiones de conexión en una dirección de red que se pueda especificar como parte de un identificador de recursos uniforme (URI). Un URI es una cadena estandarizada que especifica un recurso en una red, incluyendo internet. Contiene un esquema, la dirección de red del servidor, el nombre del recurso deseado, y los parámetros opcionales. La URL que se introduce en un navegador web por ejemplo, es el formato más familiar de URI.

5.5.2 SUN iPlanet Aplication Server (iAS).

Junto con ser compatible con el estándar J2EE, iAS tiene las siguientes funcionalidades :

- El control de fallos para los componentes EJB
- Balance de carga.
- Balance de carga JSP: se identifica a cada proceso JSP individualmente. Esto permite aplicarles balance de carga y de forma similar a los Servlets.
- El ***cache*** de páginas JSP, las llamadas a páginas JSP son almacenadas de tal forma que de ser usadas nuevamente se recurra a lo almacenado en cache

Las principales características de iPlanet son:

- Elimina los puntos de falla a través del control de fallos en cada componente J2EE
- Asegura que la información del usuario y los datos de la aplicación no se perderán durante la ocurrencia de un fallo distribuyendo la información de estado y de sesión de una transacción en varios servidores.

- Mejora las capacidades de balance de carga: los administradores pueden asignar parámetros como números y velocidades de CPUs de CPU para determinar cómo se pueden balancear las peticiones el servidor.
- Soporta el balance de carga entre CPUs dentro de un servidor multiprocesador, y proporciona capacidades de balance de carga a través de procesos J2EE. También proporciona balance de carga basado en el tiempo de respuesta y en la carga del servidor. El balance de carga también está disponible para procesos JSP.
- Proporciona características de alto rendimiento incluyendo cache y almacén de conexiones, una arquitectura multi-hilos y multiprocesador.
- Integra el monitor de transacciones ***Encina*** como característica principal del rendimiento óptimo del servidor, eficacia y administración.
- Cuenta con un servidor web asociado, el iPlanet Web Server, que proporciona mejoras de rendimiento sobre otras plataformas de servidores web.

La arquitectura de iAS tiene tres tipos internos de servidores, que normalmente son llamados motores o procesos. Son los responsables de todo el proceso dentro de iAS**.** La siguiente tabla resume los servidores internos:

Servidor Interno	Nombre del Proceso	Descripción
Executive Server	KXS	Proporciona la mayoría de los servicios como el balance de carga y el control de fallos.
Administrative Server	KAS	Proporciona servicios del sistema para administración y control de fallos de **iAS**.
Java Server	KJS	Proporciona servicios para aplicaciones Java

Tabla Cap5_A, Servidores de iAS.

5.5.3 IBM WebSphere Application Server (WAS).

Soporta una amplia gama de estándares abiertos. Entre las más importantes mejoras que WAS tiene, se destacan:

- Los servicios de Internet: ***SOAP***, ***UDDI***, ***WSDL***, ***XML***.
- Certificación de J2EE, incluyendo integración robusta y tecnología de manejo de transacciones.
- Administración sencilla y amigable basada en tecnología XML.
- Una mejora en los mecanismos de administración del servidor de aplicaciones.
- Una versión mejorada de su servidor web basado en Apache.
- Cuenta con un cliente de EJB y J2EE para facilitar la invocación directa de EJB y componentes de J2EE desde un cliente que puede ser instalado separadamente.

Fuera de todas estas características, uno de los factores que hace de WebSphere una plataforma tan valiosa para empresas en el mundo entero, es

la amplia gama de plataformas que soporta, lo que permite a las empresas sacar mayor provecho de la infraestructura que ya posee, y al mismo tiempo tener la certeza de que la inversión que se está llevando a cabo podrá crecer a plataformas más robustas según los requerimientos de la empresa. Esto es particularmente importante en ambientes de producción, donde las facilidades de agrupación de servidores o ***clustering*** y balanceo de cargas son requerimientos siempre presentes.

WebSphere está fundamentado en un código único a través de todas las plataformas soportadas, lo que asegura una total portabilidad de las aplicaciones de una a otra, simplificando mucho el crecimiento y migración hacia plataformas más robustas.

CAPITULO 6

EVALUACION Y SELECCION DE PLATAFORMA

6. Evaluación y selección de plataforma.

Los servidores de aplicación disponibles en el mercado se subdividen en dos conjuntos actualmente:

- Los construidos por un tercero basándose en el estándar J2EE.
- Y los construidos con Microsoft .NET.

Debido a que el primero corresponde a una aplicación de un estándar y el segundo a un producto, no es posible realizar una comparación directa entre J2EE y .NET, a menos que dichas comparaciones se hagan desde dos prismas distintos pero complementarios: comparar las características teóricas de ambos y comparar servidores de aplicación J2EE representativos (como los ya descritos en el capítulo cinco) contra .NET.

6.1 Microsoft .Net v/s J2EE.

Tan importante como la propia arquitectura que plantean estas dos plataformas de desarrollo, son las empresas que las apoyan ya que contribuirán de forma decisiva en su éxito o fracaso.

J2EE cuenta con el apoyo de grandes empresas de software que han realizado su propia implementación del estándar: ***IBM***, ***BEA***, Oracle o la misma ***SUN***, ofrecen sus plataformas de desarrollo de aplicaciones, en las que junto a J2EE ofrecen otros productos como bases datos, ***cache***, ***firewalls***, etc, para dar una solución integral a sus clientes.

Por su parte, Microsoft, y su gran equipo de marketing, están decididos a conseguir que .NET sea la plataforma de desarrollo de aplicaciones eBusiness preferida. Algunas empresas muy importantes ya han comenzado a desarrollar soluciones usando .NET como Accenture, Arthur Andersen, Compaq o Deutsche Bank. En el futuro las dos plataformas tendrán que convivir forzosamente, compitiendo por alcanzar una mayor cuota de mercado. J2EE

parte con la ventaja de haber comenzado antes y de ser apoyado por empresas de gran renombre que cuentan con importantes clientes utilizarán el estandar. Por otra parte Microsoft es la mayor empresa de software a nivel mundial que gracias a sus precios ajustados y a su excelente departamento de marketing puede llegar a una importante masa de empresas y desarrolladores.

6.1.1 Analogías y diferencias entre J2EE y .NET.

Estas dos plataformas coinciden en tener funcionalidades comunes, que descansan en conceptos teóricos que pueden ser analizados en conjunto para resaltar similitudes y diferencias, a continuación se analizan los más significativos.

6.1.1.1 Sistema de Tiempo de ejecución.

En .Net :

- es llamado **CLR**.
- esta pensado para cualquier lenguaje compilado en ***MSIL***
- proporciona integración para varios lenguajes
- proporciona soporte para lenguajes no orientado a objetos

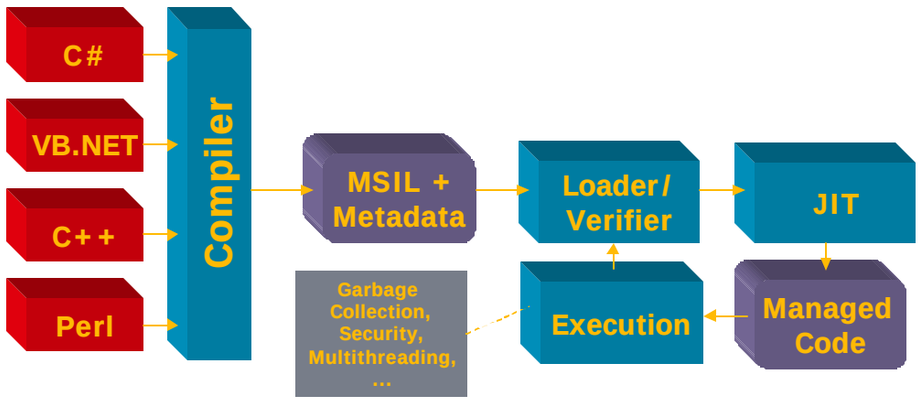


Figura Cap6_1, Sistema de Tiempo de ejecución para .NET

En J2EE:

- es llamado **JVM (Java Virtual Machine)**
- JVM esta pensado para Java y para interpretar ***JBC*** (Java Byte Code).
- Otros lenguajes pueden ser compilados al ***JBC***.
- El compilador en tiempo real existe para varios ambientes y sistemas operativos

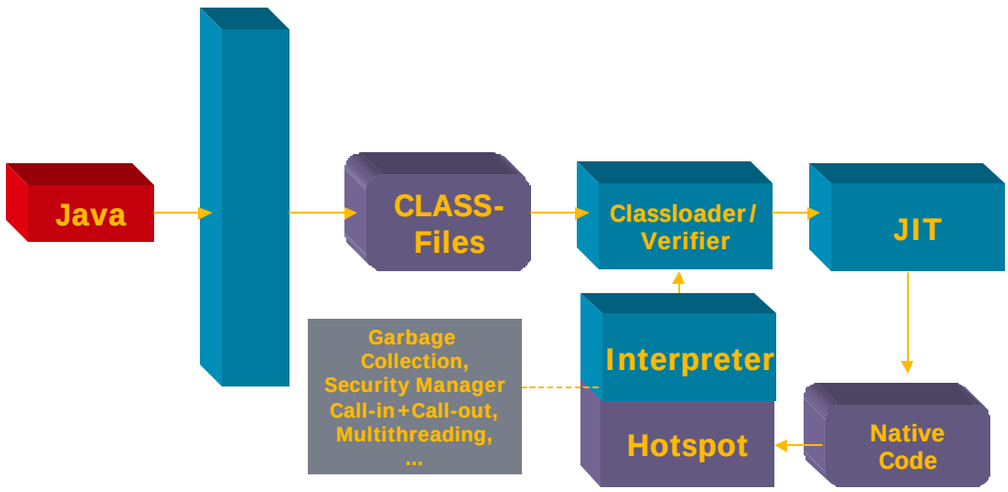


Figura Cap6_2, El sistema de tiempo de ejecución para J2EE

6.1.1.2 Acceso a bases de datos.

En .Net se utiliza ***ADO***.NET para acceder a las bases de datos, cuyas características son:

- ***ADO***.NET esta basado en ***XML*** , en donde
  - El conjunto de datos (DataSet) dinámicamente construye un esquema ***XML*** dentro de un almacen de datos.
  - Los datos relacionales y los datos en ***XML*** pueden ser manejados en forma similar.
- ***ADO***.NET trabaja fuera de línea, desconectado, una vez que se traigan los datos.

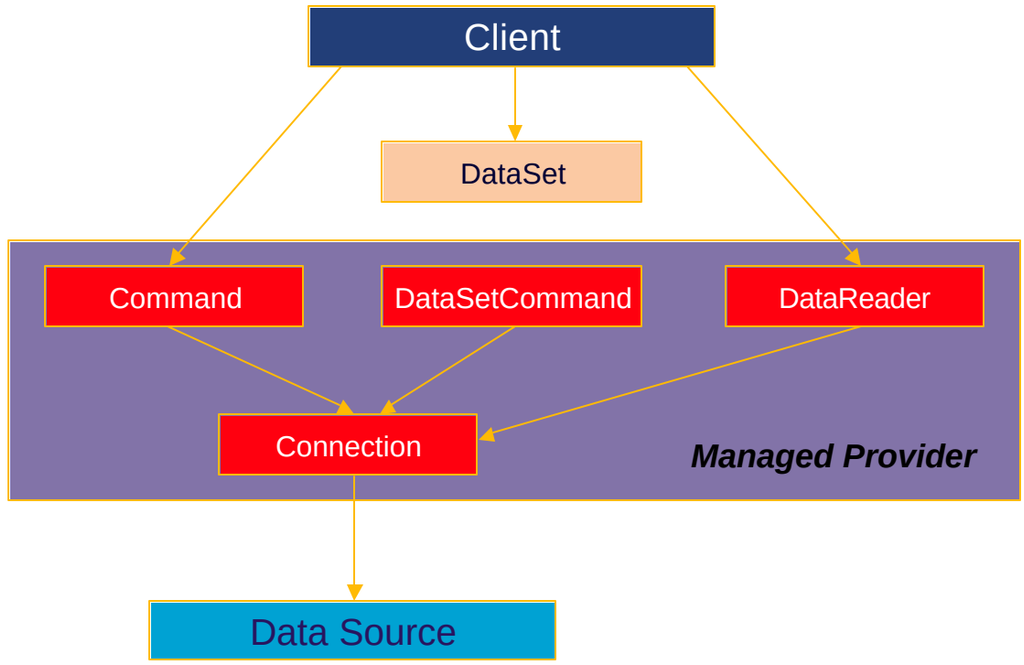


Figura Cap6_3, Acceso a base de datos en .NET

Java Proporciona ***JDBC*** para acceder a bases de datos relacionales.

