UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"REPORTE DE EJERCICIO PROFESIONAL – DESARROLLO DE APLICACIONES PARA LOS SECTORES PÚBLICO Y PRIVADO EN PLATAFORMA JAVA"

INFORME DEL EJERCICIO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

JANET CRUZ ÁNGEL

ASESOR:

MAT. LUIS RAMIREZ FLORES

SAN JUAN DE ARAGÓN, EDO. MÉX., OCTUBRE DE 2010

A mi mamá PIEDAD,

por todo el amor y entrega
con los que ha guiado siempre
mi vida.

AGRADECIMIENTOS

De manera muy especial a mi profesor y asesor de titulación, Matemático Luis Ramírez Flores, quien con su invaluable apoyo me permitió continuar con mis actividades laborales y al mismo tiempo trabajar en este reporte.

A la Jefa de Carrera de Ingeniería en Computación, la Dra. Nelly Rigaud Téllez, cuya disponibilidad facilito todo el proceso administrativo que estuvo en sus manos.

A los profesores M. en C. Marcelo Pérez Medel, M. en C. Jesús Hernández Cabrera, Maestra Silvia Vega Muytoy y al Ingeniero Enrique García Guzmán, por el tiempo dedicado a la revisión de mi informe y por su buena disposición para el mismo.

A mis hermanos Javier, Antonio y Norma, porque desde siempre han estado a mi lado para brindarme amor, apoyo y consejos.

Y finalmente, un agradecimiento especial a todas aquellas personas que de forma anónima a lo largo de mi andar en el ámbito laboral, han contribuido con sus conocimientos a enriquecer mi experiencia profesional.

CONTENIDO

INTRODUCCIÓN

Mi experiencia profesional como consultor de sistemas se basa principalmente en el uso de las herramientas de desarrollo Java y Java EE (o J2EE).

A lo largo de cinco años de desempeño en el mercado laboral, he aprendido técnicas de programación que me han permitido construir con relativa facilidad componentes de software y aplicaciones escalables y de fácil mantenimiento. También me he enfrentado a diversos desafíos cuyas enseñanzas constituyen hoy en día todo el bagaje técnico y administrativo que conforman mi perfil profesional de consultor.

El siguiente trabajo tiene como finalidad presentar una descripción del lenguaje de programación Java y de la plataforma J2EE (ahora Java EE). Enseguida, hago un breve resumen de las mejores prácticas que he aprendido y aplicado para el desarrollo de aplicaciones con dichas herramientas. Después, muestro una serie de problemas y soluciones que he aplicado de forma general en todos mis proyectos y que me han funcionado para obtener los resultados planteados.

Para finalizar, presento a detalle cada uno de los proyectos en los que he participado así como las herramientas que en ellos he empleado.

1. EL LENGUAJE DE PROGRAMACIÓN JAVA Y LA PLATAFORMA J2EE

1.1 Antecedentes

En informática, una plataforma es un ambiente de hardware o software en el cual se ejecuta un programa; la mayoría de las plataformas se describen como la combinación de un sistema operativo y el hardware que éste controla.

El lenguaje de programación Java nace como parte de un proyecto de investigación para desarrollar un software avanzado destinado a una extensa variedad de dispositivos de red y sistemas embebidos. El objetivo principal era desarrollar una plataforma pequeña pero a la vez robusta, confiable, portátil, distribuida y que operara en tiempo real.

En un inicio se eligió el lenguaje de programación C++ para desarrollar dicho software, pero pronto las dificultades encontradas con el mismo crecieron al punto de concluir que los problemas podrían ser resueltos creando por completo una nueva plataforma de programación. Así, se tomo lo mejor de una extensa variedad de lenguajes de programación como Eiffel, SmallTalk u Objective C, para definir la arquitectura de los que más tarde sería la plataforma Java.

El resultado fue el lenguaje de programación Java, una plataforma de software ideal para el desarrollo de aplicaciones seguras, distribuidas y que deben trabajar en diversos ambientes heterogéneos de hardware y software. El tipo de aplicaciones que pueden desarrollarse en Java destinadas al usuario final van desde aquellas embebidas en dispositivos inalámbricos hasta aquellas que se encuentran instaladas en las computadoras personales de los usuarios o aplicaciones que pueden accederse desde la Intranet o el mismo Internet por medio de un navegador web.

Java es un lenguaje de programación de alto nivel y al mismo tiempo una plataforma de software, conocida como plataforma Java, muy populares como herramientas de desarrollo de aplicaciones en el sector público y privado.

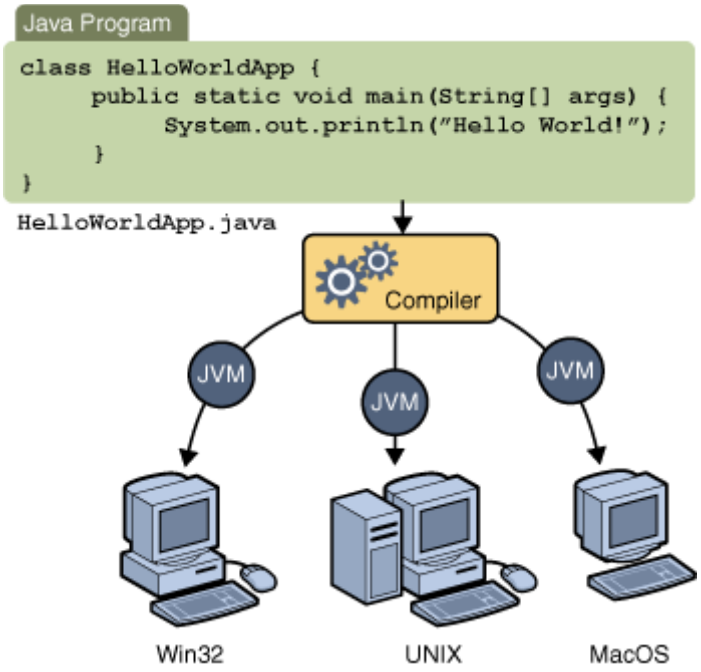
1.2 El lenguaje de programación Java

En el lenguaje de programación Java, todo el código fuente es primero escrito en archivos de texto plano con la extensión *.java*. Estos archivos son después compilados en archivos *.class* por el compilador Java *javac*. Un archivo *.class* no contiene código nativo a ningún procesador, lo que contiene son *bytecodes* – el lenguaje máquina de la Máquina Virtual de Java. El lanzador de Java (conocido como *java launcher*) es quien ejecuta la aplicación en una instancia de la Máquina Virtual.

Figura 1.1 ***Una visión general del proceso de desarrollo de software en Java***

Ya que la Máquina Virtual de Java está disponible para diferentes sistemas operativos, los mismos archivos *.class* pueden ejecutarse tanto en Microsoft Windows, en el sistema operativo Solaris OS, en Linux o en el sistema operativo Mac OS. La Máquina Virtual de Java efectúa pasos adicionales en tiempo de ejecución para dar un mayor desempeño a las aplicaciones, tales como encontrar cuellos de botella en tiempo real o recompilar a código nativo las secciones de código utilizadas frecuentemente por las aplicaciones.

Figura 1.2 ***Por medio de la máquina virtual, una misma aplicación es capaz de ejecutarse en diferentes plataformas (sistemas operativos)***



1.2.1 Características del lenguaje de programación Java[1]

Orientado a objetos

La necesidad de contar con sistemas distribuidos y basados en una arquitectura cliente servidor coincide con las características de encapsulamiento y del principio de ocultación de la programación orientada a objetos. La información que viaja a través de las aplicaciones debe estar encapsulada y la comunicación entre las mismas por seguridad, debe realizarse por medio de interfaces que describen que hacen, que necesitan y como lo hacen.

En la programación orientada a objetos, el encapsulamiento es la característica de reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. El principio de ocultación se refiere a que cada objeto está aislado del exterior y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un

[1] White Paper, The Java Language Environment. Ver bibliografía al final del documento.

objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas.

Para funcionar dentro de la creciente complejidad de los ambientes basados en las redes de información, el desarrollo de sistemas debe adoptar los conceptos de la programación orientada a objetos, razón por la cual Java fue diseñado para estar orientado a objetos.

Java proporciona una serie de librerías de objetos ya probados que permiten de manera transparente a los programadores el manejo de datos través de interfaces de entrada/salida (I/O) y de redes, así como el acceso a un juego de herramientas para crear interfaces graficas para los usuarios, por citar algunos ejemplos. La funcionalidad de estas librerías puede ser ampliada para proporcionar un nuevo comportamiento (lo que se conoce como herencia en la programación orientada a objetos) y satisfacer necesidades propias del sistema que se esté desarrollando.

Simple y Familiar

Java se caracteriza por ser un lenguaje de programación simple y fácil de aprender por los programadores ya que no requiere de largos procesos de entrenamiento para familiarizarse con el mismo. Los programadores habituados a lenguajes de programación orientados a objetos no tienen dificulta para adaptarse rápidamente a la programación de software con Java.

Confiable, Robusto, Distribuido y Seguro

El lenguaje de programación Java fue diseñado para crear software altamente confiable. Provee de una revisión extensiva en tiempo de compilación del código fuente así como un segundo nivel de revisión en tiempo de ejecución. El manejo de memoria así como la recolección automática de objetos “basura” los administra directamente el lenguaje de programación, por lo que el programador no tiene que definir ningún mecanismo de asignación o des asignación de memoria, lo que hace de Java un lenguaje de programación robusto.

La tecnología Java fue también diseñada para trabajar en ambientes distribuidos (computadoras con arquitecturas diferentes, diferentes sistemas operativos y diferentes interfaces gráficas de usuario), lo que significa que la seguridad es de altísima importancia. Java permite construir aplicaciones que no pueden ser invadidas desde fuera. Las aplicaciones desarrolladas en Java están a salvo de la intrusión de código no autorizado que intente crear virus o invadir el sistema de archivos (gracias al compilador Java y a la máquina virtual Java, que veremos más tarde).

Arquitectura Neutral y Portátil

Como se mencionó en el párrafo anterior, Java fue diseñado para soportar aplicaciones que serán instaladas en ambientes heterogéneos. En dichos ambientes, las aplicaciones deben ser capaces de ejecutarse en una gran variedad de arquitecturas de hardware y en una gran variedad de sistemas operativos, interactuando con diferentes interfaces de múltiples lenguajes de programación (C, Ensamblador, por ejemplo). Para lograr lo anterior, el compilador de Java genera a partir del código fuente bytecode, un formato de arquitectura neutral intermedia diseñado para transportar código eficientemente a múltiples plataformas de hardware y software.

Java es un lenguaje de programación interpretado, lo que le permite solucionar el problema de distribución de binarios y de versión; el mismo bytecode del lenguaje de programación Java se ejecutará en cualquier plataforma. ¿Cómo lo logra? Java es estricto en la definición del tamaño de sus tipos básicos de datos y del comportamiento de sus operadores aritméticos, por lo que los programas Java son los mismos en cualquier plataforma, no existen las incompatibilidades de tipo de datos entre las diferentes arquitecturas de hardware y software.

La neutralidad de la arquitectura de Java es en realidad una parte de la verdadera portabilidad de un sistema. La portabilidad y la arquitectura neutral de la tecnología Java es lo que se conoce como la Máquina Virtual de Java (Java Virtual Machine o *JVM*).

La Máquina Virtual de Java es la especificación de una máquina abstracta para la cual el compilador del lenguaje de programación Java puede generar bytecode. Existen implantaciones específicas de la Maquina Virtual de Java para plataformas de hardware y software en específico (hay una JVM para Windows otra para Linux, por ejemplo). Dicha máquina está basada en la especificación de la interfaz POSIX, una definición estándar industrial para una interfaz de sistema portátil. La implantación de la Máquina Virtual Java en nuevas arquitecturas es relativamente una tarea sencilla siempre y cuando la plataforma en cuestión cumpla con los requerimientos básicos como el soporte de múltiples hilos de ejecución (ejecución de diferentes tareas en diferentes hilos de ejecución).