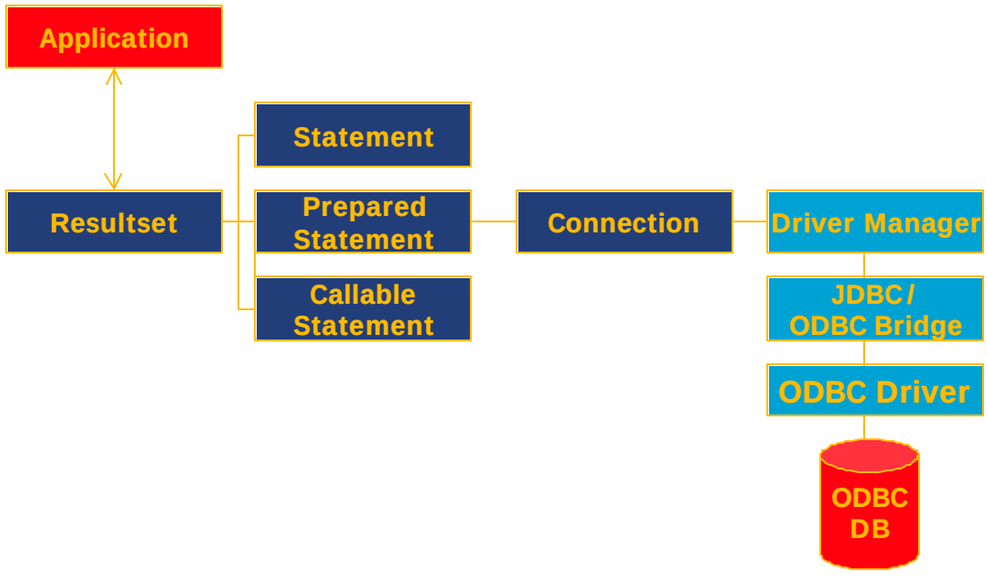


Figura Cap6_4, Acceso a bases de datos en Java

6.1.1.3 Páginas dinámicas

Para la construción de páginas dinámicas en .Net se utiliza ***ASP***.NET, cuya arquitectura se detalla en la siguiente figura:

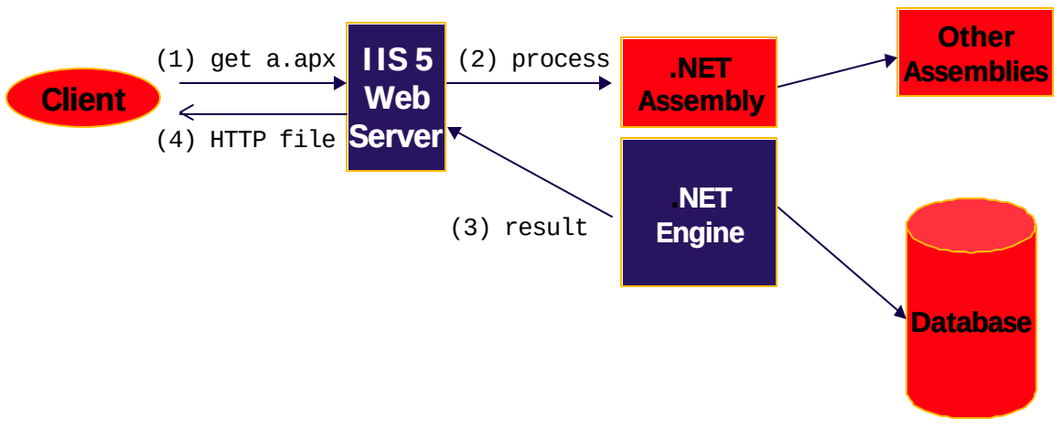


Figura Cap6_5, Arquitectura de ASP.NET

En J2EE para la construcción de páginas dinámicas se utilizan JSP, los JSP estan basados en servlets. La arquitectura es la mostrada en la siguiente figura:

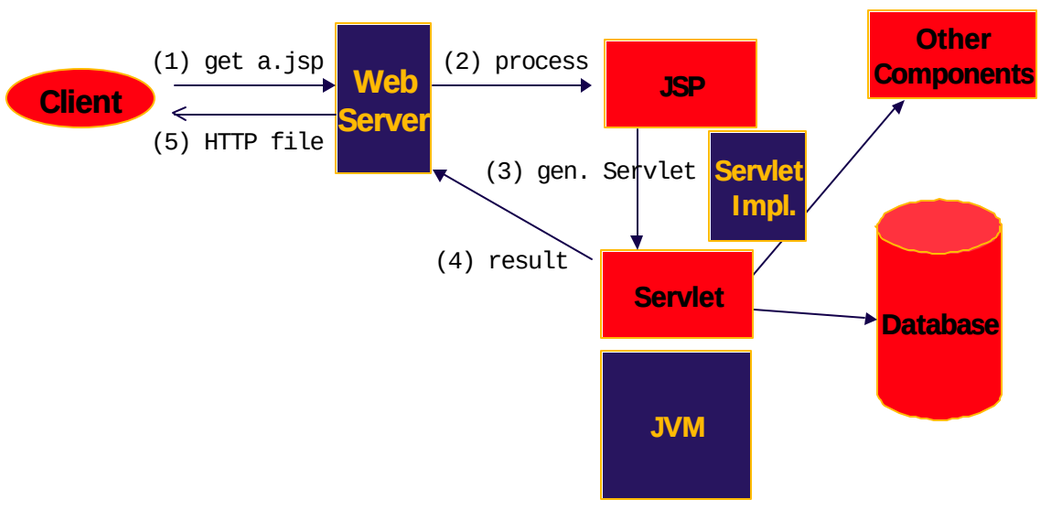


*Figura Cap6_6, Arquitectura de **JSP***

6.1.1.4 Lenguajes de programación

El único lenguaje que soporta J2EE es Java y es el que se utiliza para el desarrollo de todos los componentes. Existen sólo dos formas oficiales para acceder a la plataforma J2EE con otros lenguajes, la primera es a través de ***JNI*** y la segunda es a través de la interoperabilidad que ofrece CORBA.

Por otra parte, Microsoft .NET ofrece soporte oficial para Visual Basic.NET, C++.NET y un nuevo lenguaje C# que es equivalente, con la excepción de portabilidad, a Java.

Microsoft .NET va más allá de soportar estos lenguajes sino que también ofrece plena interoperabilidad entre ellos, por lo que es posible construir un componente en un lenguaje, introducirlo en una aplicación escrita en otro distinto e incluso heredarlo y añadir nuevas características en un tercero.

Respecto a esta capacidad de Microsoft .NET de trabajar con varios lenguajes existen dos posturas enfrentadas:

- La que esta a favor de esta capacidad, que se fundamenta en que permite una migración más sencilla para antiguos programadores, reduciendo el tiempo de formación. Además, trabajar con un lenguaje conocido proporciona gran productividad individual.
- Por otro lado, la postura contraria, explica que la existencia de varios lenguajes de programación en una única empresa acarrea efectos negativos, tales como:
    - La sencillez de mantenimiento se reduce. Si una aplicación esta realizada en varios lenguajes se necesitan expertos en varios lenguajes para su comprensión y mantención, aumentando los costos considerablemente.

- La productividad del grupo de desarrollo decrece. Si los programadores utilizan lenguajes diferentes no pueden comunicar fácilmente sus conocimientos de unos a otros.
- Transferencia de conocimientos. En el caso de que un desarrollador o grupo de desarrolladores especializados en un lenguaje abandonan un proyecto, es necesario el reemplazo con otros que conozcan el mismo lenguaje para continuar con el desarrollo, por lo tanto se hace necesario entrenar a los existentes para que actualicen sus conocimientos.

Se puede considerar que una empresa dedicada al desarrollo de software debe seguir un criterio homogéneo y realizar todos sus desarrollos utilizando un único lenguaje, ya sea Java, C# o cualquier otro, independientemente de la plataforma utilizada.

6.1.1.5 Herramientas de desarrollo

Microsoft siempre se ha caracterizado por ofrecer a los desarrolladores algunos de los mejores entornos de desarrollo del mercado. Es el caso de Visual Studio.NET ya que en un único interfaz de desarrollo se tiene un editor de código multilenguaje, un compilador, editor de recursos, conexión a base de datos, editor XML, depurador, ayuda en línea, etc.

Varias empresas ofrecen entornos de desarrollo para las aplicaciones de J2EE, entre ellas : Forte de Sun, Visual Café de WebGain, Visual Age for Java de ***IBM*** y JBuilder de Borland. Aunque muchas de ellas ofrecen excelentes productos, ninguno llega al nivel de integración y la facilidad de uso de Visual Studio.NET. Además, muchos de estos entornos de desarrollo están escritos en Java lo que significa que requieren muchos más recursos que la aplicación

equivalente escrita específicamente para un sistema operativo concreto, lo que a su vez, hace que el costo de los equipos necesarios para el desarrollo crezca.

6.1.1.6 Tabla teórica comparativa

En la siguiente tabla se puede ver un resumen de las similitudes y diferencias tanto tecnológicas como de servicios entre Microsoft .NET y J2EE.

Característica	.NET	J2EE
Tipo de tecnología	Producto	Estándar
Empresas que lo ofrecen	Microsoft	Más de 30
Librerías de desarrollo	.NET Framework SDK	Java core ***API***
Intérprete	CLR	JRE
Páginas dinámicas	***ASP***.NET	Servlets, ***JSP***
Componentes	Componentes .NET	***EJB***
Acceso a bases de datos	***ADO***.NET	***JDBC***, SQL/J
Servicios Web	***SOAP***, WDSL, UDDI	***SOAP***, WDSL, UDDI
Herramienta de programación	Visual Studio .NET	Java
Transacciones distribuidas	MS-DTC	***JTS***
Librería de encolado de mensajes	MSMQ	***JMS*** 1.0
Lenguajes utilizados	C#, Visual Basic, C++, otros	Java
Lenguaje intermedio	IL	Bytecodes

Tabla Cap6_A, comparativa teórica de características J2EE y .NET

Los parecidos entre las dos plataformas en los aspectos principales son evidentes:

Un lenguaje de programación que unido a una librería de clases, compila a un código intermedio independiente de la máquina :

- Intermediate Language en el caso de Microsoft .NET
- Bytecodes en el caso de Java.

Este código se ejecuta en máquina virtual, que proporciona un entorno de ejecución que transforma el lenguaje intermedio a código propio de la máquina en la que se corre la aplicación :

- Common Language Runtime (CLR) en Microsoft .NET
- Java Runtime Environment (JRE) en J2EE.

Aparte las dos plataformas disponen de multitud de servicios que facilitan la labor del programador, entre ellos, el acceso a bases de datos, los servicios de directorio, las transacciones distribuidas.

6.2 Malla de evaluación.

Para determinar cual de las dos plataformas propuestas es la que mejor se ajusta a las necesidades de ***BECH***, se definieron parámetros representativos de los factores de calidad esperados por dicha institución. Una vez definidos, se procedió a darles un valor numérico que representara la importancia del cumplimiento de dicho parámetro.

Posteriormente se tomaron todos estos parámetros con sus respectivos valores para dar forma a una matriz o malla de evaluación, de cuya lectura se interpreta cuál de las alternativas cumple mejor con los factores de calidad.

Es así que dentro de la malla de evaluación se define:

- El valor de ponderación como el encargado de medir la importancia que tiene cada uno de los parámetros dentro de la selección de la mejor plataforma, representado por un porcentaje.
- Una nota entre el rango de 1 y 3, calculada de la siguiente manera, para cada parámetro:
  - Se determinó que plataforma es la que mejor rendimiento arrojó, dándole a ella el valor más alto, es decir un tres.
  - Se determinó que plataforma es la que menor rendimiento mostraba, dándole a ella el valor más bajo, es decir un uno.
  - Se ajustaron las notas de las restantes plataformas por medio de una escala.

6.2.1 Definición de los parámetros a cuantificar.

A continuación se listan y describen los parámetros a considerar dentro de la malla de evaluación, junto con su respectivo porcentaje de importancia dentro de la consideración total, el cual aparece entre paréntesis.

- **Páginas por segundo (sin descarga de imágenes):** corresponde al número de páginas por segundo que el servidor puede resolver satisfactoriamente. Se considera sin descarga de imágenes para probar efectivamente al servidor de aplicaciones en forma independiente al servidor de presentación. (5%)
- **Máximo número de usuarios en un mismo instante (sin descarga de imágenes):** corresponde al número máximo de usuarios conectados en un mismo instante. Se considera sin descarga de imágenes para probar efectivamente al servidor de aplicaciones en forma independiente al servidor de presentación. (5%)

- **Páginas por segundo (con descarga de imágenes):** corresponde al número de páginas por segundo que el servidor puede resolver satisfactoriamente. Se considera con descarga de imágenes para probar al servidor de aplicaciones en conjunto con el servidor de presentación. (10%)
- **Máximo número de usuarios en un mismo instante (con descarga de imágenes):** corresponde al número máximo de usuarios conectados soportados en un mismo instante. Se considera con descarga de imágenes para probar al servidor de aplicaciones en conjunto con el servidor de presentación. (10%)
- **Máximo número transacciones en línea por segundo durante un tiempo dado:** corresponde al número máximo de transacciones, es decir la cantidad de accesos al servidor de aplicaciones en las cuales se gatillo alguna escritura a la capa de datos durante un tiempo determinado, para certificar la estabilidad del servidor de aplicaciones. (10%)
- **Líneas de código:** Que el tamaño de las páginas dinámicas y componentes involucrados (código fuente) sea el mínimo posible. (5%)
- **Seguridad:** Que existan niveles de seguridad eficientes que den protección y confianza al momento de realizar transacciones por medio del web. (20%).
- **Soporte:** Que exista un respaldo técnico de parte de algún proveedor de ésta tecnología dentro de Chile. (10%).
- **Escalabilidad:** Que la tecnología sea expandible en el tiempo, ya sea por la incorporación de más hardware o de nuevas versiones del software (10%).

- **Administración:** Facilidades de interfaz y operabilidad al momento de la administración de los servicios internet. (5%).
- **Precio/Desempeño:** Esta relación solo sirve como antecedente debido a que los aspectos económicos de la malla de evaluación no pesaran mayormente en la decisión del área informática de ***BECH***, sino que recaerán en gerencias de recuros económicos, es decir a partir de un análisis informático con respecto a la mejor solución disponible en el mercado se agregará el factor ecónomico, lo que a momentos de economizar puede terminar en la selección de una plataforma en vez de otra tan solo por factores economicos bajando los niveles de cumplimiento de los factores de calidad en pos de una ganancia económica.

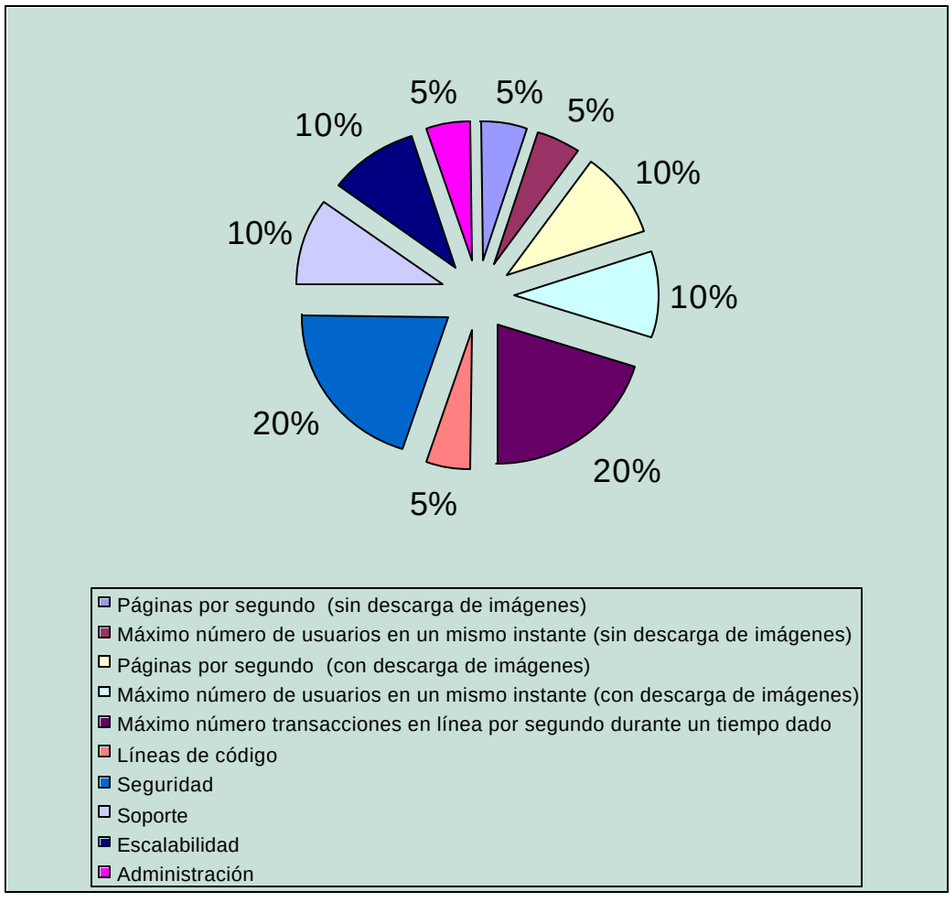


Figura Cap6_7, Parámetros cuantificables en malla de evaluación.

6.3 Descripción de fuentes investigadas.

Para poder obtener valores que indiquen el comportamiento de .NET y J2EE para cada uno de los parámetros a cuantificar se debe recurrir a información publicada por los propios fabricantes, pero para mantener la objetividad en la comparación, se debe contar además con informes realizados por terceros, independientes de las presiones comerciales que se pudieran ejercer con el fin de obtener mejores evaluaciones por parte de los propios fabricantes.

En este capítulo se recurrirá a estos dos tipos de fuentes, para el caso de un tercero se hará referencia a una serie de ***benchmark*** ejecutados por MiddleWare Company sobre servidores de aplicación J2EE y .NET.

6.3.1 MiddleWare Company.

Middleware Company es una compañía que se especializa en entrenamiento y consultorías basadas en tecnologías avanzadas para Java empresarial. Fue fundada en 1998 para asistir la migración de empresas tales como ***BEA***, Oracle, Cisco, Nextel, MetLife y otras a la plataforma Java.

Esta compañía realizó una serie de benchmarks entre Junio y Septiembre de 2002 a servidores de aplicación J2EE y .NET, pruebas que serán detalladas y analizadas a continuación.

6.3.2 Java PetStore y .NET Pet Shop.

En mayo de 2001 Sun Microsystems introdujo la Java Pet Store (tienda de mascotas Java, en adelante Java PS) como una demostración de implementación para aplicaciones web basadas en J2EE. En noviembre de ese mismo año Microsoft anunció que había re-implementado la Java Pet Store usando Microsoft .NET frameWork y C# para ilustrar las ventajas de la

plataforma .NET sobre J2EE. La .NET Pet Shop (Tienda de Mascotas .NET), funcionalmente es idéntica a la Sun Java Pet Store, pero construida totalmente bajo .NET

Microsoft editó información de varios ***benchmark*** ejecutados sobre .NET Pet Shop mostrando que el rendimiento de este producto era significativamente mejor bajo altas cargas de usuarios que su equivalente Java. Las comparaciones de ***benchmark*** están basadas en comparar el rendimiento de .NET Pet Shop con los ***benchmark*** publicados por Oracle con respecto a Java PS. Muchos desarrolladores Java, junto con ***SUN***, IBM y Oracle han mantenido que las comparaciones hasta la fecha no eran validas, debido a que la aplicación de Java PS no fue propiamente optimizada para rendimiento y menos para que se corrieran benchmark sobre ella.

Debido a la existencia de esta aplicación en las dos plataformas en forma idéntica en cuanto a la funcionalidad final, resulta atractivo realizar ***benchmarks*** a versiones homologadas de la tienda de mascotas para asegurar resultados objetivos.

6.3.3 Preparando el ambiente antes de ejecutar el benchmark.

Para realizar las pruebas de la forma más objetiva y transparente se definieron las siguientes reglas básicas:

- Que la implementación J2EE y .NET fueran funcionalmente 100% equivalentes sin diferencias de comportamiento
- Ambas aplicaciones fueron creadas con estándares de código óptimos, siguiendo las así llamadas mejores - prácticas (best-practice) de tal forma que cada una sirva como un patrón de diseño válido que clientes reales puedan seguir cuando construyan sus propias aplicaciones.

- Cada aplicación debe tener una implementación en tres capas, con el uso de componentes para encapsular la capa de negocios y la lógica de acceso a los datos.
- Las aplicaciones deben ser diseñadas de tal forma que puedan ser instaladas en cluster a través de múltiples servidores para permitir escalabilidad.
- Los benchmarks a utilizar deben ser ejecutados en aplicaciones y configuraciones de bases de datos realistas que reflejen un ambiente de producción.

Todo el código fuente, datos cargado y scripts de prueba para realizar el benchmark sobre las aplicaciones (.NET y J2EE) están publicadas en www.TheServerside.com para verificaciones.

Middleware Company probó con dos servidores de aplicación basados en J2EE, para evitar problemas legales al momento de publicar los resultados de su ***benchmark***, estos dos servidores de aplicación fueron denominados el servidor de aplicación A y el servidor de aplicación B, sin embargo se incluyó código específico de WebLogic en la versión descargable de J2EE PetStore, lo que junto con otros reportes en las mismas versiones descargables del ***benchmark*** llevan a deducir con seguridad que WebSphere es el otro servidor de aplicaciones utilizado en el benchmark.

A partir de esto en las siguientes tablas y gráficos a utilizar se especifica las siguiente asociación:

- Servidor de aplicaciones A = Servidor de aplicaciones WebLogic.
- Servidor de aplicaciones B = Servidor de aplicaciones WebSphere.

Por lo mismo los servidores de aplicación probados fueron:

- .NET 1.0 en Windows 2000.

- .NET 1.1 sobre un servidor Windows .NET
- Servidor de aplicaciones WebLogic
- Servidor de aplicaciones WebSphere

6.3.4 El benchmark.

El ***benchmark*** esta compuesto por tres conjuntos de pruebas, las cuáles proporcionan un resumen completo y justo de las características de escalabilidad y rendimiento de .NET y J2EE para las dos aplicaciones. Estos conjuntos de pruebas corresponden a:

- Benchmark de aplicaciones web.
- Benchmark de transacciones distribuidas durante 24 horas.
- Benchmark de servicios web.

6.3.4.1 Benchmark de aplicaciones web.

Este ***benchmark*** ejercita la mayoría de las funcionalidades básicas de la aplicación. Las pruebas fueron ejecutadas para determinar una curva de respuesta así como una medida de tiempos de respuesta para la aplicación desde cargas de usuarios bajas hasta cargas altas, empujando a cada producto a altos niveles de stress para determinar el máximo número de usuarios permitidos en configuraciones de servidores de aplicación provistas de 2, 4 y 8 ***CPU***. Los scripts de prueba fueron ejecutados con un tiempo de espera de respuesta de 10 segundos. Las opciones de usar cache en los servidores fue deshabilitada para forzar a que los serviores resolvieran cada solicitud recibida.

Para proporcionar una información más detallada, este conjunto de ***benchmark*** fue ejecutado en dos formas:

- **Sin descarga de imágenes**, de tal forma que el servidor de aplicaciones básico es utilizado. Esta prueba indica que el producto también maneja la lógica de la aplicación, incluyendo scripting en el lado del servidor, administración del estado de las sesiones, activación de objetos y conectividad a bases de datos
- **Con descarga de imágenes,** simulando la memoria cache de un navegador, de tal forma que cada usuario que ingrese al sitio descargue una imagen única cualquiera tan solo una vez durante una iteración del script. Esta prueba muestra que tan bien el servidor de aplicación actúa en forma combinada como un servidor web y servidor de aplicaciones.