El intérprete de Java puede ejecutar código bytecode directamente en cualquier máquina para la cual dicho intérprete y el sistema de ejecución hayan sido creados. En una plataforma interpretada como la basada en la tecnología Java, la fase de enlazado de un programa es simple, ligera (alto desempeño) y se efectúa de forma incremental.

Alto desempeño

Para lograr un alto desempeño, Java adopta un esquema con el cual el intérprete se ejecuta a toda velocidad sin necesidad de revisar el ambiente en tiempo de ejecución. Otra ventaja es que el colector de basura de la Máquina Virtual de Java (*automatic garbage collector*) se ejecuta en un hilo de fondo con baja prioridad, asegurando así una alta probabilidad de que la memoria estará disponible cuando se requiera, permitiendo de esta manera un mejor desempeño de las aplicaciones.

Las aplicaciones que hacen uso de una gran capacidad de cómputo son diseñados de tal forma que las secciones que necesitan de un "cómputo intensivo" puedan reescribirse en código máquina según se requiera y contar a la vez con su interfaz con Java. De esta manera, el usuario percibe que las aplicaciones interactivas responden rápidamente aun cuando éstas son interpretadas.

Múltiples Hilos de Ejecución y Dinámico

La capacidad multihilos de ejecución (*multithreading*) de Java permite a los programadores construir aplicaciones con varios hilos de actividades concurrentes que resultan con un alto grado de interactividad para con el usuario final.

Mientras que el compilador de Java es estricto en la revisión estática que realiza en tiempo de compilación del código fuente, el lenguaje junto con el sistema de ejecución son dinámicos en sus etapas de enlazado. Las clases son enlazadas conforme se requieran. Nuevos módulos de código pueden ser enlazados a demanda desde fuentes variadas, incluso desde fuentes remotas.

1.2.2 Ventajas de programar con Java

Resumiendo las bondades de programar con Java se puede destacar que:

- Java es fácil de aprender, sobre todo para los programadores ya familiarizados con C y C++
- Un programa Java puede ser hasta cuatro veces más pequeño que el mismo programa escrito en C++.
- El lenguaje de programación Java fomenta el uso de las buenas prácticas de programación, posee una recolección automática de los objetos que ya no están siendo utilizados para evitar "fugas de memoria". Su orientación a objetos así como la facilidad de heredar funcionalidad del API de Java permite la reutilización de código existente y ya probado, introduciendo un mínimo de defectos en la programación.
- Impide la dependencia de plataformas y permite que un programa sea portátil ya que evita el uso de librerías escritas en otros lenguajes. Esto lo logra con cada versión de la Máquina Virtual de Java que existe para cada sistema operativo.
- Ya que las aplicaciones escritas en el lenguaje de programación Java son compiladas en bytecodes, código independiente de hardware alguno, se ejecutan consistentemente sin problemas en cualquier plataforma Java.

1.3 La Plataforma J2EE[2]

Como se menciono ya anteriormente, Java también es una plataforma, un ambiente particular en el cual se ejecutan las aplicaciones escritas en lenguaje de programación Java.

[2] Understanding Java Platform, Enterprise Edition. Ver bibliografía al final del documento.

Existen tres plataformas para el lenguaje de programación Java:

- Java Platform, Standard Edition (Java SE)
- Java Platform, Micro Edition (Java ME)
- Java Platform, Enterprise Edition (Java EE) o Java 2 Platform Enterprise Edition (J2EE)

Todas las plataformas Java están constituidas de una Máquina Virtual Java y de la Interfaz de Programación de Aplicaciones o API de Java (en inglés *application programming interface*), que es una colección de componentes de software que pueden ser utilizados para crear otros nuevos componentes de software o de aplicaciones. A continuación se hablará brevemente de los dos primeros tipos de plataformas para después profundizar más en Java EE.

La Plataforma Java Edición Estándar (Java Standard Edition o Java SE)

El API de Java SE provee la funcionalidad central del lenguaje de programación Java. Define desde los tipos básicos de datos y de objetos hasta las clases de alto nivel que son utilizadas para establecer actividades como conexiones de red, seguridad, acceso a base de datos, desarrollo de interfaces graficas para el usuario (*graphic user interface*) y el parseo de archivos XML, por dar algunos ejemplos.

Adicionalmente al API, Java SE está compuesta por una Máquina Virtual, herramientas de desarrollo de aplicaciones, tecnologías de despliegue de aplicaciones y otras librerías de clases que son utilizadas comúnmente en las aplicaciones Java (impresión en consola, colecciones de datos, etc.).

La Plataforma Java Edición Micro (Java Micro Edition o Java ME)

La plataforma Java ME también provee de un API y de una versión de una Máquina Virtual para ejecutar aplicaciones escritas en lenguaje Java en pequeños dispositivos móviles, como los teléfonos celulares. El API de Java ME es un subset del API de Java SE junto con unas librerías de clases especiales. Muy frecuentemente las aplicaciones Java ME son clientes de los servicios que ofrecen las aplicaciones Java EE, como la descarga de juegos a los teléfonos celulares, desde un portal en Internet, por ejemplo.

La Plataforma Java Edición Empresarial (Java Enterprise Edition o Java EE)

La plataforma Java EE, también conocida como Java to Enterprise Edition (J2EE) hasta la versión 1.4, hace uso de la plataforma Java SE para funcionar. Provee también de un API y de un ambiente de ejecución que ayuda a los desarrolladores a desarrollar y ejecutar aplicaciones de red de gran escala, seguras, confiables, escalables, multicapas. Como su nombre lo indica, proporciona lo necesario para construir aplicaciones empresariales.

1.3.1 Una Visión general de las Aplicaciones Empresariales

Se conoce como “aplicaciones empresariales” a las aplicaciones robustas, seguras, escalables, confiables y multicapas, ya que están diseñadas para resolver problemas que enfrentan comúnmente las grandes empresas. Sin embargo, las aplicaciones empresariales no son solamente útiles a las empresas de gran tamaño, a las grandes agencias o a las dependencias gubernamentales, los beneficios son también de gran ayuda para las pequeñas empresas y para los mismos programadores que se enfrentan día a día a un creciente mundo conectado de una manera u otra a alguna red de información.

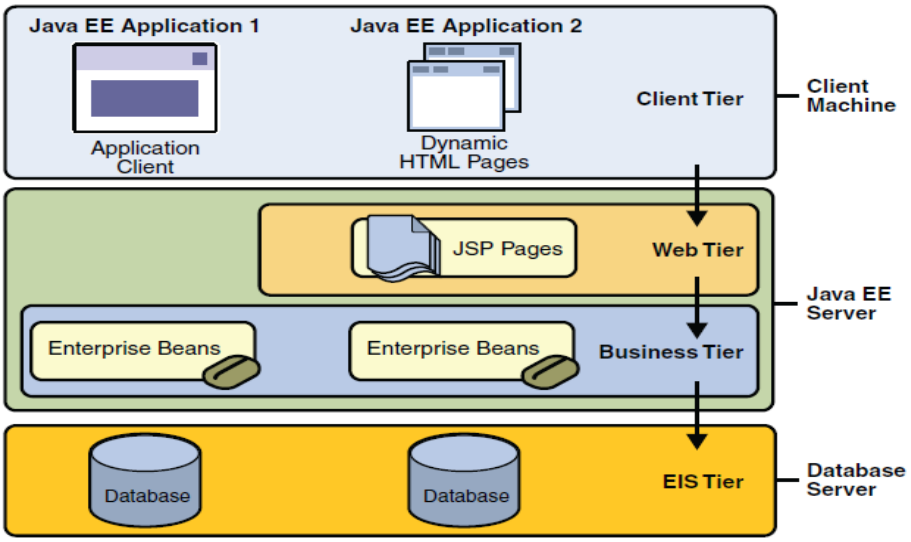
El hecho de que las aplicaciones sean seguras y confiables las hace frecuentemente complejas. La plataforma Java EE proporciona un modelo de desarrollo, una serie de librerías y un ambiente de ejecución que permite al programador enfocarse en la funcionalidad de la aplicación que va a desarrollar y no en la complejidad que representa hacerla robusta o segura. De ahí la razón por la cual Java EE es una plataforma ampliamente utilizada para construir aplicaciones empresariales tanto en el sector público como en el privado; por lo mismo, existe una amplia gama de proveedores de software como IBM, Oracle, Apache (entre otros), que implementan los estándares que definen a la plataforma Java EE.

1.3.2 Las Aplicaciones Multicapas

En una aplicación multicapas, la funcionalidad de la aplicación está separada dentro de áreas funcionalmente aisladas llamadas capas. Típicamente las aplicaciones multicapas cuentan con una capa del *cliente*, una capa de *negocio* y una capa de *datos* (llamada frecuentemente *capa de sistema de información empresarial*). La capa del cliente consiste en un programa que envía peticiones a la capa de negocio. La capa de negocio recibe éstas solicitudes y los datos a procesar por la aplicación, guardándolos de manera permanente en un almacén de datos (*datastore*) dentro de la capa de sistema de información empresarial.

El desarrollo de aplicaciones Java EE se concentra principalmente en la capa de negocio para hacer de la administración de las aplicaciones empresariales una tarea mucho más fácil, más robusta y más segura. A continuación, se describirán algunas de las capas involucradas en las aplicaciones empresariales y las tecnologías Java EE que se pueden utilizar en éstas mismas.

Figura 1.3 ***Aplicación Multicapa***



La Capa del Cliente

Consiste en aplicaciones cliente que acceden al servidor Java EE y que están por lo general ubicadas en equipos distintos al del servidor. El cliente realiza peticiones al servidor; éste procesa las solicitudes y regresa una respuesta al cliente. Hay muchas aplicaciones que pueden ser clientes Java EE y no siempre son aplicaciones Java (la mayoría de las veces no lo son). Un cliente Java EE puede ser un navegador, una aplicación stand-alone (Java o no Java), inclusive algún otro servidor.

La Capa Web o Capa de Presentación

Consiste en una serie de componentes que toman a su cargo la interacción entre los clientes y la capa de negocio. Sus principales tareas son las siguientes:

- La generación de contenido de forma dinámica en diferentes formatos para el cliente.
- La recepción de la información de entrada que ingresa el usuario a través de la interfaz que de proporciona el cliente y el retorno de los resultados apropiados que proporcionan los componentes de la capa de negocio.
- El control del flujo de pantallas o páginas del cliente.
- El mantenimiento del estado de los datos durante la sesión del usuario.
- El almacenamiento temporal de algunos datos en unos objetos llamados *Java Beans*.

Tecnologías Java EE utilizadas en la capa Web

La siguiente tabla es una lista de tecnologías J2EE utilizadas en la capa web por aplicaciones Java.

Tabla 1.3.1 ***Tecnologías Java EE para la capa Web***

Tecnología	Propósito
Servlets	Clases programadas en Java que procesan dinámicamente peticiones; construyen respuestas generalmente en forma de páginas HTML.
JavaServer Faces	Un componente para aplicaciones Web que permite incluir elementos de interfaz de usuario (User Interface UI) tales como campos de texto y botones en una página; convierte o valida datos provenientes de la interfaz de usuario, mantiene el estado de dichos datos y guarda éstos mismos en componentes que se manejan desde el lado del servidor.
JavaServer Faces Facelets	Las aplicaciones Facelets son un tipo de aplicaciones JavaServer Faces, pero las primeras utilizan páginas XHTML en lugar de páginas JSP.
Expression Language	Un set de etiquetas estándares (semejantes a las utilizadas en HTML) utilizados en las páginas JSP y Facelets para hacer referencia a componentes Java EE.
JavaServer Pages (JSP)	Son documentos de texto que se compilan en Servlets y definen de forma dinámica cómo añadir el contenido a las páginas estáticas, como las páginas HTML, por ejemplo.
JavaServer Pages Standard Tag Library	Una librería de etiquetas que encapsulan la funcionalidad principal que es común a las páginas JSP.
JavaBeans	Son objetos Java que actúan temporalmente como almacenes transitorios de los datos que se ingresan a través de las páginas de una aplicación.

La Capa de Negocio

Esta capa consiste en una serie de componentes que proveen de lógica de negocio a una aplicación. La lógica de negocio es código de programación que proporciona funcionalidad a un dominio particular como la industria financiera o un sitio de comercio electrónico. En una aplicación empresarial propiamente diseñada, la funcionalidad central existe en los componentes de la capa de negocio.

Tecnologías Java EE utilizadas en la Capa de Negocio

Algunas de las siguientes tecnologías Java EE son empleadas en la capa de negocio dentro de las aplicaciones empresariales Java EE:

- ***Enterprise JavaBeans***

  Los Enterprise JavaBeans son una tecnología que se ejecuta en el servidor y se emplea para desarrollar e instalar componentes con lógica de negocio dentro de una aplicación empresarial.

  Los componentes Enterprise JavaBeans, también conocidos como enterprise beans, es una tecnología segura, escalable, transaccional y multiusuario. Existen tres tipos de Enterprise beans: beans de sesión (session beans), beans de entidad (entity beans) y beans dirigidos a mensajería (message-driven beans).

  Los beans de sesión y de entidad cuentan con dos tipos de interfaz: la interfaz de componente (component interface), que define los métodos con la lógica de negocio del bean y la interfaz de hogar (home interface), la cual especifica los métodos para crear, encontrar, remover y acceder metadatos para el bean. Los beans dirigidos a mensajería no tienen ninguna de las dos interfaces mencionadas.

  Las interfaces de hogar y de componente pueden ser locales o remotas. Las interfaces remotas son interfaces de tipo *RMI* (Remote Method Invocation, que es un mecanismo ofrecido por Java para invocar un método de manera remota) que permiten al cliente de un bean ser independientes de la ubicación del bean. Sin importar si el cliente de un bean que implementa una interfaz remota está ubicado en la misma máquina virtual o en una distinta, el cliente utiliza la misma API para acceder a los métodos del bean. Los argumentos y los resultados se pasan por valor entre el cliente y el enterprise bean remoto.

  Un cliente de un enterprise bean que implementa una interfaz local debe encontrarse en la misma máquina virtual que el bean. Porque los objetos que son argumentos así como los resultados que regresan dicho bean se pasan por referencia entre ellos, no existe una sobrecarga de trabajo en la serialización de datos que viajan por la red

- ***El API Java Persistence para Entidades***

  Es una librería que simplifica el modelo de persistencia de objetos estandarizando el mapeo. Este API maneja la forma en que los datos relacionales son mapeados a objetos Java (conocidos como “entidades persistentes”); también se encarga de la forma en que dichos objetos son almacenados en una base de datos relacional para ser accedidos posteriormente y así como de la forma de preservar el estado de una entidad aun cuando la aplicación que la utiliza ya ha terminado.

La Capa de Sistemas de Información Empresarial (Enterprise Information System)

La Capa de Sistema de Información Empresarial consiste en servidores de base de datos, sistemas de planeación de recursos empresariales y de otro tipo de base de datos como las que manejan los mainframes. Todos estos recursos están ubicados típicamente en una máquina diferente a la del servidor Java EE y se accede a ellos a través de la capa de negocio.

Tecnologías Java EE utilizadas en la Capa de Sistemas de Información Empresarial

La siguiente lista de tecnologías Java EE forman parte la Capa de Sistemas de Información Empresarial para las aplicaciones empresariales Java EE:

- ***El API Java Persistence***

- ***El API Java Database Connectivity (JDBC)***

  Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el lenguaje SQL del modelo de base de datos que se emplee.

  El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar.

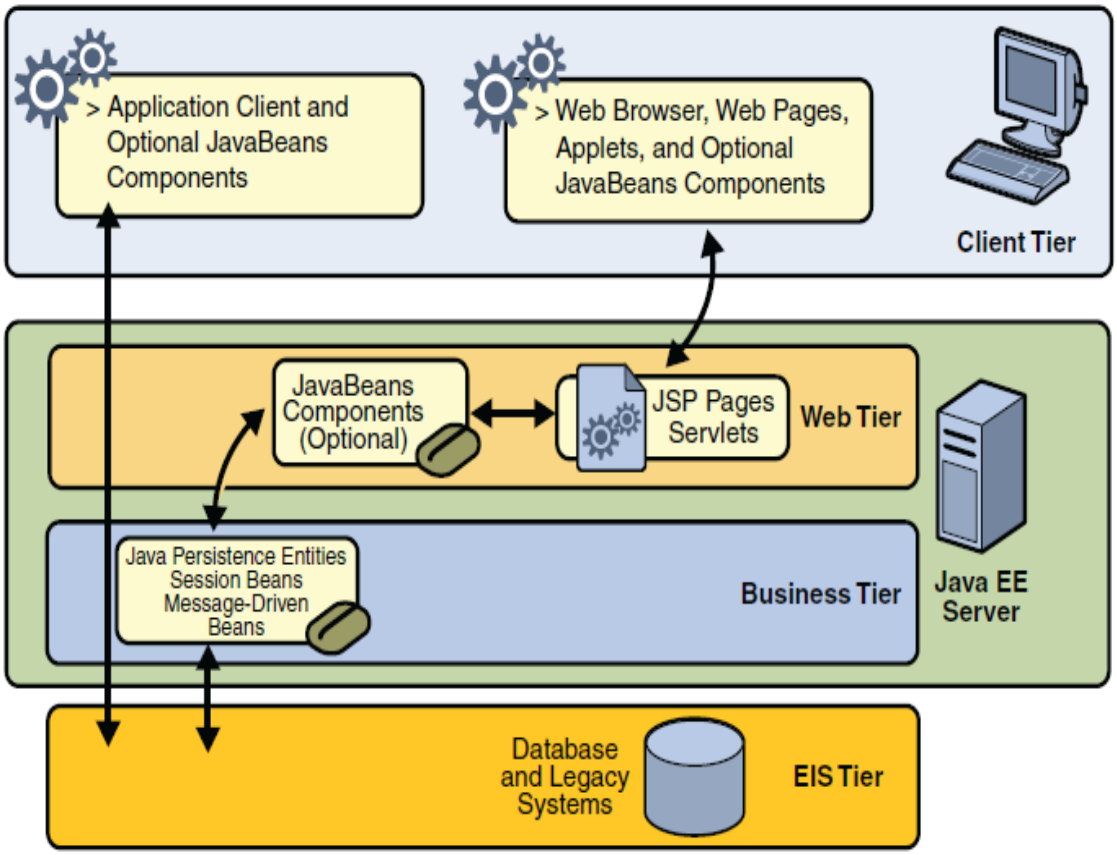
  Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

- ***El API Java Transaction (JTA o API para transacciones Java)***

  Hay ocasiones cuando no se quiere que una sentencia tome efecto a menos que otra sentencia se complete. Por ejemplo, cuando el propietario de una cafetería actualice la cantidad de cafés vendidos por semana, también querrá actualizar la cantidad de dinero obtenido por dichas ventas. Sin embargo, no querrá actualizar una cantidad sin actualizar la otra, de otra manera, la información se volvería inconsistente. La única forma de asegurar que ambas cantidades se actualicen es a través de una transacción. Una transacción es un set de una o más sentencias que se ejecutan en conjunto como una unidad, así que o todas las sentencias se ejecutan o ninguna lo hace.

  Básicamente, el propósito de la API JTA es definir las interfaces Java locales requeridas para la gestión de transacciones que hay que administrar en un ambiente de cómputo distribuido para aplicaciones empresariales.

Figura 1.3.2 Las Capas Cliente, de Negocio y de Sistemas de Información Empresarial



1.3.3 Servidores Java EE [3]

Un servidor Java EE es un servidor de aplicaciones que implementa las API's de la plataforma Java EE. Se les llama servidores de aplicaciones porque permiten al desarrollador proporcionar información de la aplicación a los clientes, así como los servidores Web proporcionan páginas Web a los navegadores.

Los servidores Java EE albergan diferentes tipos de componentes que corresponden a las diversas capas que conforman una aplicación multicapas. Un servidor Java EE proporciona servicios a éstos componentes en forma de contenedores.

Contenedores Java EE

Los contenedores Java EE son la interfaz entre el componente y la funcionalidad de bajo nivel que proporciona la plataforma Java EE para soportar dicho componente. La funcionalidad del componente está definida por la plataforma Java EE y es diferente para cada tipo de componente. No obstante, un servidor Java EE permite a los diferentes tipos de componente trabajar conjuntamente para proporcionar la funcionalidad requerida en una aplicación empresarial.

[3] Understanding Java Platform, Enterprise Edition. Ver bibliografía al final del documento.

El Contenedor Web

El contenedor Web es la interfaz entre los componentes Web y el Servidor Web. Un componente Web puede ser un Servlet, una página JSP o una página JavaServer Faces Facelets. El contenedor administra el ciclo de vida de los componentes, envía las solicitudes a los respectivos componentes de la aplicación y proporciona interfaces para poner en contexto los datos, como la información acerca de la actual situación.

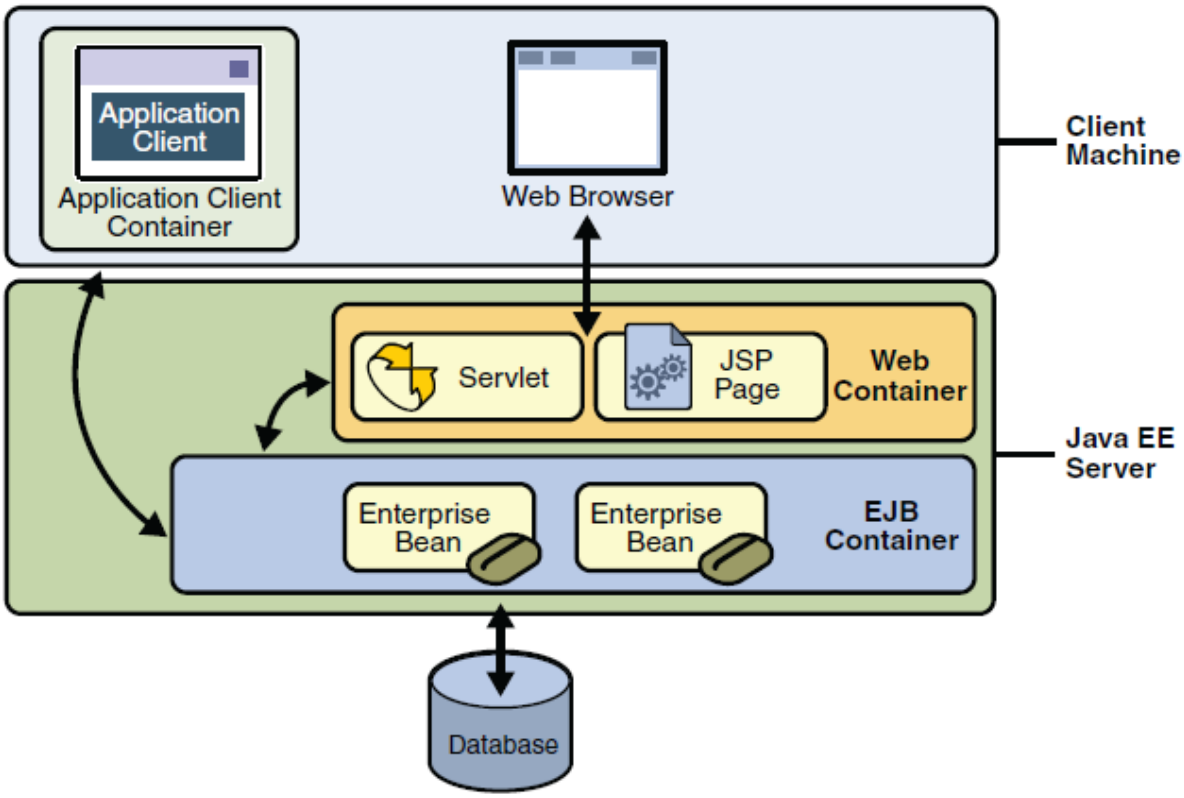
El Contenedor de la Aplicación Cliente

Este contenedor es la interfaz entre las aplicaciones cliente Java EE (aplicaciones Java SE especiales que utilizan componentes del servidor Java EE) y el propio servidor Java EE. Dicho contenedor se ejecuta en la máquina cliente y es la puerta de enlace (*gateway*) entre la aplicación cliente y los componentes del servidor Java EE que ésta utiliza.