Los resultados de este ***benchmark*** son :

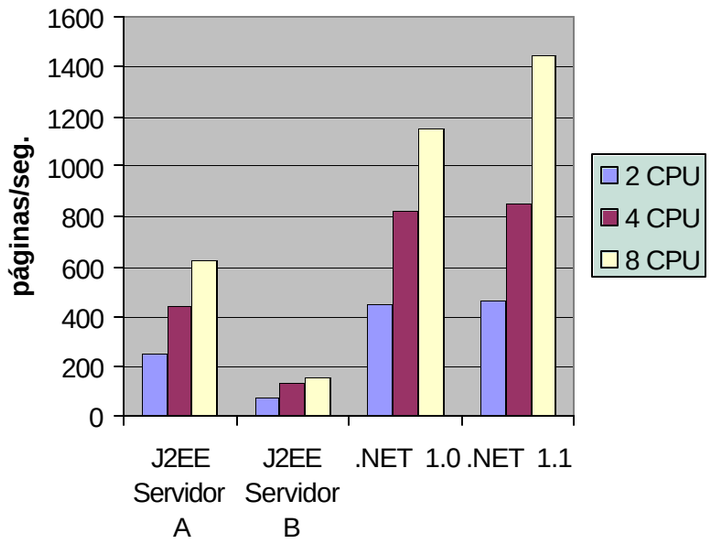


Figura Cap6_8, Mejores respuestas sin descarga de imágenes

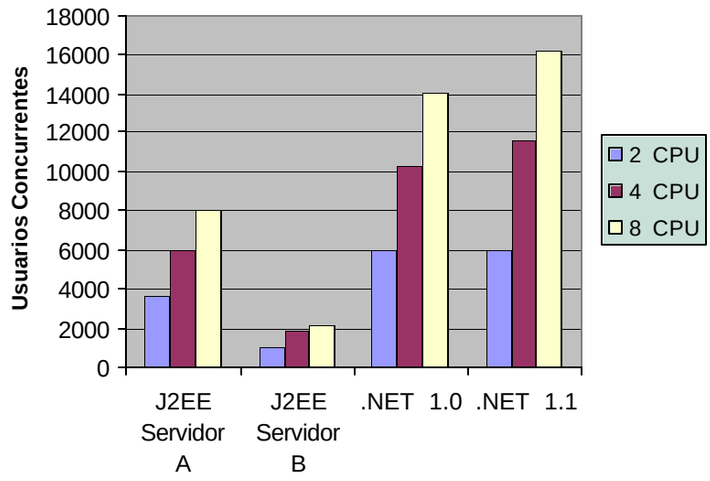


Figura Cap6_9, Carga máxima de usuarios concurrentes sin descarga de imágenes

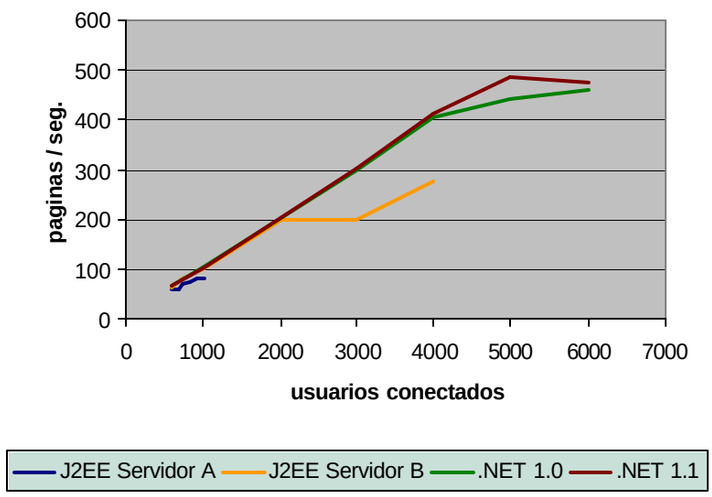


Figura Cap6_10, Curvas de Salida para servidor de aplicaciones con 2 CPU

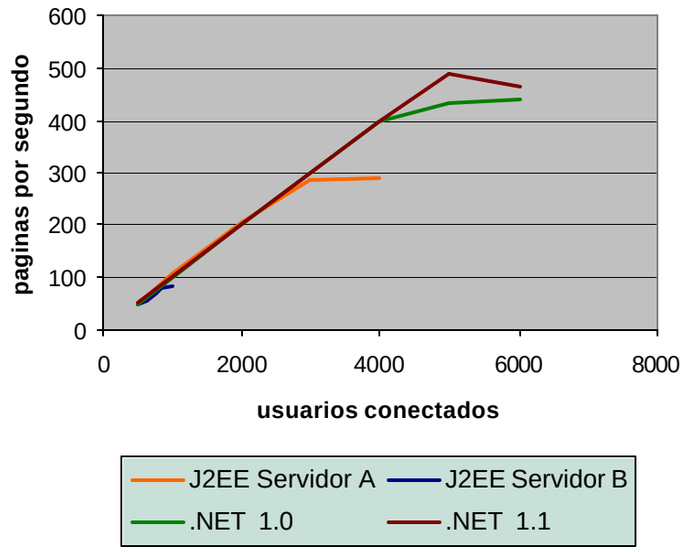


Figura Cap6_11, Curvas de salida para páginas por segundo con servidor de aplicaciones con 2 CPU

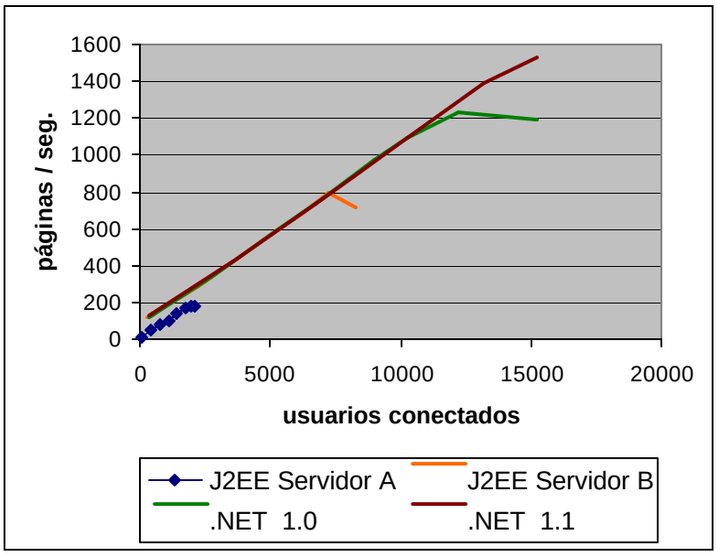


Figura Cap6_12, Curvas de salida para páginas por segundo con servidor de aplicaciones con 8 CPU

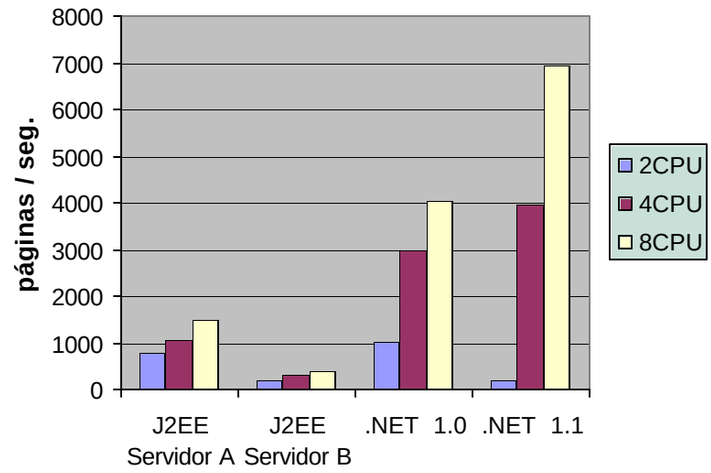


Figura Cap6_13, Mejores respuestas con descarga de imágenes

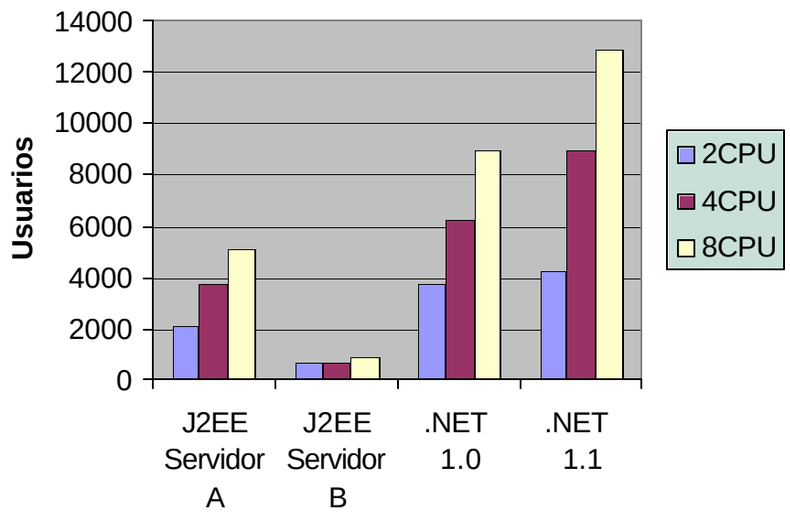


Figura Cap6_14, Carga máxima de usuarios concurrentes con descarga de imágenes

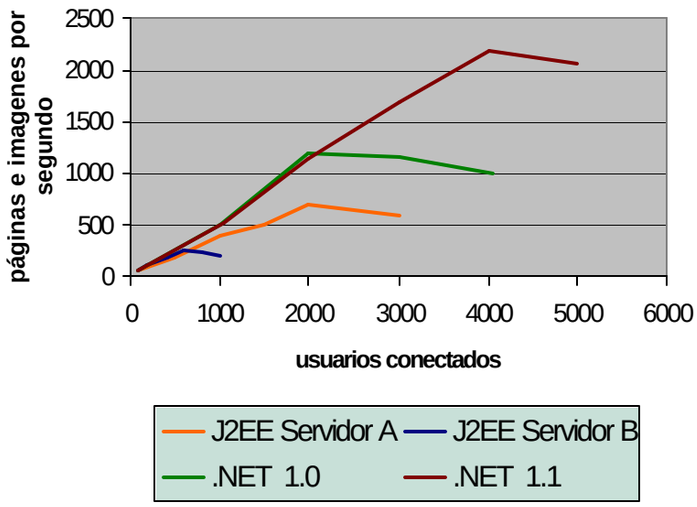


Figura Cap6_15, Curvas de Salida para páginas por segundo con Servidor de aplicaciones con 2 CPU

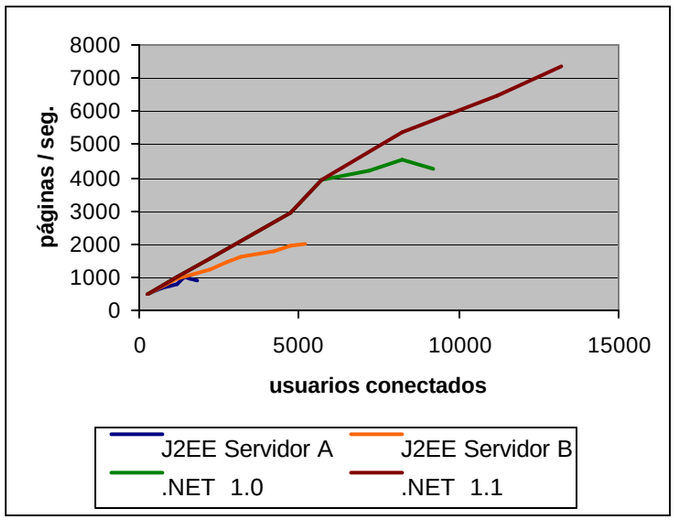


Figura cap6_16, Curvas de salida para páginas por segundo con servidor de aplicaciones con 8 CPU

6.3.4.2 Benchmark de transacciones distribuidas durante 24 horas.

Este ***benchmark*** fue diseñado para estresar la capacidad de procesar transacciones distribuidas de cada producto probado. El ***script*** de prueba consistía de usuarios ingresando al sitio y procediendo a ordenar individualmente 100 items. Por cada item ordenado, el proceso de compra fué completado, con el ultimo paso del proceso siendo realmente el que active una transacción distribuida, seguida por una desconexión al final del ***script***. Cada usuario por lo tanto completa 100 transacciones distribuidas en forma individual durante una sesión. La prueba fue ejecutada con carga de un usuario proporcionando una respuesta peak para cada producto por una duración de 24 horas para comprobar si esta respuesta era sostenida en el tiempo.

Además de estas medidas absolutas de rendimiento, una medida de precio-rendimiento fue calculada para cada producto basándose en la configuración de 4CPU durante la prueba de 24 horas. Esta medida esta basada en el costo por transacción/segundo. El costo incluye los costos de hardware de la capa de negocios, las licencias de software correspondientes a los servidores de aplicación (basado en los precios publicados por cada producto configurado para 4 CPU) más el precio de los sistemas operativos usados por los servidores de aplicación. La medida de la relación precio-rendimiento no incluye bases de datos ni contratos de mantención.

A continuación se presenta una tabla resumen con los resultados de este benchmark:

Servidor de Aplicaciones	Carga máxima de Usuarios	Ordenes de transacciones ditribuidas por segundo	Precio / Rendimiento US$
Servidor de Aplicación J2EE WebLogic	4000	59	1305
Servidor de Aplicación J2EE WebSphere	1000	18	4722
.NET 1.0/ Windows 2000	4000	79	468
.NET 1.1/ Windows.NET	6000	117	316

Tabla Cap6_B, tabla resumen del benchmark de transacciones realizadas

6.3.4.3 Benchmark de servicios web.

Este ***benchmark*** mide las características de rendimiento de servicios web para J2EE y .NET. La prueba incluye dos configuraciones básicas:

- **Activación directa del servicio Web,** por 100 computadores distribuidos fisicamente cada uno simulando multiples usuarios haciendo llamadas ***SOAP*** directamente para activar los servicios Web. Este benchmark prueba la habilidad del servidore de aplicaciones para manejar solicitudes ***SOAP*** entrantes y actua como un proveedor de servicios Web Service.