El Contenedor EJB

El contenedor EJB es una interfaz entre los componentes Enterprise Java Beans, los cuales proporcionan lógica de negocio en una aplicación Java EE, y el servidor Java EE. El contenedor EJB se ejecuta en el servidor Java EE y administra la ejecución de una aplicación de Enterprise beans.

Figura 1.3.3 Servidores y Contenedores J2EE



2. MEJORES PRÁCTICAS DE JAVA Y J2EE

2.1 Las Mejores Prácticas de Java

2.1.1 Convenciones de codificación en Java

Una buena práctica en la programación de aplicaciones, ya sea en Java o en cualquier otro lenguaje de programación, es escribir código fuente legible, codificado bajo estándares que sean universales para cualquier programador.

Si bien es cierto que la razón primaria por la que se genera código fuente es para que un traductor lo lea y genere el código ejecutable, lo cierto es que debido a la depuración de errores (*debugging*), al mantenimiento y a las refactorizaciones (modificaciones del código fuente sin cambiar su comportamiento, lo que se conoce informalmente como limpiar el código), el código fuente se lee mucho más por humanos que por máquinas. Lo que es más: se lee muchas más veces de las que se lo escribe.

De manera general, las convenciones de codificación de software son importantes para los programadores por las siguientes razones:

- 80% del costo del tiempo de vida de una pieza de software se dedica al mantenimiento.
- Difícilmente el autor original de cualquier programa le da mantenimiento en toda la vida del mismo.
- Las convenciones de codificación mejoran la legibilidad de cualquier programa, permitiendo así a los programadores entender rápidamente código fuente que inicialmente ellos no generaron.
- Si se incluye el código fuente como parte del producto final a entregar al cliente, es necesario asegurarse de que el código está bien estructurado, bien documentado y bien empaquetado.

Java establece algunas convenciones de codificación comunes a cualquier lenguaje de programación (comentarios, indentación, etc.) y otras específicas al mismo. A continuación revisaremos los puntos más importantes de ambos tipos de convenciones de los que me ha servido para generar código en mi experiencia laboral.

Convenciones de Codificación Comunes

- ***Indentación* de las líneas de código**, práctica muy útil para definir los bloques de programación.
- ***Comentarios***, pequeñas descripciones de lo que hace una o varias líneas de código fuente cuando éstas por si solas no soy explicitas en la lógica de negocio que implementan.

Convenciones de Codificación para Java[4]

- ***Nombre de archivos***

  Los archivos de código fuente que definen las clases y las interfaces deben coincidir con el nombre de clases e interfaces públicas que dichos archivos definen y terminar con la extensión *.class*.

- ***Longitud de línea***

  Anteriormente SUN (ahora Oracle) definió la longitud de líneas de código y de comentarios a no más de 80 caracteres, esto porque algunas terminales y herramientas (que varían según el sistema operativo) llegan a tener problemas para manipularlas correctamente. Cabe mencionar que las líneas empleadas para ilustrar un ejemplo en la sección de comentarios, no deben sobrepasar los 70 caracteres.

- ***Nombrado y declaración de variables***

  Los nombres de las variables deben ser cortos pero siempre significativos, describiendo en lo posible el propósito de la variable. La notación empleada es la de camello (totalRegistros, por ejemplo).

  Cuando se habla de variables temporales, se suelen utilizar los nombres *i*, *j*, *k*, *m*, y *n* para variables de tipo entero y c, d y e para variables de tipo carácter.

  Las variables se declaran una a la vez y por línea de código, ejemplo:

  int contador;
  float monedaNacional;

- ***Nombrado de constantes***

  El nombre de las constantes de clase debe escribirse con mayúsculas con palabras separadas por guiones bajos ("_").

- ***Declaración y nombrado de paquetes, clases e interfaces***

  <u>*Declaración*</u>: en un archivo *.java,* la primera línea que se declara es la del paquete; enseguida se declararan todas las clases con su nombre calificado o paquetes que se van a importar para tener acceso a los objetos definidos en dichos paquetes. Por último

  <u>*Nombrado*</u>: Para los paquetes tenemos que el prefijo del nombre de un paquete único se escribe siempre en letras minúsculas en código ASCII y deben corresponder al dominio de más alto nivel (actualmente com, edu,

[4] Code Conventions for the JavaTM Programming Language. Ver bibliografía al final del documento.

gov, mil, net, org) o también a alguno de los códigos que identifican a los países en inglés según las especificaciones ISO Standard 3166, 1981

Los componentes subsecuentes del nombre de un paquete varían de acuerdo a la convención interna propia de nombrado de cada organización. Dichas convenciones pueden especificar que el nombre de cierto directorio de componentes sean división, departamento, proyecto, maquina o nombre de sesión

Ejemplos

- **sun.eng**
- **com.apple.quicktime.v2**
- **edu.cmu.cs.bovik.cheese**

Para los nombres de clases deben ser sustantivos, con notación mixta (primera letra de cada palabra interna del nombre en mayúscula y el resto en minúscula, ejemplo TestAccount). Los nombres elegidos deben ser lo más simples y descriptivos posibles; deben evitarse las abreviaciones o los acrónimos a menos que éstos sean mucho más conocidos que en su forma completa, como URL o HTML. Lo anterior aplica también a las interfaces.

Seguir estas convenciones de codificación no es complicado; existen herramientas como Jalopy y CheckStyle para Java que ayudan a entregar componentes codificados bajo dichas convenciones de programación y sirven para revisar sintaxis, longitud de líneas, nombrado de clases, métodos, variables, uso de variables, etc.

2.1.2 Conceptos clave en el diseño orientado a objetos

La idea de que un buen diseño debe estar basado en una alta cohesión y un bajo acoplamiento viene de los años 1980. Fue uno de los primeros principios proclamados del diseño estructurado que se retomo posteriormente en la programación orientada a objetos.

Junto con la cohesión y el acoplamiento, la modularidad es otro concepto que establecen las bases para diseñar y desarrollar aplicaciones escalables y de fácil mantenimiento.

Modularidad

El aporte más importante que hizo el diseño estructurado fue la idea de que, para resolver un problema complejo de desarrollo de software, conviene separarlo en partes más pequeñas, que se puedan diseñar, desarrollar, probar y modificar, de manera sencilla y lo más

independientemente posible del resto de la aplicación. Esas partes, cuando se quiere usar un nombre genérico, habitualmente se denominan módulos[5].

Se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes, como se menciono anteriormente.

En el diseño orientado a objetos, a diferencia del diseño estructurado, la modularización se da a nivel de clases, por lo tanto, en el análisis y diseño se "modulariza" primero el problema (en el análisis) y luego, partiendo de esas clases conceptuales, del dominio del problema, se modulariza la solución (diseño).

Por supuesto, la orientación a objetos también tiene módulos funcionales, que serían los métodos u operaciones de las clases, pero estos tienen una importancia menor respecto del módulo por excelencia, que es la clase.

Finalmente, en el diseño orientado a objetos, suele aparecer otro tipo de módulo más, el paquete, de escasa relevancia semántica, pero importante para agrupar clases en el diseño de aplicaciones medianas.

Alta cohesión[6]

El concepto de cohesión hace referencia a la forma en que se agrupan unidades de software (piezas de software que realizan algún cometido como lo son los módulos, las funciones, las bibliotecas etc.) en una unidad mayor. Por ejemplo, la forma en que se agrupan funciones en una biblioteca de funciones o la forma en que se agrupan métodos en una clase o la forma en que se agrupan clases en un paquete.

En el diseño orientado a objetos hay tres tipos de módulos: clases, métodos y paquetes. Con los métodos, se suele hacer una prueba para ver si son cohesivos analizando si éstos se pueden describir con una oración simple, con un solo verbo activo. Si hay más de un verbo activo en la descripción del método, se debe analizar su partición en más de un método y volver a hacer la prueba.

Respecto a las clases, éstas tendrán alta cohesión cuando se refieran a una única entidad. Se puede garantizar una fuerte cohesión disminuyendo al mínimo las responsabilidades de una clase: si una clase tiene muchas responsabilidades probablemente haya que dividirla en dos o más.

La prueba o test a aplicar para las clases sería ver si se puede describir a la clase con una oración simple que tenga un único sustantivo en el sujeto. Si la clase estuviera representando alguna operación (por la aplicación de algún patrón de diseño, por ejemplo),

[5] Ingeniería de software. Ver bibliografía al final del documento.

[6] Diseño de Software Orientado a Objetos. Ver bibliografía al final del documento.

también habría que tratar de "sustantivarla" y aplicarle la prueba para ver si es cohesiva. Una clase con alta cohesión suele cumplir el principio de única responsabilidad.

En los paquetes no es usual analizar cohesión. Sin embargo, nada impide aplicarle los mismos tests que a las clases; lo crucial en los paquetes es el acoplamiento.

Bajo Acoplamiento

El acoplamiento es el grado de interdependencia entre las unidades de software de un sistema informático.

El acoplamiento define lo dependiente que son las unidades de software entre sí, es decir, el grado en que una unidad puede funcionar sin recurrir a otras. Por ejemplo, dos funciones son absolutamente independientes entre sí (es decir, tienen el nivel más bajo de acoplamiento) cuando una puede hacer su trabajo por completo sin recurrir a la otra. En este caso, se dice que ambas están desacopladas.

Para obtener un buen diseño y una buena programación en los componentes que forman las aplicaciones informáticas, debe existir un bajo acoplamiento entre las unidades de software que lo componen.

El bajo acoplamiento permite:

- Mejorar y facilitar el mantenimiento de las unidades de software.
- Aumentar la reutilización de las unidades de software.
- Evitar el efecto onda, ya que un defecto en una unidad puede propagarse a otras, haciendo incluso más difícil detector la ubicación puntual del problema.
- Minimizar el riesgo de tener que cambiar múltiples unidades de software cuando se va a modificar alguna de éstas.

Es imposible lograr un desacoplamiento total de las unidades de software, pero mientras menos acopladas se encuentren, menor será el impacto que deban sufrir cuando existan modificaciones en las mismas.

2.2 Las Mejores Prácticas de J2EE (Java EE)

Los Patrones de Diseño o *Design Patterns* son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

Hay que entender primero que un contexto es el entorno, situación, o condiciones interrelacionadas dentro de las cuales existe algo. Segundo, un problema es una cuestión insatisfecha, algo que se necesita investigar y resolver. Un problema se puede especificar mediante un conjunto de causas y efectos. Normalmente un problema está restringido al

contexto en el que ocurre. Y finalmente, la solución se refiere a la respuesta al problema dentro de un contexto que ayuda a resolver las dificultades.

La clave para la reutilización de componentes en informática es anticiparse a los nuevos requisitos y cambios, de modo que los sistemas evolucionen de forma adecuada. Cada patrón de diseño permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos facilitando así la reusabilidad, extensibilidad y mantenimiento en un sistema.

Beneficios de los patrones de diseño:

- Contribuyen a reutilizar el diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, ya que ésta nos provee de numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento de un componente, mejora la seguridad, eficiencia y consistencia de nuestros diseños, y nos proporciona un considerable ahorro en la inversión de tiempo y trabajo.

- Mejoran (aumentan, elevan) la flexibilidad, la modularidad y la extensibilidad, factores internos que como ya vimos anteriormente, íntimamente relacionados con la calidad percibida por el usuario.

- Incrementan nuestro vocabulario de diseño, ayudándonos a diseñar desde un mayor nivel de abstracción.

De manera general, los patrones de diseño se clasifican como se muestra a continuación:

- *Patrones Creacionales*: Sirven para la inicialización y configuración de objetos.

- *Patrones Estructurales*: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.

- *Patrones de Comportamiento*: Más que describir objetos o clases, describen la comunicación entre ellos.

2.2.1 Patrones de diseño empleados en mi experiencia laboral

Singleton

El patrón de diseño *Singleton* es un patrón creacional y garantiza que una clase sólo tenga una sola instancia de ella misma, proporcionando un punto de acceso global a ella. Éste patrón es útil para limitar a solo una el número máximo de instancias de una clase dada. En este caso, si más de un objeto necesita utilizar una instancia de la clase Singleton, esos objetos compartirán la misma instancia de dicha clase. En un uso más avanzado, este patrón puede ser utilizado también para administrar n instancias de una cierta clase.

Facade

El *Facade* o la Fachada, es un patrón estructural que simplifica el acceso a un sistema o a un conjunto de componentes proporcionando una única clase (interfaz) que todos los clientes utilizarán para comunicarse con dichos componentes. Dicho de otro modo, el objetivo del patrón fachada es tratar de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace las veces de pantalla o fachada. Aquí, la idea principal es la de ocultar en todo lo posible la complejidad de un sistema (o del conjunto de clases o componentes que lo forman), de forma que solo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada.

Una ventaja más de usar una clase fachada para comunicar dos partes o componentes, es la de aislar los posibles cambios que se puedan producir en alguna de las mismas. Con el patrón Facade, los clientes no necesitan conocer las clases ni la implementación que hay tras la clase Facade y se puede modificar las clases "ocultas" sin necesidad de cambiar la lógica de los clientes, sólo hay que realizar las modificaciones necesarias en la fachada para que el acceso siga siendo transparente. Por ejemplo, si se modifica la forma de almacenamiento de información de una de las partes, la otra, que está a cargo de hacer la presentación de dichos datos, no tiene porque enterarse de éste cambio, y viceversa

Command

El patrón de comportamiento *Command* o *Comando*, encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones. Revisemos más a detalle en qué consiste éste patrón de diseño.

La cadena de responsabilidad envía solicitudes a lo largo de una cadena de clases, pero el patrón Command envía una solicitud solo a un módulo específico.

El Command encierra una solicitud para una sola acción específica dentro de un objeto y da al cliente la habilidad de hacer peticiones sin saber nada acerca de la acción que va a ser ejecutada y permite modificar la implementación de la acción sin afectar al cliente en forma alguna.

Uno de los propósitos del patrón Command es mantener los objetos de un programa o de una interfaz de usuario completamente separados de las acciones que cada uno de ellos inician. En otras palabras, esos objetos de los programas deben estar completamente separados entre sí y no deben saber cómo los otros objetos trabajan. La interfaz de usuario recibe un comando y a su vez lo delega a un objeto command para que éste ejecute las tareas para las cuales fue creado. La interfaz de usuario no sabe (y no tiene porque saberlo) que tarea va a ejecutarse.

El objeto Comnmand puede también ser utilizado cuando se necesita que un programa no ejecute dicho comando hasta el momento en el que los recursos estén disponibles. En este

caso, los comandos se encolan para ejecutarse posteriormente. Finalmente, se pueden también utilizar los objetos command para deshacer peticiones.

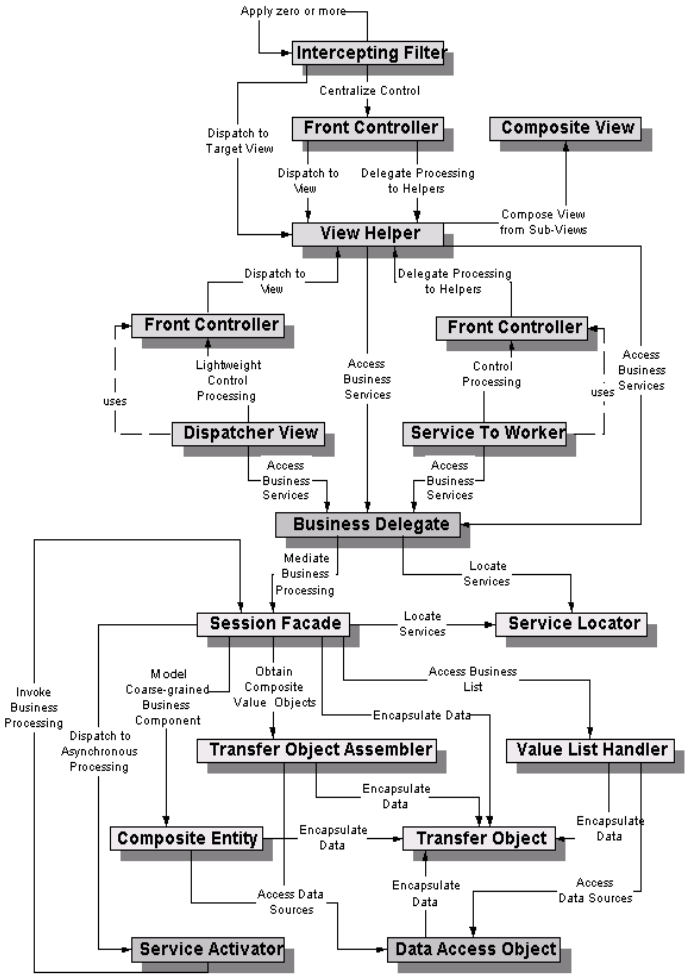
Patrones de Diseño J2EE

Como ya se menciono anteriormente, los patrones de diseño son soluciones ya probadas y aplicables a un problema que se presenta una y otra vez en el desarrollo de distintas aplicaciones y en distintos contextos.

Los patrones J2EE describen típicos problemas encontrado por desarrolladores de aplicaciones empresariales y proveen soluciones para estos problemas. En esencia, estos patrones contienen las mejores soluciones para ayudar a los desarrolladores a diseñar y construir aplicaciones para la plataforma J2EE.

Aunque estos patrones son representados desde un nivel lógico de abstracción, todos contienen en sus descripciones originales en la Web de Sun[7], varias estrategias que ofrecen una gran cantidad de detalles para su implementación.

Figura 2.1 ***Patrones de Diseño J2EE***



[7] Core J2EE Patterns: Patterns index page. Ver bibliografía al final del documento.

El patrón de diseño J2EE Objeto de Acceso a Datos

En el capítulo anterior, se menciono que las aplicaciones J2EE cuentan con una capa de Sistemas de Información Empresarial encargada de acceder a los datos útiles a la aplicación. La capa de Sistemas de Información Empresarial contiene uno o más servidores de base de datos y recibe las solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

El Objeto de Acceso a Datos (Data Access Object o DAO) es un patrón J2EE que se usa para abstraer y encapsular todos los accesos a una fuente de datos. Una fuente de datos puede ser una base de datos relacionales, una base de datos orientadas a objetos, archivos de texto plano, etc. El DAO es el encargado de administrar la conexión con la fuente de datos para obtener o almacenar información.

Este patrón esconde por completo los detalles de la implementación de la fuente de datos ya que la interfaz expuesta por el DAO a los clientes no cambia cuando la implementación subyacente de la fuente de datos se modifica, permitiendo así al DAO adaptarse a los diferentes esquemas de almacenamiento sin afectar a los componentes de negocio o a los clientes. En esencia, el DAO actúa como un adaptador entre un componente y la fuente de datos, logrando así que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento.

2.2.2 Uso de archivos de configuración en las aplicaciones Java y J2EE

El desarrollo de aplicaciones siempre involucra una serie de constates y valores por defecto, que en conjunto conforman la configuración de una aplicación. La configuración de una aplicación debe ser lo más dinámica posible, ya que el ambiente en el que va a instalarse puede cambiar (direcciones ip de servidores web, servidores de base de datos, servidores FTP, usuarios, contraseñas, directorios de almacenamiento de archivos, etc.), por lo que dejar dicha configuración dentro del código fuente es un grave error. Cada modificación implicaría una recompilación del código, actividad que no es *mantenible* y puede ser un proceso largo y tedioso de repetir ya dentro de un ambiente productivo.

Es por ello que es imprescindible utilizar mecanismos que permitan modificar la configuración de las aplicaciones de manera cómoda y efectiva. Esto se logra utilizando archivos de propiedades o archivos xml. En ambos archivos se almacena información similar, por lo que no es de extrañar que en la distribución de APIS de software libre encontremos ambos tipos de archivos de configuración, como *Log4j*, una API ampliamente utilizada para el manejo del log de un programa o una aplicación.

Archivos de propiedades *.properties*

El API de Java contiene la clase *Properties* que nos permite crear archivos de configuración y manejarlos con facilidad.

Un archivo *.properties* nos permite almacenar información en el formato clave = valor (o valores).

Figura 2.2 ***Ejemplo de un archivo de propiedades***

```
##########################################
# Archivo de configuración
##########################################
# Parametro 1
depuracion = true
#Otros parametros
alto = 480
ancho = 750
color = negro
```

Archivos de propiedades *.xml*

Los archivos .xml son mucho más muy comunes en las aplicaciones J2EE y cumplen básicamente la misma funcionalidad que los archivos de configuración .properties, solo que su uso se ha popularizado más entre la comunidad que programa aplicaciones web.

Por ejemplo, las aplicaciones web J2EE concentran su configuración principal en un archivo denominado web.xml; en el, se definen páginas de inicio, de error, url's de componentes, propiedades del sistema, recursos de base de datos, de ejb's, entre otros componentes, centralizando así la configuración de la aplicación, evitando la recompilación de código fuente y concentrando en un solo lugar la configuración de dicha aplicación.

Si hay que modificar dicha configuración, basta con dar de baja el servidor con el contenedor Web, hacer la modificación correspondiente en el archivo web.xml, guardar los cambios e iniciar nuevamente el contenedor Web que alberga la aplicación Web en cuestión.

Figura 2.3 ***Ejemplo de archivo de configuración xml***

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

      <display-name>ServiciosEntidad</display-name>

      <context-param>
            <param-name>log4jConfigLocation</param-name>
            <param-value>/WEB-INF/classes/log4j.xml</param-value>
      </context-param>

      <context-param>
            <param-name>jBossConfiguration</param-name>
            <param-value>/WEB-INF/classes/jboss.xml</param-value>
      </context-param>

      <context-param>
            <param-name>sqlConfiguration</param-name>
            <param-value>/WEB-INF/classes/sqlMapping.xml</param-value>
      </context-param>
      <servlet>
            <display-name>
            Apache-Axis Servlet</display-name>
```

```
            <servlet-name>AxisServlet</servlet-name>
            <servlet-class>
            org.apache.axis2.transport.http.AxisServlet</servlet-class>
      </servlet>

      <servlet-mapping>
            <servlet-name>AxisServlet</servlet-name>
            <url-pattern>/servlet/AxisServlet</url-pattern>
      </servlet-mapping>

      <welcome-file-list>
            <welcome-file>index.html</welcome-file>
            <welcome-file>index.htm</welcome-file>
            <welcome-file>index.jsp</welcome-file>
            <welcome-file>default.html</welcome-file>
            <welcome-file>default.htm</welcome-file>
            <welcome-file>default.jsp</welcome-file>
      </welcome-file-list>

      <error-page>
            <error-code>404</error-code>
            <location>/axis2-web/Error/error404.jsp</location>
      </error-page>

      <resource-ref id="ResourceRef_1231978742879">
            <description>
            Referencia al DS SICOCA</description>
            <res-ref-name>jdbc/informixDS_SICOCA</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Application</res-auth>
            <res-sharing-scope>Shareable</res-sharing-scope>
      </resource-ref>

      <resource-ref id="ResourceRef_1231778742899">
            <description>
            Referencia al DS SAAI</description>
            <res-ref-name>jdbc/informixDS_SAAI</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Application</res-auth>
            <res-sharing-scope>Shareable</res-sharing-scope>
      </resource-ref>