- **Activación remota de los servicios Web** por el servidor de aplicación via un objeto proxy. En esta configuración, dos servidores de aplicación fueron configurados, uno para actuar como el proveedor de servicios Web remoto y el otro para actuar como un servicio Web cliente. Los 100 computadores generan carga hacienda solicitudes ***HTTP***/***HTML*** para el computador de servidores de aplicación, el cuál hace solicitudes ***SOAP*** a la máquina proveedora de servicios Web Service. Esta prueba esta diseñada para probar el rendimiento de los servicios Web clientes de los servidores de aplicación y el rendimiento cuando se hacen activaciones remotas a objetos basados en ***SOAP***.

A continuación se muestran las tablas con los resultados obtenidos:

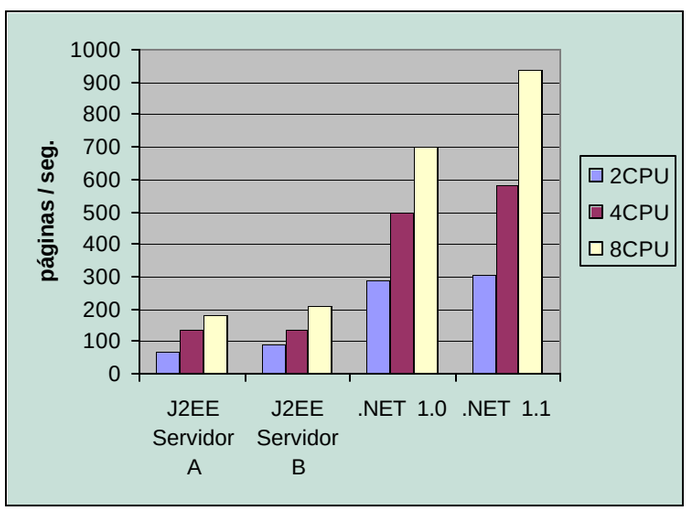


Figura Cap6_17, Mejores Respuestas para servicios WEB

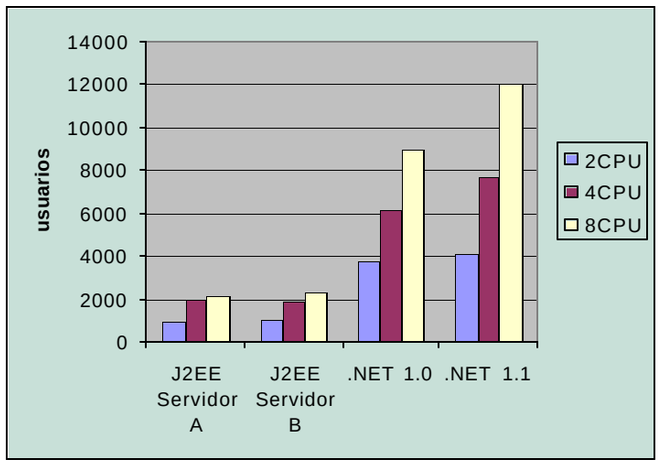


Figura Cap6_18, Carga máxima de usuarios para serivicios web

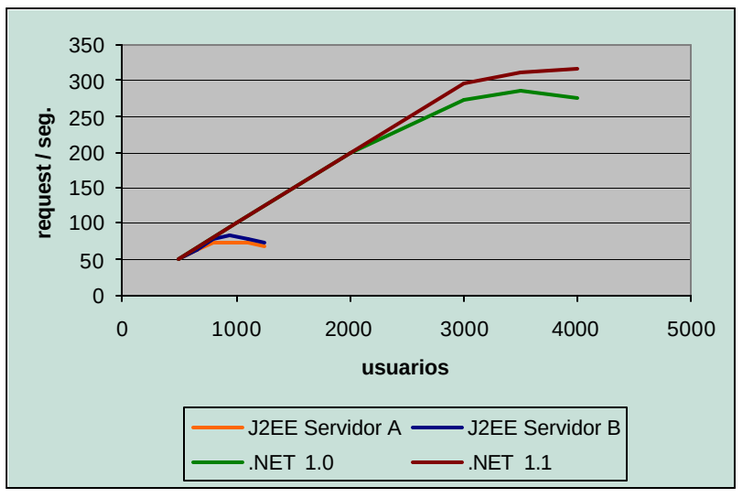


*Figura Cap6_19, Curvas de salida para llamadas a servicios web **SOAP** con 2 CPU*

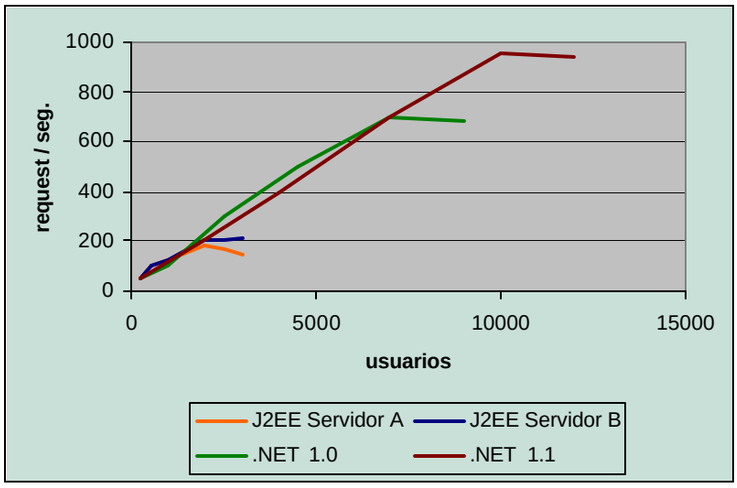


*Figura Cap6_20, Curvas de salida para llamadas a servicios web **SOAP** con 8 CPU*

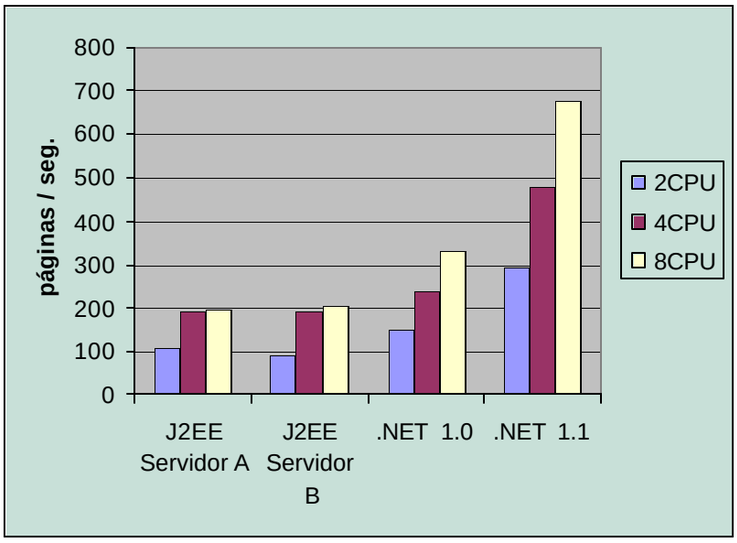


Figura Cap6_21, Mejor respuesta de servicios web via proxy

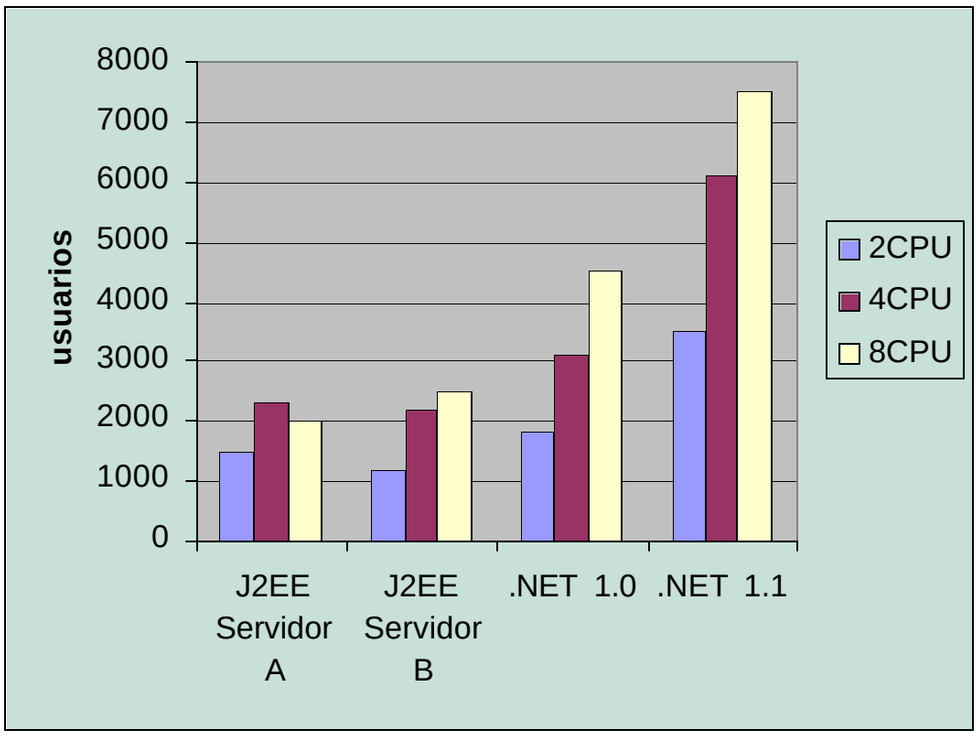


Figura Cap6_22, Carga máxima de usuarios para servicios web via proxy

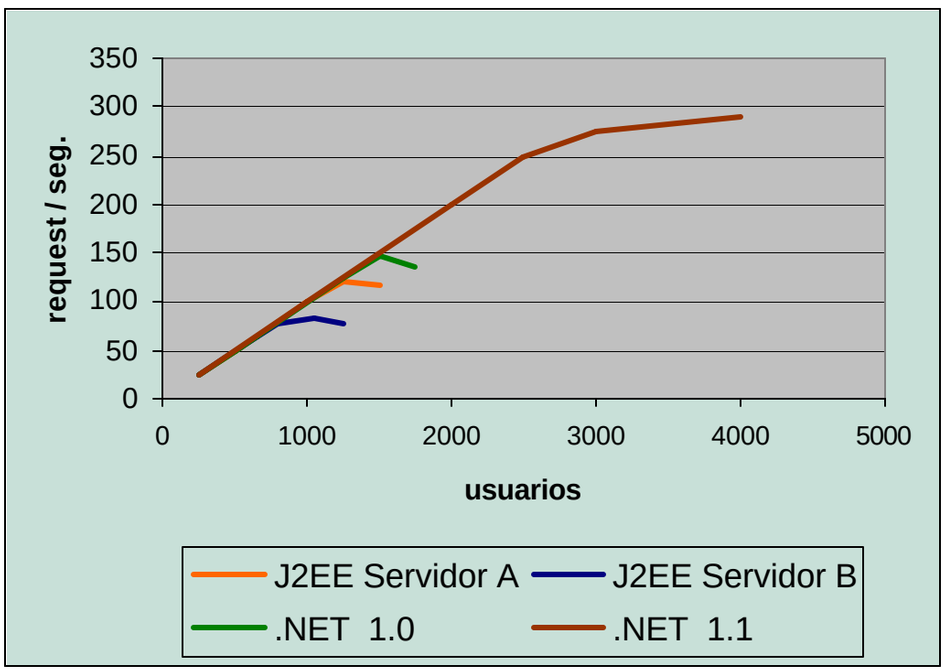


Figura Cap6_23, Curva de respuesta para servicios web via proxy con servidor de aplicación con 2 CPU

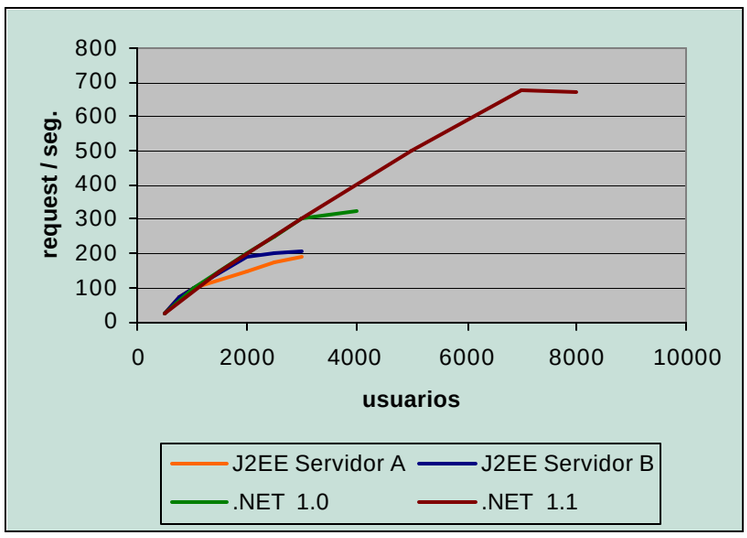


Figura Cap6_24, Curva de respuesta para servicios web via proxy con servidor de aplicación con 8 CPU

6.4 Utilizando la malla de evaluación.

Una vez obtenidos los resultados de los ***benchmarks***, se procede a obtener los valores para la utilización de la malla de evaluación.