      <resource-ref id="ResourceRef_1237978742877">
            <description>
            Referencia al DS SCAAA</description>
            <res-ref-name>jdbc/informixDS_SCAAA</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Application</res-auth>
            <res-sharing-scope>Shareable</res-sharing-scope>
      </resource-ref>
</web-app>
```

3. DESAFÍOS ENCONTRADOS EN EL DESARROLLO DE APLICACIONES PARA EL SECTOR PRODUCTIVO

La participación que he tenido en diferentes proyectos dentro del sector público y privado, tanto como programador analista como líder de un equipo de trabajo, me ha enfrentado en repetidas ocasiones a diferentes retos o desafíos. Ya sea que éstos desafíos sean de documentación, técnicos, tecnológicos o de administración, todos me han ayudado a crecer en el ámbito profesional como programador, como líder de equipos de trabajos y en general, como consultor de sistemas.

Cabe mencionar que el concepto de usuario, al menos en lo que yo he podido constatar estos años de experiencia laboral profesional, no siempre se refiere a una persona físicamente tangible que utiliza directamente un sistema, el concepto de usuario puede también utilizarse para designar el departamento de informática o de sistemas de la empresa o institución para la cual se está laborando.

3.1 Desafíos como programador

3.1.1 Poca o nula documentación en las aplicaciones legadas

Una de las actividades más comunes como programador es el mantenimiento, e incluso la modernización, de aplicaciones legadas. Es decir, aplicaciones productivas de varios años atrás a las que hay que dar mantenimiento o en su defecto, hay que analizarlas nuevamente para optimizarlas y si el usuario de las mismas lo solicita, migrarlas de plataforma o de tecnología.

Cualquiera que sea el caso, la documentación de dichas aplicaciones es muy importante para el mantenimiento y/o la modernización de estos sistemas y sin embargo, es una situación recurrente la ausencia de documentación, tanto de los procesos de negocio como de la programación empleada.

Es sabido que los programadores que desarrollan en un inicio los sistemas empresariales difícilmente continuarán dándole ellos mismos el mantenimiento que éstas requieren, la rotación del personal de informática es muy común en el medio y es entonces cuando la documentación juega un papel primordial para “entrenar” al nuevo personal que se hará cargo de las aplicaciones, ya sea para terminar de construirlas o para actualizarlas.

La documentación de los procesos de negocio que componen a las aplicaciones del cliente permite conocer el negocio empresarial que lleva a cabo un sistema dentro de la institución de manera rápida y sencilla; hay ocasiones en que un usuario del sistema puede confirmar el funcionamiento de manera global o a detalle en ausencia de una documentación formal, pero no siempre es posible el contacto con el mismo (ya sea por cuestiones administrativas propias de la institución o porque simplemente ¡no existe dicho usuario! y el mantenimiento se va a realizar a un módulo que aún no termina de programarse pero el

desarrollador inicial no dejo la documentación requerida). Es entonces cuando se recurre a la documentación de procesos de negocio. Si ésta no existe, entonces se recurre al análisis del código fuente del sistema.

El código fuente bien documentado siempre es de gran ayuda para comprender en un corto tiempo lo que las clases o interfaces hacen dentro del sistema; la ausencia de comentarios en el código fuente, aunado al mal estilo de codificación del mismo (falta de indentación, mala definición de bloques de programación, uso nulo de convenciones de nombrado de variables, etc.), hace más difícil y tardado el análisis que hay que realizar para dar mantenimiento o modernizar una aplicación dada.

En ambos casos, existe una inversión de tiempo que a mi punto de vista es innecesaria ya que en su momento la o las persona(s) con el conocimiento técnico y del negocio debieron generar la documentación requerida. Más allá de proporcionar la documentación necesaria para que sea de ayuda a quienes se harán en un futuro cargo del sistema, es una práctica obligatoria para todas las aplicaciones que se desarrollan, sin importar empresa o institución.

Ninguna de las instituciones para las que he trabajado ha estado exenta de la falta de documentación, y he aprendido que es bien importante documentar tanto el código fuente que programo (utilizando las convenciones de codificación y agregando los comentarios pertinentes) como los procesos que llevan a cabo las aplicaciones que por completo yo he desarrollado, siempre tomando como base los lineamientos de documentación de la institución.

3.1.2 Legado de módulos o aplicaciones no escalables

Como ya mencione en el primer capítulo de este trabajo, las aplicaciones empresariales hoy en día deben ser escalables para incorporar siempre los nuevos requerimientos que surgen con el avance tecnológico así como para adaptarse el crecimiento mismo de las instituciones para las cuales dichos sistemas son creados.

Durante mi experiencia laboral, he tenido que hacerme cargo de componentes a los que hay que dar mantenimiento porque no cumplen ya con los requerimientos del negocio del usuario. En ocasiones, es necesario modificarlos de forma importante para que puedan cumplir con dichos requerimientos y en casos extremos, he tenido que programarlos desde cero, porque la forma en la que actualmente se encuentran, de ninguna manera permite añadir los nuevos requerimientos que se solicitan.

Es bastante común que no se desarrollen componentes escalables, es decir, componentes que puedan incorporar nueva funcionalidad de manera rápida y sencilla sin ocasionar un alto impacto en el resto de componentes con los cuales los primeros interactúan.

Lo anterior suele ocurrir porque no existe una cultura en el desarrollo de aplicaciones que se anticipe a los futuros cambios que sufren siempre los sistemas empresariales de información. Ya sea por falta de conocimiento y/o de experiencia, es muy común desarrollar

clases, interfaces, módulos o componentes con baja cohesión o un alto acoplamiento, repercutiendo negativamente en la escalabilidad de los mismos. Si hay que reprogramar, ello impacta en un esfuerzo adicional de trabajo que puso ser evitado si dichos componentes se hubiesen construido para ser escalables.

Han sido los años de experiencia laboral y la profundización en mis conocimientos técnicos los que me han enseñado que programar componentes escalables es la mejor forma de anticiparse a los cambios constantes de los sistemas de información, evitándome así tiempo, esfuerzo y dolores de cabeza cuando hay que hacer alguna modificación en los mismos.

3.1.3 Deficiencia de recursos en hardware y software para programar

La construcción de aplicaciones en Java y J2EE puede requerir de una cantidad importante de recursos (principalmente de memoria RAM), sobre todo si se va a construir software con la ayuda de los *IDE's* (*Integrated Design Environment* o Ambiente Integrado de Diseño), que son herramientas de desarrollo muy útiles para codificar, rastrear problemas y probar los componentes que se están desarrollando.

Los servidores de aplicaciones así como los clientes a bases de datos son también otro tipo de herramientas de desarrollo que comúnmente se emplean en la construcción de sistemas. Como es de imaginar, estas herramientas necesitan ejecutarse en equipos de cómputo con una mayor capacidad de procesamiento y de memoria RAM que el resto de las computadoras que se emplean para cuestiones escolares o administrativas.

Desafortunadamente, son las computadoras para actividades administrativas las que prevalecen en las áreas o departamentos de informática de las organizaciones para las que he trabajado. En ocasiones, es solo una cuestión administrativa de los departamentos de sistemas el poder ampliar los recursos de cómputo (vía solicitud), con el fin de obtener un ambiente de desarrollo adecuado (ya no ideal) para la construcción de software, pero en otras, es prácticamente imposible cambiar los equipos ya designados para la programación. Todo depende del presupuesto con el que cuente el departamento de informática en cuestión y hay que "ajustarse" a la situación.

Para el programador, lo anterior se ve reflejado en el tiempo que consumen todas las actividades de desarrollo que debe efectuar. Por ejemplo, si el tiempo para la programación de un componente cualquiera estaba estimado para que durara 4 horas, el no contar con un ambiente adecuado puede extenderlo hasta dos horas o más, ya que los procesos de compilación, instalación o rastreo de errores, así como las pruebas se vuelven muy lentos en dichos equipos de computo inadecuado, impactando la productividad tanto del desarrollador como la del proyecto. Y sin embargo, es una situación que se repite continuamente en el desarrollo de software.

Una forma de enfrentar este tipo de carencias de hardware es evitar la programación prueba y error, donde el programador no escribe código fuente de calidad y a conciencia y en

el que irá haciendo las correcciones pertinentes según los errores encontrados en tiempo de compilación o de ejecución. Esta práctica trae consigo una pérdida de tiempo considerable si tomamos en cuenta que la ejecución de los procesos de compilación e instalación y/o rastreo de problemas es lenta en un ambiente de hardware poco apropiado.

Es también frecuente el enfrentarse al uso de software para desarrollo de aplicaciones sin licencias, es decir, software ilegal o pirata (software alterado para que pueda ser utilizado sin pagar la licencia a sus desarrolladores originales).

Las desventajas de no contar con las licencias en éste tipo de software en los proyectos en los que he participado han sido principalmente el no contar con el soporte técnico ni los manuales por parte de los proveedores, tampoco se pueden descargar las actualizaciones de dichas herramientas, lo que impide en ocasiones contar con los parches que pueden arreglar algunos problemas que llegan a presentar dichas aplicaciones de desarrollo.

Esta falta de licencias también puede convertirse en un riesgo para la empresa o institución que incurre en este tipo de prácticas puesto que puede hacerse acreedora a multas de cantidades importantes.

Por fortuna, esta situación se presenta con mucha menos frecuencia que la carencia del hardware adecuado para desarrollar, ya que existen herramientas de desarrollo bajo los conceptos de *software libre* (tipo de software que respeta la libertad de los usuarios sobre el producto adquirido (gratuitamente o no) y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente) y de *shareware* (modalidad de distribución de software en la que el usuario puede evaluar de forma gratuita el producto, pero con limitaciones en el tiempo de uso, en algunas de las formas de uso o con restricciones en las capacidades finales).

Cuando ha estado en mis manos la elección de herramientas de trabajo, opto por utilizar software libre, que no implica problemas como los antes mencionados y aunque no siempre pone al alcance la documentación gratuita necesaria, si cuenta con un gran número de foros de desarrolladores dónde pueden encontrarse diversas soluciones a algunos problemas técnicos que éstas pudiesen presentar.

3.1.4 Acceso restringido a los recursos de hardware y software

La infraestructura informática de una empresa o una institución cualquiera se compone generalmente de servidores, de recursos de red y de ambientes de desarrollo y de pruebas. La infraestructura limitada que se ofrece para el desarrollo, las pruebas y el mantenimiento de las aplicaciones empresariales también debe considerarse como un desafío para que el programador cumpla satisfactoriamente con los objetivos que se le encomiendan cumplir. A continuación voy a describir algunos problemas derivados del acceso restringido de los recursos de hardware y software dentro de una institución.

Para asegurar que una aplicación en construcción o en mantenimiento funcionará correctamente al ser liberada a producción, existen ambientes de prueba cuya finalidad es recrear las condiciones que prevalecen en un ambiente de producción. Dicho de otra forma, en un ambiente de pruebas se implementan aspectos de seguridad, de bases de datos y de servidores de aplicación (entre otros), similares a los que existen en un ambiente productivo y en la medida de lo posible se trata de que un ambiente de pruebas sea una *imagen* de un ambiente de producción. Con ello, se busca probar la estabilidad, la confiabilidad y la robustez del sistema que se está programando.

En todas las instituciones donde me he desempeñado laboralmente existe un departamento o área de administración que controla los ambientes ya mencionados, con la finalidad de organizar y proteger tanto los sistemas que en éstos se encuentran como la información que dichas aplicaciones emplean. En consecuencia, la administración de estos ambientes cuenta con una serie de procedimientos para autorizar el acceso a los mismos.

Lo anterior es una práctica indiscutible y necesaria; el problema surge cuando el tiempo de respuesta para atender una solicitud de instalación de una aplicación en el ambiente de pruebas de prolonga (desde un par de horas hasta días) , impactando así considerablemente las pruebas a las que hay que someter los desarrollos para garantizar su correcto funcionamiento.

Para mitigar el impacto de retraso que pudiese ocasionar una solicitud de instalación en infraestructura, la única forma posible de mediar con este tipo de inconvenientes es planificar con anticipación las peticiones de instalación de aplicaciones en el ambiente de pruebas, ya que poco se puede hacer contra los procedimientos administrativos de cada institución.

Otra dificultad que se presenta en el área de sistemas es la falta de servidores, en número y en capacidad de cómputo, para albergar a todas las aplicaciones empresariales que se requieren instalar en los ambientes de desarrollo y de pruebas. En estos casos no es extraño encontrar servidores saturados en memoria RAM y en espacio en disco duro porque dicho servidor concentra diversas aplicaciones de diferentes áreas del departamento de sistemas. Al día de hoy, no he encontrado proyecto profesional alguno que cuente con servidores exclusivos, es un escenario muy costoso y el área de informática (o incluso la misma empresa o institución) casi nunca está dispuesta a invertir en este rubro.

Algo similar ocurre con el ancho de banda de una red institucional: suele ser un recurso limitado aún para el área que desarrolla sistemas, un área que por las actividades que desempeña debiese contar con una mayor transmisión de datos, tanto en el desarrollo de aplicaciones como para las pruebas que hay que realizar a las mismas. Nuevamente es una cuestión de presupuesto, ajena a nosotros los programadores, por lo que en la mayoría de las veces solo un poco de paciencia (y el uso correcto de la tecnología, con el fin de evitar cuellos de botella) es el mejor antídoto ante éste tipo de situaciones.

3.1.5 La curva de aprendizaje de nuevas tecnologías

El mundo de la informática y la computación está en constante cambio; nuevas tecnologías y metodologías para la programación se van desarrollando a una velocidad exponencial y en muy poco tiempo. La inversión de tiempo y esfuerzo en estudiar las nuevas tecnologías de desarrollo de sistemas es necesaria para todo programador que desee estar a la vanguardia (y en consecuencia, mantener un alto nivel de competitividad en comparación de otros programadores). El único desafío que yo he encontrado en la curvas de aprendizaje de una tecnología dada, es el tiempo con el que se cuenta para asimilar los nuevos conocimientos y ponerlos en práctica a la brevedad.

Una forma de lograr acortar la curva de aprendizaje es a través del empleo de ejemplos sencillos e ilustrativos que explican de manera general los aspectos más importantes de la tecnología que se está estudiando. Ello se puede lograr consultando ya sea a un compañero del equipo de trabajo que cuente de antemano con el conocimiento técnico del cual se carece o en su defecto, se pueden consultar los diversos foros de programación que hoy en día están disponibles en internet, en los que se abordan diversos temas relacionados con la tecnología de la información.

3.1.6 Uso de tecnologías inadecuadas para el desarrollo de sistemas

Dentro de uno de mis proyectos, tuve la oportunidad de tomar la decisión acerca de qué tecnologías utilizar para la construcción de las aplicaciones que en su momento se requirieron y definieron. Mi decisión estuvo determinada en base a los conocimientos técnicos que poseo, pero esa libertad no ha sido una constante en el resto de los proyectos en los que he trabajado. La mayoría del tiempo, la decisión de utilizar ciertas tecnologías sobre otras la toma el área de arquitectura de cada institución o empresa.

En éste momento preciso en que redacto éstas líneas, me encuentro trabajando en un proyecto que enfrenta problemas de estabilidad y de desempeño porque la interfaz del usuario se construyó con una tecnología que no es la adecuada para éste propósito.

Documentum[8], la tecnología en cuestión, está diseñada para administrar el ciclo de vida de un documento. El ciclo de vida se refiere a las distintas fases o etapas por las que va pasando el documento, desde su creación a su eliminación o selección para su custodia permanente.

Una de las limitantes que posee *Documentum* para emplearlo como interfaz de usuario es que no permite agregar comportamiento dinámico a sus formas, ya que no hay manera de hacer que se ejecuten archivos *javascript* ajenos a la herramienta. En consecuencia, las pantallas de la aplicación no son dinámicas ni flexibles como lo sería cualquier página Web

[8] Para saber más sobre EMC Documentum, ver bibliografía al final del documento.

(inclusive la más sencilla, hecha con HTML puro) y con el fin de obtener un mínimo de dinamismo en las pantallas, el equipo de desarrollo hemos tenido que emplear una serie de trucos de programación que nos implican más tiempo y una carga mayor de trabajo.

3.1.7 Carencias en la definición de requerimientos y en la planificación de actividades.

Aunque la definición de requerimientos o la planeación de actividades dentro de un equipo de trabajo no dependen directamente del programador (pues usualmente no son tareas que el desarrollador efectúe), si son dos factores que influyen en el tiempo de trabajo a invertir en la construcción de un sistema.

La elaboración de un diseño o producto conforme a requerimientos, da lugar a la posibilidad de fallar en el cumplimiento de lo solicitado por parte del usuario o del cliente. Cuando no se definen adecuadamente los requerimientos del sistema, la probabilidad de incumplimiento es aún mayor. Así, toda diferencia entre lo que se nos pidió y lo que estamos entregando como resultado de nuestro trabajo es considerado un fallo de calidad y se le suele llamar *inconformidad*. Si la inconformidad es muy grave y compromete la aceptación del producto final por parte del usuario, es necesario dedicar esfuerzos adicionales para lograr que el producto sea conforme a lo especificado en los requerimientos. Este esfuerzo adicional es lo que conocemos como *retrabajo*, el cual representa una inversión adicional de tiempo y esfuerzo de programación para el proyecto.

La presión que se ejerce sobre el equipo de desarrollo para corregir y terminar a la brevedad las modificaciones necesarias cuando se está en una inconformidad, puede provocar que el programador modifique los componentes de tal manera que sus clases pierdan cohesión o se tornen acopladas a otros componentes. Un riesgo del retrabajo puede provocar que se pierdan de vista las buenas prácticas de programación.

Si a lo anterior se suman una mala estimación y planeación en la definición de los tiempos para el desarrollo, pruebas, correcciones y entrega del sistema al usuario final, la carga de trabajo para el equipo de desarrollo (incluyendo a los líderes) se vuelve una tarea agotadora e incluso desagradable. Las jornadas laborales pueden extenderse horas, incluso fines de semana, por lo que el retrabajo nunca es bienvenido.

Levantar la mano ante el líder de proyecto cuando se detecta una indefinición en los requerimientos, cuando se encuentra un problema técnico o de tecnología que impida avanzar en el desarrollo o cuando no se pueden continuar las pruebas, por poner algunos ejemplos, es la mejor forma de prevenir que las deficiencias de análisis y administrativas repercutan negativamente en el tiempo de trabajo de los desarrolladores.

Toda mi experiencia como programador me ha enseñado que hacer las cosas bien desde un principio (o en su defecto, lo mejor posible), contemplando la carencia de recursos de hardware y software con anticipación, así como la planeación de pruebas con el área de servidores, ayudan enormemente para no comprometer en la medida de lo posible mi tiempo

laboral con mi vida personal. Finalmente, la experiencia adquirida a lo largo de los proyectos enseña qué hacer, cómo hacerlo, con que hacerlo y que no hacer, aventajando al programador experimentado sobre el programador junior, que apenas inicia su carrera profesional.

3.2 Desafíos como líder de un equipo de trabajo

En el área de la tecnología de la información, la implementación de nuevas técnicas, herramientas o sistemas se efectúa en la mayoría de los casos a través de proyectos. A lo largo de éste trabajo he utilizado el término proyecto sin definirlo hasta ahora. A continuación, haré una breve descripción de lo que es un proyecto.

Un proyecto es un conjunto estructurado de objetivos y acciones específicas que construyen metódica y progresivamente una realidad nueva.

Características:

- Generalmente es innovador (técnica y tecnológicamente hablando, en dimensión, etc.)
- Temporal: tiene un inicio y un fin determinado
- Adaptable a modificaciones frecuentes
- Orientado hacia un objetivo final

El líder de proyecto juega un papel fundamental para el éxito de un proyecto, ya que es el principal responsable de la experiencia global (buena o mala) que los actores del proyecto (analistas, programadores, *testers*, líderes de equipos de trabajo, etc.) viven durante su desarrollo. No ejecuta las tareas, pero las define; coordina, facilita, comunica, administra y anima el trabajo de su equipo; requiere de organización. El seguimiento sigiloso de un proyecto, marcará la diferencia en el resultado final.

Al inicio de cualquier proyecto, hay que descubrir las necesidades reales de cliente y no solo sus deseos. Una clara definición de las necesidades es la base de un proyecto exitoso. Es primordial para el buen término del proyecto corroborar que tanto el líder del proyecto como el cliente no tienen el mismo entendimiento del problema y la misma ilustración del resultado deseado.

En un proyecto de dimensiones considerables, las tareas a ejecutar se dividen en varios equipos de trabajo, cada uno de ellos con un líder responsable, respectivamente. Un líder de un equipo de trabajo reporta a su vez a sus líderes inmediatos, si los hubiera, o directamente al líder de proyecto, si es el caso.

Un líder de equipo de trabajo ejecuta tareas similares a las que lleva a cabo el líder de proyecto pero a una escala mucho muy pequeña, por ejemplo, el líder de trabajo del equipo de testers administra únicamente aspectos relacionados a las pruebas que aseguran la calidad del producto de software que se va a entregar al cliente o, el líder del equipo de trabajo de

desarrolladores solo se enfoca en administrar y dar seguimiento a todas las acciones involucradas en el desarrollo del software en cuestión.

Mi corta experiencia profesional como líder de un equipo de trabajo me ha enfrentado a diferentes situaciones relacionadas con cuestiones técnicas, tecnológicas y administrativas; se podría decir que a situaciones similares a las que he enfrentado como programador, pero desde otra "perspectiva".

Se define a la administración como la acción de planear, organizar, dirigir, controlar y evaluar una determinada actividad. La administración es una de las actividades principales que ejecuta un líder dentro de un proyecto (ya sea de un equipo de trabajo o ya sea de proyecto).

Hablaré pues de los desafíos que he tenido como líder de un equipo de trabajo.

3.2.1 Administración de las indefiniciones en los requerimientos del proyecto

Las carencias que se encuentran frecuentemente en la definición de los requerimientos de un sistema, contribuyen a lo largo del proyecto para que se susciten inconformidades con el mismo por parte del cliente (inclusive puede afectarse la percepción del usuario final de la aplicación sobre la calidad del trabajo realizado, lo que dificulta la continuidad de los consultores como sus proveedores de software).

Cuando existe una inconformidad, en consecuencia se presenta el retrabajo y éste influye directamente en el cumplimiento de las entregas en tiempo y forma del sistema que se está desarrollando. Para hacer frente a ésta situación, es necesario aplicar y administrar las actividades correctivas que se acuerden para el proyecto dentro de un equipo de trabajo es responsabilidad de su respectivo líder.

Comunicar al equipo de trabajo en qué consisten las modificaciones correctivas, los tiempos estimados para implementarlas y probarlas así como la entrega de las mismas es una de mis responsabilidades con mi equipo.

Pero como lo mencione en los desafíos como programador, el retrabajo no es bien visto por nadie y en muchas ocasiones provoca descontento dentro del equipo de trabajo, traduciéndose en resistencias para ejecutar las tareas asignadas al mismo. Aquí un poco de persuasión y paciencia, además de dejar claro que no es una decisión propia sino colectiva, son en conjunto la fórmula que personalmente me ha funcionado en situaciones como la antes descrita.

3.2.2 Administración de recursos técnicos y tecnológicos

Difícilmente dentro de los proyectos se cuentan con los recursos técnicos o tecnológicos adecuados para cumplir sin contratiempos los objetivos de desarrollo y pruebas del sistema. Esta situación es contradictoria pues se sabe que para lograr el éxito de un

proyecto se debe contar de antemano con los recursos humanos, financieros y técnicos necesarios.

Aceptar un proyecto que tiene deficiencias con alguno o algunos de los recursos ya mencionados representa un riesgo para trabajar sin contratiempos en el desarrollo del sistema. Por ejemplo, si el número de recursos humanos es insuficiente, la carga de trabajo para los miembros que conforman el equipo se vuelve muy pesada y las horas de la jornada laboral pueden incrementarse para que se pueda cumplir con los objetivos.

Otro desafío para el éxito del proyecto es contar con personal con la preparación tecnológica adecua. El riesgo por aceptar un proyecto que implique una innovación tecnológica y que no exista el personal lo suficientemente preparado en la o en las nuevas herramientas a emplear, es muy elevado puesto que se puede comprometer la continuidad del proyecto.