6.4.1 Valores para los parámetros a evaluar

Para poder utilizar la malla de evaluación se comenzó por asignar la nota por cada parámetro, a continuación se describe el porque de cada una de las notas entregadas tanto a J2EE como a .NET. Para fundamentar algunos de estos valores se recurrió a los resultados obtenidos por el benchmark descrito en el punto 6.3.4

Las notas cuentan con dos decimales.

- **Páginas por segundo (sin descarga de imágenes):** al observar la figura *Cap6_8*, se puede determinar que el mejor y el menor rendimiento corresponden respectivamente a.NET 1.1 (nota 3) y WebLogic (nota 1).

Manteniendo la escala se obtienen los resultados, desplegados en la siguiente tabla:

Plataforma	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Valor	630	160	1150	1440
Nota	1.73	1	2.54	3

Tabla Cap6_C, Resumen por servidor para páginas por segundo sin descarga de imagen

- **Máximo número de usuarios en un mismo instante (sin descarga de imágenes):** al observar la figura *Cap6_9*, se puede determinar que el mejor y el menor rendimiento corresponden respectivamente a.NET 1.1 (nota 3) y WebLogic (nota 1). Manteniendo la escala se obtienen los resultados, desplegados en la siguiente tabla:

Plataforma	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Valor	8000	2150	14050	16250
Nota	1.82	1	2.68	3

Tabla Cap6_D, resumen por servidor paramáximo número de usuarios en un mismo instante sin descarga de imagen

- **Páginas por segundo (con descarga de imágenes):** al observar la figura *Cap6_13*, se puede determinar que el mejor y el menor rendimiento corresponden respectivamente a.NET 1.1 (nota 3) y WebLogic (nota 1). Manteniendo la escala se obtienen los resultados, desplegados en la siguiente tabla:

Plataforma	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Valor	1500	400	4050	6955
Nota	1.33	1	1.52	3

Tabla Cap6_E, resumen por servidor para páginas por segundo con descarga de imagen

- **Máximo número de usuarios en un mismo instante (con descarga de imágenes):** al observar la figura *Cap6_14*, se puede determinar que el mejor y el menor rendimiento corresponden respectivamente a.NET 1.1 (nota 3) y WebLogic (nota 1). Manteniendo la escala se obtienen los resultados, desplegados en la siguiente tabla:

Plataforma	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Valor	5100	850	9000	12900
Nota	1.70	1	2.35	3

Tabla Cap6_F, resumen por servidor paramáximo número de usuarios en un mismo instante con descarga de imagen

- **Máximo número transacciones en línea por segundo durante un tiempo dado:** al observar la tabla Cap6_B, se puede determinar que el mejor y el menor rendimiento corresponden respectivamente a.NET 1.1 (nota 3) y WebLogic (nota 1). Manteniendo la escala se obtienen los resultados, desplegados en la siguiente tabla:

Plataforma	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Valor	59	18	79	117
Nota	1.83	1	2.23	3

Tabla Cap6_G, *resumen por servidor para máximo número de transacciones en línea por segundo en un tiempo dado*

6.4.2 Malla final

La malla de evaluación final considera cada uno de los parámetros a cuantificar, además de la importancia (en porcentaje) de dicho parámetro. A cada uno de los diez parámetros cuantificados en la malla de evaluación, se le asignado la nota según el tipo de servidor y a la vez se ha calculado el porcentaje de importancia según la nota, este valor aparece entre paréntesis.

Parámetro	**Valor**	**WEBLOGIC**	**WEBSPHERE**	**.NET 1.0**	**.NET 1.1**
Páginas por segundo (sin descarga de imágenes)	5%	1.73 (0.0865)	1 (0.05)	2.54 (0.127)	3 (0.15)
Páginas por segundo (con descarga de imágenes)	10%	1.33 (0.133)	1 (0.1)	1.52 (0.152)	3 (0.3)
Máximo número de usuarios en un mismo instante (sin descarga de imágenes):	5%	1.82 (0.091)	1 (0.05)	2.68 (0.134)	3 (0.15)

Parámetro	Valor	WEBLOGIC	WEBSPHERE	.NET 1.0	.NET 1.1
Máximo número de usuarios en un mismo instante (con descarga de imágenes):	10%	1.70 (0.17)	1 (0.1)	2.35 (0.235)	3 (0.3)
Máximo número transacciones en línea por segundo durante un tiempo dado	20%	1.83 (0.366)	1 (0.2)	2.23 (0.446)	3 (0.6)
*Líneas de código	5%	2 (0.1)	2 (0.1)	3 (0.15)	3 (0.15)
*Seguridad	20%	3 (0.6)	2.85 (0.57)	2.1 (0.42)	2.1 (0.42)
*Soporte	10%	1.8 (0.18)	2 (0.2)	3 (0.3)	3 (0.3)
*Escalabilidad	5%	3 (0.15)	3 (0.15)	2 (0.1)	2 (0.1)
*Administración	10%	1.9 (0.19)	2 (0.2)	3 (0.3)	3 (0.3)
TOTALES	100 %	**2.0665**	**1.72**	**2.364**	**2.77**

Tabla Cap6_H, R*esultado de la malla de evaluación aplicada a los distintos servidores de aplicación*

***** Estas notas corresponden a apreciaciones resultantes del ánalisis teórico realizado.

A partir del resultado de la malla de evaluación se puede determinar que la plataforma más adecuada para el nuevo sitio web para ***BECH*** es Microsoft .NET.

Además es necesario hacer notar que debido a que actualmente ***BECH*** maneja herramientas y tecnología Microsoft los problemas de migración no debieran ser tantos como si se tratará de un cambio de proveedor tecnológico, como el que hubiera significado un servidor de aplicaciones J2EE.

CAPITULO 7

PROPUESTA DE PLAN DE IMPLEMENTACION

7. Propuesta de plan de implementación.

En este capítulo se toman las soluciones teóricas diseñadas en el capítulo tres y se crea un plan de implementación para ellas bajo la plataforma Microsoft.NET seleccionada en el capitulo seis, este plan contiene las actividades a seguir y los tiempos esperados para la finalización de cada una de ellas. Contempla el poblamiento de la nueva versión del sitio con todas las funcionalidades que actualmente tiene ***BECH***.

7.1 Comparación entre DNA y .NET.

Debido a que el plan de implementación se basa en migrar desde ***DNA*** a .NET, a continuación se presentan algunas consideraciones al respecto.

Las similitudes entre estas dos tecnologías son :

- Ambos comparten el modelo de tres capas planteado originalmente por ***DNA***.
- La entrada al modelo desde el cliente sigue siendo por el protocolo ***HTTP***.

Las diferencias a considerar son:

- ***ASP*** es ahora ASP.NET, lo que acarrea beneficios como: código compilado, utilitarios que facilitan el seguimiento del código, soporte de cualquier lenguaje .NET, soporte nativo a servicios web.
- ***ADO*** es ***ADO***.NET, con el beneficio que acarrea el uso integrado de ***XML*** debido a ser un estandar abierto.
- MSXML es ahora la clase ***XML*** .NET.
- Los componentes ***COM*** son ahora componentes .NET, a construir con un de los lenguajes soportados por esta plataforma, en particular el

por defecto (C#), lo que implica que no se necesita registrar dichos componentes como en el caso de una ***DLL COM***.

7.2 Plan de implementación.

Cumple con la función de especificar los costos, en cuanto a tiempo y área administrativa que trabaja en cada actividad.

Debido a que no es posible dejar de desarrollar aplicaciones en un entorno tan dinámico como el existente en ***BECH*** se contempla una fecha límite en la cual todo desarrollo nuevo debe ser realizado tanto en la versión actual como en la nueva. A la vez se contempla un período de tiempo durante el cual se deben migrar todas las funcionalidades de la versión actual a la nueva, finalmente terminada la etapa de pruebas en ambiente de pre-producción se debe instalar la nueva versión en producción.

Se consideró un plazo total para el plan de implementación de 100 días, las actividades a contemplar para este plan de implementación se describen a continuación.

- **Definición y construcción de nuevos servicios de datos**: esta actividad contempla la reducción de servicios de datos para evitar sinonimia. Un ejemplo típico es unificar las consultas de saldos en un único servicio que diferencie por producto. Trabajo realizado por el área de datos, cuyo plazo de desarrollo es de 50 días.
- **Especificación y construcción de bases de datos:** para el funcionamiento del motor de presentación, módulo de negocio, sistem ***log*** y manejador de errores se deben crear las bases de datos adecuadas. Trabajo realizado por el área de seguridad de la información, cuyo plazo de desarrollo es de 40 días.

- **Especificación de los *XML* a utilizar.** Esto significa que deben ser especificados dichos archivos para cada servicio disponible por ***BECH.*** Trabajo realizado en conjunto por el área de datos y el área de informática, cuyo plazo de desarrollo es de 25 días.
- **Especificación y construcción de los *XSL* a utilizar,** por el motor de presentaciónservicio. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 50 días.
- **Construcción del motor de negocio**, una vez definidos los archivos ***XML*** y creados los ***XSL***, se debe integrar todos estos elementos para dar forma a dicho motor. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 25 días.
- **Construcción del manejador de errores**, se debe codificar en VB .NET. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 20 días.
- **Construcción del motor de presentación**, se debe implementar a partir de las tablas definidas, los archivos ***XSL*** y el motor de negocios. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 30 días.
- **Construcción de páginas dinámicas,** se codifican en código ***ASP***.NET. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 80 días.
- **Construcción sistema log transaccional.** Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 23 días.
- **Generación de herramientas administrativas.** Conjunto de herramientas que permitan configurar el motor de presentación y el manejador de errores. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 32 días.

- **Instalación de la nueva versión obtenida en ambiente de desarrollo,** para realizacón de pruebas de funcionalidad. Trabajo realizado por el área de informática, cuyo plazo de desarrollo es de 3 días.
- **Pruebas y correcciones de la nueva versión en desarrollo.** Verifcación de funcionalidades instaladas y corrección de posibles errores. Trabajo realizado en conjunto por el área usaria de internet y el área de informática, cuyo plazo de desarrollo es de 10 días.
- **Instalación en ambiente de pre-producción.** Trabajo realizado por el área de administración de cambios, cuyo plazo de desarrollo es de 3 días.
- **Pruebas de la nueva versión en pre-producción.** Verifcación de funcionalidades instaladas y detección de posibles errores. Trabajo realizado por el área usaria de internet, cuyo plazo de desarrollo es de 2 días.
- **Instalación en producción.** Trabajo realizado por el área de administración de redes, cuyo plazo de desarrollo es de 2 días.

Una carta Gantt de estas actividades se encuentra en el Anexo Nº 3, el día de inicio es el 1 de Julio del presente y el de término el 17 de Noviembre de 2003.

CAPITULO 8

CONCLUSIONES

8. Conclusiones.

El estudio realizado en este proyecto de tesis contempla la mayoría de las aristas involucradas en realizar un diseño modular para la creación de un nuevo sitio web para ***BECH***.

Este diseño es de alta complejidad debido a la cantidad de factores involucrados, tales como: la importancia de ser una institución estatal, la cantidad de usuarios, la existencia de un sitio web que sigue incorporando desarrollos y los estándares de calidad existentes en el competitivo mercado bancario a nivel nacional.

Los aportes de esta tesis son:

- Establecer un marco teórico que describe soluciones a los principales problemas de un sitio web del estilo de ***BECH***, posibles de extrapolar a cualquier sitio web transaccional, debido a que dichas soluciones son módulos complementarios e independientes de la tecnología en la que pudieran ser implementados.
- Se presentó una implementación teórica de las soluciones propuestas a los principales problemas del actual sitio web de ***BECH,*** contemplando las tecnologías de desarrollo usadas actualmente por dicha institución, lo que permite contar con una versión optimizada del sitio en caso de que la institución decida postergar el cambio de plataforma.
- Establecer la mejor plataforma de servidores de aplicación disponible en el mercado para un sitio web empresarial de las características de ***BECH.***
- Se determina un plan de implementación para las etapas involucradas en la construcción de la nueva versión del portal internet de ***BECH***. Lo que permite contar con una evaluación sobre los costos de tiempo

a considerar al momento de implementar la nueva versión del sitio web.

A la vez se pueden mencionar las siguientes mejoras al presente trabajo:

- Considerar el diseño de herramientas administrativas para el motor de presentación y manejador de errores que permitieran la configuración de estos módulos por parte de algún utilitario.
- Incorporar sistemas de gestión. Debido a que el sistema ***log*** propuesto como solución permite registrar cada transacción realizada, se podría diseñar un módulo de gestión que realizara consultas sobre dichos registros con miras a realizar análisis de marketing o estadísticas relacionadas con el negocio.

BIBLIOGRAFIA

Autor	Descripción
(Steve McConell 1996)	Desarrollo y gestión de proyectos informáticos.1ra Edición,España: McGraw-Hill.
(Pressman 1993)	PRESSMAN, R. (1993). Ingeniería de Software, un Enfoque Práctico. 3ra Edición, España: McGraw-Hill.
Douglas K. van Duyne, James A. Landay, Jason I. Hong (2003)	The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. 2º Edición,Addison-Wesley
CHIKOFSKY, E.	Reverse engineering and design recovery: a taxonomy", IEEE Software, vol. 7, nº 1, enero, 1990.
Arnold, R. S.	Software Reengineering, IEEE Computer Society Press, Los Alamitos, California, 1993.

REFERENCIAS ELECTRONICAS.

URL	Descripción
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndna/html/msdn_windnapps.asp	Microsoft DNA
http://www.microsoft.com/net/	Sitio oficial de Microsoft .NET
http://www.microsoft.com/net/basics/framework.asp	Microsoft .NET
http://msdn.microsoft.com/library/?frame=true	Microsoft .NET
http://www.gotdotnet.com	Microsoft .NET
http:/java.sun.com/j2ee/	Sitio oficial de J2EE
http://www.flashline.com/Content/topics/j2ee.jsp	Conceptos J2EE
http://www.j2eeolympus.com/J2EE/ejb/EJB.jsp	Introducción a EJB
http://www.middleware-company.com/j2eedotnetbench/	Benchmark entre .NET y J2EE
http://www.serverwatch.com/tutorials/article.php/1355131	Servidores web versus servidores de aplicación.
http://www.woomeranet.com.au/bazaar.enterprise2.html	Comparación de servidores de negocio

Apéndice Nª 1, Abreviaturas

Las siglas y abreviaturas más utilizadas en este proyecto son:

ADO	:	ActiveX Data Object	Objeto de datos ActiveX
API	:	Application Program Interface	Interfaz de Programas de Aplicación
ASP	:	Active Server Page	Página Activa del Servidor
B2B	:	Business to Bussines	Empresa a Empresa
B2C	:	Business to Costumers	Empresa a Consumidor
BECH	:	Banco del Estado de Chile	
CICS	:	Customer Information Control System	Sistema De Control De la Información Del Cliente
COM	:	Component Object Model	Modelo de Componentes de Objeto
CPU	:	Central Processing Unit	Unidad Central de Procesamiento
DFD	:	Data Flow Diagram	Diagrama de Flujo de Datos
DLL	:	Dynamic Link Library	Librería de Enlace Dinámico
DMZ	:	Demilitarized Zone	Zona Desmilitarizada
DNA	:	Distributed InterNet Architecture	Arquitectura Distribuidas en Internet.
EJB	:	Enterprise JavaBeans	
FTP	:	File Transfer Protocol	Protocolo de Transferencia de Archivos
HTML	:	HyperText Markup Language	Lenguaje de Descripción de HiperTexto
HTTP	:	HyperText Transport Protocol	Protocolo para Transferencia de HiperTexto
IBM	:	International Business machines	Maquinas de Negocio Internacional
ID	:	ID	Identificador
IIS	:	Internet Information Server	Servidor de Información Internet
IP	:	Internet Protocol	Protocolo Internet
ISA	:	Industry Standrad Architecture	Arquitectura Estándar de la Industia
ISP	:	Internet Service Provider	Proveedor de Servicio Internet
JDBC	:	Java DataBase Connectivity	Conectividad a Bases de Datos Java
JMS	:	Java Message Service	Servicio de Mensajes java
JNDI	:	Java Naming and Directory Interface	Interfaz Java para Nombres y Directorios
JSP	:	Java Server Page	Página del Servidor Java
JTA	:	Java Transaction Api	API para Transacciones Java
LAN	:	Local Area Network	Red de Area Local
LOG	:	LOG	Registro
MTS	:	Microsoft Transaction Server	Servidor Transaccional
NFS	:	Network File System	Sistema de Archivos de Red
SNMP	:	Simple Network Management Protocol	Protocolo Simple de Administración de Redes
SMTP	:	Simple Mail Transfer Service	Protocolo Simple para Transferencia de Correo
SP	:	Store Procedure	Procedimiento Almacenado
SQL	:	Structured Query Language	Lenguaje de consulta estructurado
SSL	:	Secure Sockets Layer	Puerta de Enlace Segura
SSN	:	Secure System Network	Sistema de Red Segura
TCP	:	Transmission Control Protocol	Protocolo de Control de la Conexión

URL	:	Uniform Resource Locator	Localizador Uniforme de Recursos
VB	:	Visual Basic	Visual Basic
VPN	:	Virtual Private Network	Red Privada Virtual
WAP	:	Wireless Application Protocol	Protocolo de Aplicación Inhalambrico
WWW	:	World Wide Web	Telaraña de Ambito Mundial
XML	:	Extensible Markup Language	Lenguaje de Marcas Extensible
XSL	:	Extensible Stylesheet Language	Lenguage de hoja de estilos extensible

APENDICE N° 2, GLOSARIO DE TERMINOS

- **Cache:** mecanismo de alta velocidad para almacenamiento.
- **Componente:** archivo del tipo librería, es decir es código ya compilado, que usualmente se utiliza para conexiones a bases de datos y para aplicar reglas de negocio.
- **CucBech:** Cliente universal de ***CICS*** para ***BECH***, es un componente propietario, utilizado para acceder a las bases de datos ***IBM HOST***
- **DLL:** Es una componente creada en lenguaje Visual Basic, formada por una o varias clases las cuales contienen un conjunto de métodos que son los encargados de realizar las distintas funcionalidades que posee la DLL.
- **Firewall:** sistema diseñado para prevenir acceso no autorizado hacia o desde una red privada, pueden ser implementados por hardware o software.
- **Información de negocio:** consiste en los datos propios del quehacer de ***BECH***, es decir: datos de clientes, valores financieros, archivos de saldo, etc. No entran en esta categoría la información necesaria para interpretar el diseño a desplegar.
- **Jscript:** implementación de Microsoft del lenguaje JavaScript.
- **Página dinámica:** archivo propio del lenguaje utilizado por el servidor web, este archivo contiene código ejecutable tanto en el cliente como en el servidor, siendo este último el de mayor relevancia, ya que es el que le da dinamismo a su funcionamiento.

- **Plantilla:** archivo que permite aplicar un diseño o formato a una respuesta estructurada.
- **Plataforma:** hardware o software que definen un estándar sobre el cual se puede desarrollar un sistema.
- **Respuesta estructurada:** se entiende por este concepto una salida de datos que contenga herramientas de búsqueda tales como índices, arreglos u otro, que permitan interpretar con mayor facilidad su contenido.
- **Respuesta no estructurada:** datos en forma de texto sin interpretar.
- **Servicio de datos:** programa que se encuentra dentro del ambiente de una base de datos y que accesa distintas tablas para entregar una respuesta a quien lo invoque.
- **Sitio web financiero:** se utiliza este concepto para especificar el rol de negocio del sitio web de ***BECH***, es decir, no se trata de un sitio comercial (***B2B***,***B2C***, etc) o de otro tipo, sino de uno que manejará conceptos comunes al área financiera. Este sitio web financiero será formado por la unión de dos sitios: el público y el privado, en este capítulo se usa este concepto en general para cualquier institución y se hará hincapié al momento de ejemplificar con ***BECH***.
- **Scripts:** lista de comandos que pueden ser ejecutados sin la intervención de un usuario.
- **VBScript:** lenguaje script desarrollado por Microsoft, usado en el navegador Internet Explorer.

ANEXO 1. (Lista de Servicios)

Lista completa de servicios ofrecidos a los clientes de BancoEstado.

	Servicio	Cliente
1	Simulación Ahorro Estudio Seguro	Servicio Público
2	Simulación Ahorro Leasing	Servicio Público
3	Simulación de Crédito Universitario	Servicio Público
4	Simulación de Crédito Personal	Servicio Público
5	Información de Tasas de pizarra y de Monedas	Servicio Público
6	Cartola Cta Cte	Empresas- Instituciones-Personas
7	Cartola Línea de Crédito	Empresas- Instituciones-Personas
8	Saldo Cta Cte	Empresas- Instituciones-Personas
9	Saldo Línea de Crédito	Empresas- Instituciones-Personas
10	Saldo de Ahorro	Empresas-Instituciones-Personas
11	Saldo Visa	Personas
12	Cambio Clave Personal	Personas
13	Cartola Histórica de Cta Cte	Empresas- Instituciones-Personas
14	Cartola Histórica L. de Crédito	Empresas- Instituciones-Personas
15	Consulta Transacciones Realizadas en Internet	Empresas- Instituciones-Personas
16	Consulta de Cobranza (Estado al cedente)	Empresas- Instituciones-Personas
17	Consulta Cuota Crédito de Consumo a pagar	Personas
18	Consulta Dividendo Hipotecario a pagar	Personas
19	Cambio Clave Usuario y de Empresa	Empresas- Instituciones
20	Modificación de Perfiles por el Administrador	Empresas- Instituciones
21	Exportación de Cartolas a Planilla Excel	Empresas- Instituciones-Personas
22	Pago Línea de Crédito con cargo a Cta Cte	Personas
23	Pago Visa con cargo a Cta Cte	Personas
24	Pago cuota crédito de consumo con cargo a Cta Cte	Personas
25	Pago cuota crédito de consumo con cargo a Ahorro Vista	Personas
26	Pago dividendo con cargo a Cta Cte	Personas
27	Pago dividendo con cargo a Ahorro Vista	Personas
28	Transferencia de fondos Cta Cte --> Ahorro	Personas
29	Transferencia de fondos Cta Cte --> Cta Cte	Personas
30	Transferencia de fondos Ahorro --> Cta Cte	Personas
31	Transferencia de fondos Ahorro --> Ahorro	Personas
32	Transferencia de fondos Visa --> Cta Cte	Personas
33	Pago de Impuestos con Tesorería (Aduana, Contribuciones)	Empresas- Instituciones-Personas
34	Pago de Impuestos con SII (Renta, IVA)	Empresas- Instituciones-Personas
35	Transferencias de fondos para Empresas-Instituciones	Empresas- Instituciones

36	Transmisión de archivos de convenios para Empr-Instit	Empresas- Instituciones
37	Saldo de Chequera Electrónica	Personas
38	Cartola de Chequera Electrónica	Personas
39	Saldo Línea de Crédito de Chequera Electrónica	Personas
40	Cartola de Línea de Crédito de Chequera Electrónica	Personas
41	Cartola Histórica de Chequera Electrónica	Personas
42	Cartola Histórica de Línea de Crédito Chequera Electrónica	Personas
43	Transf. Fondos desde Cta Cte a Chequera Electrónica	Personas
44	Transf. Fondos desde Ahorro a Chequera Electrónica	Personas
45	Transf. Fondos Cheq. Electrónica a Cta Cte	Personas
46	Transf. Fondos Cheq. Electrónica a Ahorro	Personas
47	Pago Tarjeta Crédito con cargo a Chequera Electrónica	Personas
48	Pago Línea Crédito con cargo a Chequera Electrónica	Personas
49	Pago cuota Créd.Consumo con cargo a Chequera Electr.	Personas
50	Pago Dividendo con cargo a Chequera Electrónica	Personas
51	Transferencia de fondos hacia otros clientes del Banco	Personas
52	Pago Cotizaciones Previsionales (Previred)	Empresas-Instituciones-Personas
53	Consulta Saldo Ahorro Vivienda (desde aplicación)	Funcionarios Ministario Vivienda
54	Bloqueo Ahorro Vivienda (desde aplicación)	Funcionarios Ministario Vivienda
55	Ultimos Movimientos Tarjeta Visa	Empresas-Instituciones-Personas
56	Ultimos Movimientos Tarjeta Mastercard	Empresas-Instituciones-Personas
57	Saldo Tarjeta Mastercard (para Visa se extiende a Empresas)	Empresas-Instituciones-Personas
58	Avance en cuotas desde Tarjeta Mastercard	Empresas-Instituciones-Personas
59	Estado de Cuentas Tarjeta de Crédito Mastercard	Empresas-Instituciones-Personas
60	Simulación de Crédito Hipotecario	Servicio Público
61	Solicitud Preaprobación de Crédito de Consumo	Servicio Público

ANEXO N° 2 Lista de reportes de error

En el desarrollo de este proyecto de tesis se consideraron todos los reportes de error emanados desde la mesa de ayuda de internet que posee BECH (MAU), además de aquelloshechos por los distintos usuarios del sitio web del banco, usuarios perteneciente al área de banca a distancia, que son los encargados de probar contantemente el normal funcionamiento del sitio web de BECH.

Actualmente la mesa de ayuda (MAU) cada vez que recibe un reporte de error de algún cliente, redacta un correo electrónico, el cual es enviado al área de informática con un código único que lo identifica (ejemplo, MAU-1-4582215). Una evz que se resuelva el error, el área de informática debe indicar a la MAU que el reporte ha sido solucionado y cual era la causa de éste.

De la lista original se han tomado algunos ejemplos, los cuales se ilustran a continnuación.

FECHA	NOMBRE	ASUNTO	Nº REPORTE	STATUS	SOLUCION	FECHA CIERRE
18/09/02	Claudia Trigo	Adjunto mensaje de error que se está desplegando en Pago con Tarjetas. "Señor (a): Error:-2147012889 The server name or address could not be resolved por favor intenmte más tarde".	Reporte MAU-1-4323091	**Cerrado**	Problemas de comunicación con Transbank	18/09/02
02/10/03	Ctas. Ahorro Ch.Exterior	Se han recibido varios reportes de clientes que estan en el exterior que no pueden enviar giros electronicos (Solicitud electrónica de giro en cuenta de ahorro). Este mensaje, aparece al hacer clic en cualquiera de los tres formularios disponibles para Chilext. Siendo que cada cliente tiene el tipo de cuenta que solicita el mensaje..	Reporte MAU-1-4458606	**Cerrado**	este tema está resuelto. Lo que aún falta es el segundo reporte que dice respecto de transferencias de fondos desde cuentas de ahorro hacia otros productos del mismo rut, está reportado y la espera de solución, este problema afecta potencialmente a todos los clientes de ahorros no solo a los Chilenos en el exterior.	11/10/02
15/11/03	Lucila Contreras	Cliente : LUCILA CONTRERAS VERA, Rut : 8.545.297-5 Situación: Pos Banco en Línea sólo accede a la información fechada hasta 27.10.2002 de su tarjeta de crédito. No accede al último estado de movimiento, donde registra 16 transacciones.	Reporte MAU-1-4590145	**Cerrado**	Faltaba cargarle los movimientos en el Host al Cliente , se encuentra regularizado en Producción	15/11/02
03/12/02	Eduardo Fernández Escobar	Pago de Dividendos por Internet: Respecto al crédito hipotecario, tenía entendido lo siguiente: 1.- Que el cupón de dividendo se encontraba disponible en bco. en línea a partir del 25 (mes anterior) y hasta el 10 (mes de vencimiento) o hasta que se cancelara, el evento que primero se diera. 2.- Que no se podían hacer pagos electrónicos de dividendos atrasados o adelantados. Al parecer, ambos puntos han cambiado, favor que nos envíen la información correcta, ya que los clientes nos están reclamando, te copio los mensajes de un cliente.	Reporte MAU-1-4590345	**Cerrado**	Faltaba cargarle los movimientos en el Host al Cliente , se encuentra regularizado en Producción. En forma normal, estos son cargados los días 25 de c/mes hasta el vencimineto de la deuda.	03/12/02
03/01/03	Magaly Messina Navarro	Transferencia de Fondos, Rut 6.034.808-1 Informo de problema de cliente (Funcionaria banco): Al realizar transferencia de fondos desde cuenta corriente a cuenta ahorro vista se genera error ""Microsoft VB Script runtime error '800a005' Invalid procedure call or argument /transa/transfer/respuesta.asp""	Reporte MAU-1-4582214	**Cerrado**	Se Solicito el paso a Pre-Producción con la solución y modificación de la Página Respuesta.asp Nº 24501. Con fecha 09/01/2003 se solicita el paso a Producción.	10/01/03
06/01/03	Mensaje a clientes en sitio Privado.	Se requiere colocar un mensaje a los clientes que ingresen al Sitio Privado, en el sector Cuenta de Ahorro y Trasferencia de Cuentas de Ahorro. Debe salir un mensaje que diga "Estimado cliente a partir del 1° de Enero de 2003 se modifican las comisiones de Cuentas de Ahorro", con dos botones, uno que diga "OK", "Bien" o "Salir".	Sin Nº	**Cerrado**	Listo en Desarrollo, se debe solicitar el paso a Pre y Producción. Se necesita en forma Urgente el paso a Producción de la Solicitud Nº 24301, esta corresponde a la transferencia de Personas por Ahorro Giro Incondicional y los Usuario la requieren en forma Urgente. En Producción	09/01/03

FECHA	NOMBRE	ASUNTO	Nº REPORTE	STATUS	SOLUCION	FECHA CIERRE
19/01/03	Juan Guevara Oliver	Cliente: Sr. Juan Guevara Oliver, Rut: 6.011.905-8 Problema: Cliente al ingresar al menú de pago de su crédito hipotecario, no se le despliega la cuenta de cargo, esta debe ser el Ahorro Vista 166095620	Reporte MAU-1-4716495	**Cerrado**	Se modificó Página que contenía el error	21/01/03
24/01/03	Vasquez Del Miglio Rodrigo	Problemas en pag. web en la opción pago automático con tarjeta de crédito, adjunto correo donde informan del problema. Te envío print de pantalla con error en el pago automático con tarjeta de crédito , en este caso en mastercard.	Reporte MAU-1-4723498	**Cerrado**	Corresponde a Problemas de Comunicación, ya sea por Transbank o por Nosotros esto lo ve el señor Coyopae Hernandez Sergio Rafael o en su defecto el Sr. Elgueta Elgueta Roberto. El problema no es de la Aplicación	25/01/03
26/01/03	Erwin Héctor González Candia	RUT:6.663.580-5 Situación: Accede a Banco en línea, ingresa a la opción desde cta. de ahorro, en la columna DESDE solo aparece su cta. giro diferido (53366422389) y no aparece la cta. ahorro a plazo (52160974575). En la columna hacia se despliegan todas sus ctas.: Cuenta Corriente, cta. ahorro plazo y giro diferido.	Reporte MAU-1-493666	**Cerrado**	El cliente no tiene problemas, se revisaron los antecedentes enviados y parecen las dos cuentas que mencionas, en el adjunto sale la información. Favor cerrar el reporte.	27/01/03
24/02/03	ENRIQUE BERNAIN DESMARAS	RUT:3.183.272-1 Problema: Al ingresar a Pago Automático con tarjeta de crédito sistema arroja mensaje "Servicio exclusivo para cliente con tarjeta de crédito bancoestado", siendo que cliente si tiene tarjeta de crédito bancoestado.	Reporte MAU-1-5061365	**Cerrado**	Cliente si puede pagar la Tarjeta de Crédito Visa desde CTV CCT. Al día 24.02 aún no estaba cargado	25/02/03
10/03/03	Pago Tarjetas	Al chequear Pago de cuentas con tarjeta de crédito sistema responde con Time Out. Adjunto print de pantalla	Reporte MAU-1-6971514	**Cerrado**	Problemas con el sitio de www.pagocontarjetas.cl	10/03/03
29/03/03	Progreso S.A.	cliente no puede ver saldos para estado cedentes, los veía hasta el miércoles pasado 09-04-03. Envío print de pantalla, tiene IE 5.0, se borraron temporales, cookies y continúa problema.La consulta la hizo el Ususario Camilo Muñoz Cid Rut 11.256.571-K	Reporte MAU-1-7118123	**Cerrado**	El problema radicaba en que en la B.D. de producción no estaba el SP que entregaba los Datos, por esto se solicitó nuevamente la creación del SP. Erick, el que vale es el de srv_desarrollo3 existe el SP con el nombre Svc_EncEst el que internamente esté como Svc_svc_EncEst nos da lo mismo y esto es debido a que todas las páginas *.asp (son varias), hacen llamadas a este SP en cambio la modificación en la Base es una sola.	30/03/03
05/04/03	Marcel Farias Leon	RUT:10.127.335-0, Cliente Lleva varios dias tratando de pagar el credito hipotecario y sale un mensaje que dice "Señor: Ha ocurrido un problema, por favor reintente posteriormente" ¿que puedo o debo hacer?.	Reporte MAU-I-73315258	**Cerrado**	El problema radica en que el único archivo safe Internet Hipotecario que existía a contar del 24.03.2003 se separó en dos archivos uno para Mutuos y otro para Letras, la aplicación contempla la modificación desde marzo en Pre-Producción situación que aún no ha sido certificada (Sol.Nº 26813).	06/04/03
27/04/03	Estudio Seguro	Se detecta irregularidades de valores UF y PESOS en simulacion ahorro estudio seguro	Reporte MAU -1-8076520	**Cerrado**	Si lo que se está informando corresponde al simulador, esto esta correcto (hoy $17.014,45 para hoy), indica la uf correcta. Debe cerrarse el reporte.	27/04/03

FECHA	NOMBRE	ASUNTO	Nº REPORTE	STATUS	SOLUCION	FECHA CIERRE
12/05/03	Claudia Trigo	Cliente rut:9.211.654-9 al ver su cartola de cuenta corriente le muestra saldo en linea de credito, siendo que el cliente no posee linea de credito con el banco. ESTE PROBLEMA YA SE HABIA SOLUCIONADO, O NO?	Reporte MAU -1-8046629	**Cerrado**	El problema se presento en una de las maquinas de produccion, pues en ella no se encontraba la version actualizada de la página de cartola para cuenta corriente	12/05/03
25/05/03	MAU	Al consultar por los estados al cedente en aplicación de empresas, entrega mensaje de conección time out, dando un error sin el formato del banco, que ocurre?	Reporte MAU -1-9059520	**Cerrado**	Lo que ocurre que en esa página se esta acediendo directamente a las bases de datos sin pasar por inguna componente, se corrije el problema y se presenta el error con el formato del banco	28/05/03
29/05/03	MAU	Cliente rut: 8.540.337-0 al consultar por el saldo en su tarjeta de credito visa, entrega mensaje de “ha ocurrido un error”	Reporte MAU -1-91008510	**Cerrado**	Ocurre que e servicio SAIE023, saldo de tarjeta de credito, esta arrojando un error no manejado por la componente de consulta de saldos, servicio corregido y componente modificada.	04/06/03
10/06/03	Rodrigo Mazo	Al solicitar el crédito hipotecario no se despliegan los campos de comunas , ciudades ni sucursales, aparecen los combos sin valores	Reporte MAU -1-96003792	**Cerrado**	El problema se debia a que la bases generales y la tabla ttab_gen se encontraban cerradas,favor cerrar reporte.	10/06/03

ANEXO N° 3, Carta Gantt con plan de implementación

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Proyecto Migración Sitio Bech	100 días	ma 01/07/03	lu 17/11/03
2	Definición y construcción de nuevos Servicios de Datos	50 días	ma 01/07/03	lu 08/09/03
3	Especificación y construcción de las bases de datos	40 días	ma 01/07/03	lu 25/08/03
4	Especificación de los archivos XML a utilizar	25 días	ma 01/07/03	lu 04/08/03
5	Especificación y construcción de los archivos XSL a utilizar	50 días	ma 01/07/03	lu 08/09/03
6	Construccion del motor de Negocio	25 días	ma 09/09/03	lu 13/10/03
7	Construccion del Manejador de Errores	20 días	vi 12/09/03	ju 09/10/03
8	Construccion del Motor de Presentacion	30 días	ma 09/09/03	lu 20/10/03
9	Construccion Paginas Dinámicas	80 días	ma 01/07/03	lu 20/10/03
10	Construccion Sistema Log Transaccional	23 días	lu 15/09/03	mi 15/10/03
11	Generación de herramientas administrativas	32 días	ju 21/08/03	vi 03/10/03
12	Instalacion de nuevo sitio en ambiente de Desarrollo	3 días	ma 21/10/03	ju 23/10/03
13	Prueba y corrección en ambiente de Desarrollo	10 días	vi 24/10/03	ju 06/11/03
14	Instalacion en ambiente de Pre-producción	2 días	vi 07/11/03	lu 10/11/03
15	Pruebas en Ambiente de Pre-producción	2 días	ma 11/11/03	mi 12/11/03
16	Paso a Produccion	2 días	ju 13/11/03	vi 14/11/03