En ocasiones es mejor aceptar construir una aplicación con una tecnología ampliamente difundida que desarrollar una aplicación con nuevas herramientas que no tiene el vasto soporte técnico que se les requiere.

Al no contar con expertos en las tecnologías empleadas, debe dedicarse tiempo a la investigación de las mismas, lo que implica tiempo y costo para el proyecto, pues conlleva a una serie de actividades extras que pueden no estar contempladas en el plan de trabajo.

Cuando no se cuenta con la infraestructura adecuada y se tienen por ejemplo servidores de aplicaciones inestables o una red de conexión demasiada lenta (por carecer de un ancho de banda adecuado o contar un número elevado de usuarios), el desarrollo y las pruebas del sistema se entorpecen

Por lo anterior, me ha sido indispensable contemplar en el plan de trabajo de mi equipo un tiempo de holgura que permita a los miembros del equipo efectuar el trabajo de investigación que sea necesario para resolver cualquier duda tecnológica que se presente; asimismo, planear y solicitar al personal de infraestructura las instalaciones de nuestros desarrollos en los servidores de aplicaciones correspondientes o en los ambientes de pruebas para aplicar los controles de calidad a los mismos, son las estrategias que he empleado con resultados eficaces para que el trabajo del equipo no se detenga por éste tipo de inconvenientes técnicos.

3.2.3 Administración de los recursos humanos

Personalmente, la administración de los recursos humanos ha sido la parte más complicada durante mi experiencia como líder, puesto que es una actividad que requiere de una gran capacidad de organización para asignar y supervisar las diferentes tareas que tienen que efectuar los diversos integrantes del equipo de trabajo. Lo anterior no siempre resulta sencillo pues los actores que conforman un equipo de trabajo poseen diferentes

personalidades, trabajan de maneras distintas entre sí y cada uno de ellos cuenta con carencias y fortalezas.

Es también mi responsabilidad comunicar al equipo de trabajo oportunamente las decisiones que se toman dentro del proyecto, por ejemplo: qué se especifica (estándares, lineamientos, plantillas a seguir, documentos de diseño etc.), qué se va a desarrollar y cómo se va a desarrollar (herramientas de desarrollo), qué se va a integrar (componentes o módulos del sistema en cuestión) así como el qué, cuándo y cómo se va a entregar el producto que se está desarrollando.

Dentro de un proyecto conformado de diversos equipos de trabajo, el líder de un equipo de trabajo se encarga de dirigir el curso de las acciones que los miembros del mismo deben seguir a partir de las decisiones generales que el líder de proyecto y otros superiores ya han tomado para ser aplicados y vigilados en todo el proyecto.

El líder de un equipo de trabajo labora conjuntamente con los líderes superiores del proyecto para la definición de resoluciones; posee cierto poder de decisión pero no al grado que sus jefes inmediatos. Es un intermediario entre quienes deciden el rumbo del proyecto y quienes construyen, prueban y modifican el mismo.

El líder del equipo de trabajo es el encargado de organizar los tiempos asignados a las actividades que cada miembro del equipo debe ejecutar. Debe anticipar los contratiempos que pudiesen presentarse en el curso de las mismas y brindar tiempos de holgura para absorber el impacto que dichos contratiempos.

A su vez, debe dar un seguimiento puntual al avance de las actividades de cada uno de sus recursos para asegurar el cumplimiento de los objetivos o detectar a tiempo los problemas que pueden influir en las fechas de entregas. De este modo, se solicita se alerta en tiempo a los líderes inmediatos o al mismo líder del proyecto y se solicita el apoyo requerido con anticipación.

Es también importante señalar que el líder de un equipo de trabajo debe gestionar las solicitudes necesarias para que las personas bajo su cargo cuenten con los recursos técnicos, tecnológicos y de negocio para efectuar sin problemas su trabajo en tiempo y forma.

Cuando se está a cargo de un equipo de trabajo, uno de los desafíos es hacer frente a las personalidades de cada una de los actores del equipo. Todo individuo es un ser único y por tanto diferente en su forma de pensar, sentir y actuar; lo anterior se traslada al ambiente laboral ya que el carácter de cada persona se manifiesta también en los lugares de trabajo.

Por ejemplo, cuando se requiere de un “esfuerzo extra” (extender el horario laboral e incluso trabajar fines de semana) dentro de proyecto, lo que para algunos miembros del equipo significa un esfuerzo necesario e indispensable, para otros miembros en un castigo por una mala planeación o definición de tiempo y recursos de los líderes en el proyecto y por lo

tanto no tienen porque laborar ese tiempo extra. Nuevamente, el retrabajo o el esfuerzo extra que hay que brindar para entregar en tiempo al usuario o al cliente, nunca es bien recibido y es causa frecuente de molestias.

En estos casos, las personas suelen acatar de mala forma la disposición o definitivamente no presentarse a laborar, lo que puede llegar a derivar en situaciones de roce que van desde las llamadas de atención, las penalizaciones administrativas como descuentos a salario o en ocasiones extremas, si el comportamiento es reiterativo, el término de la relación laboral con dichos integrantes.

Con anterioridad mencione que cada individuo posee una forma de pensar y actuar muy particular, por lo que es necesario dentro de un equipo de trabajo conciliar las diferentes formas de trabajo de cada uno de éstos individuos. Por ejemplo, algunas personas no tienen problemas para trabajar con otras en equipo; otras son muy herméticas con el trabajo que realizan y prefieren realizar ellas mismas cualquier modificación que involucre lo que han venido desarrollando pero a su vez se niegan a trabajar con otros módulos o componentes ajenos a ellas; otras más prefieren trabajar desde muy temprano para no salir tan tarde de la oficina o viceversa, prefieren iniciar su jornada laboral después del horario laboral definido y salir de la oficina ya noche.

Como es de imaginarse, estas diversas formas de laborar pueden representar un problema al momento de conjuntar las acciones que deben seguirse dentro del equipo de trabajo para alcanzar los objetivos de desarrollo, pruebas, modificaciones o entregas. Una manera de enfrentar el problema es el establecimiento de reglas claras:

- ***Trabajo en equipo***: la interacción con todos los actores de equipo de trabajo, incluso aquellos que no son parte del equipo pero si del proyecto, es esencial para realizar las actividades asignadas, ya que una sola persona no posee todo el conocimiento técnico y de negocio que se requiere para lograr los objetivos. Un proyecto depende de la interacción y participación de varios actores, no depende de un solo individuo.

- ***Cooperación para con el resto de los miembros del equipo***: todos los elementos de un equipo de trabajo deben trabajar en conjunto sin importar si se es o no el propietario original del componente o proceso a modificar.

- ***Horarios de trabajo***: todos deben de laborar dentro de un horario establecido para evitar que haya una descoordinación en las actividades conjuntas del equipo.

En los proyectos de dimensiones considerables, es común que se definan formas de trabajo como lo son las herramientas a emplear para el diseño, el desarrollo y las pruebas. En el mundo de la consultoría de sistemas, las personas participan en diferentes proyectos a lo largo de su trayectoria profesional, donde aprenden una o varias formas de trabajo que van llevando consigo al término del proyecto.

Algunas ocasiones, las personas se van especializando tanto en ciertas herramientas o metodologías que adaptarse al uso de nuevas tecnologías o nuevas formas de trabajo les es complicado, ya sea porque ello les representa alguna dificultad de aprendizaje o presentan alguna resistencia al cambio. En casos mayores es simplemente una cuestión de mala actitud.

Cuando se trata de un problema de entrenamiento, se brinda la capacitación necesaria para que el recurso adquiera el perfil que se le solicita o en su defecto, se asigna a alguien más (con las habilidades requeridas) a las actividades que demandan el conocimiento técnico del cual carece el primer recurso. Si por el contrario, hay una actitud "negativa" a aprender o manejar las herramientas que se emplean en el proyecto, se procede a solicitar el cambio de área del recurso o se prescinde del mismo de forma definitiva (un caso extremo y muy raro que solo me ha tocado presenciar una sola vez).

Por último, es bien sabido que un líder debe tomar decisiones para alcanzar los objetivos que tiene planteados para su equipo de trabajo. Tiene libertad plena para asignar actividades y tiempos de forma interna y cuando es necesario aplicar alguna medida correctiva para retomar el curso del avance de las actividades a realizar, lo hace.

Si nuevamente se presenta alguna resistencia dentro del equipo de trabajo por parte de algún recurso, se platica con el mismo y si no se llega a algún acuerdo, se escala a los siguientes niveles de la administración del proyecto, es decir, a los líderes inmediatos o incluso al líder del proyecto, según sea el caso. En conflictos con los recursos humanos, el líder de un equipo de trabajo no tiene la facultad para tomar una decisión definitiva sobre la continuidad de las personas en el equipo, pero si la responsabilidad de revisar primero las inconformidades que surjan dentro del mismo y negociar para llegar a algún punto de acuerdo.

Finalmente, resumiendo mi muy breve experiencia laboral como líder de un equipo de trabajo, los desafíos más comunes a los que he hecho frente han sido principalmente de organización y planeación de tiempos, problemas de actitud por parte de los recursos que tengo a cargo así como la toma de decisiones que determinan el éxito o el fracaso en el alcance de los objetivos que como equipo tenemos asignados.

4. CASOS DE ÉXITO

En informática, la *consultoría* se refiere a la labor de asesoría externa realizada por una empresa especializada para la implementación o integración de una aplicación, la actualización del existente, el desarrollo de un proyecto determinado, etcétera. Además del aspecto puramente informático, implica actividades considerables de organización y gestión del negocio.

Todas las instituciones públicas y privadas para las que he otorgado consultoría en sistemas, son entidades que manejan un importante volumen de información, por lo que les es vital la administración de una enorme cantidad de información, echando mano de los consultores externos pues difícilmente cuentan con personal debidamente calificado y en cuestión de costos, les es mucho más rentable "emplear" los servicios informáticos de terceros.

Desde el inicio de mi carrera profesional, he trabajado como un consultor de sistemas en el sector público y privado, dónde me he desempeñado principalmente como programador Java (de hecho, me inicie como programador junior en C y Java, pero adquirí un mayor conocimiento en este último) y ya de manera reciente y breve, como líder de un equipo de trabajo.

Básicamente, las funciones más importantes que he llevado a cabo durante estos cinco años de experiencia laboral como programador han sido el desarrollo de nuevas aplicaciones, el soporte y mantenimiento a estas mismas o a las ya existentes y como líder, la administración y dirección de las actividades de trabajo de un equipo de programadores.

En el presente capítulo, haré una breve descripción de las actividades realizadas en cada uno de los proyectos en los que he formado parte y los que finalmente me han la experiencia laboral que he conseguido a lo largo de este años como profesional de la información.

4.1 Experiencia laboral con Java y J2EE en el Sector Privado

4.1.1 Programador Junior

Cliente: **Grupo Iusacell (Julio – Diciembre de 2005)**
Empresa consultora: **Getronics de México**

Getronics es una empresa transnacional holandesa de consultoría ITC (Tecnología Informática y de la Comunicación) que cuenta con presencia en varios países de Europa, Asia y América. En México, su actividad principal es la asesoría en telecomunicaciones y redes y en menor medida, proporciona también servicios de consultoría informática.

Getronics fue la primera empresa que me dio la oportunidad profesional al finalizar la carrera de ingeniería en computación gracias a que uno de sus clientes, *Grupo Iusacell*, se interesó en mis conocimientos en el lenguaje de programación C. *Grupo Iusacell* o

simplemente *Iusacell*, es una empresa de telefonía móvil y una de las más importantes en el país por de números de usuarios con el que cuenta.

Ya en Iusacell, en el área de Aprovisionamiento, la primera actividad asignada fue analizar el código fuente en C de la aplicación que empleaban para aprovisionar los dispositivos celulares, es decir, para dotarlos de funcionalidad (línea, servicios, mensajes de texto, etc.).

El análisis formaba parte de un plan de modernización de tecnología del área de Aprovisionamiento: los nuevos desarrollos se estaban programando en Java y se buscaba trasladar las aplicaciones legadas en C a componentes Java y aplicaciones J2EE. En una etapa inicial, dicho análisis estuvo orientado a partes específicas del negocio de aprovisionamiento; así, con la información resultante, revisé que los componentes modernizados en Java ejecutarán la misma lógica de negocio analizada y documenté el funcionamiento encontrado para el momento futuro en que el resto de los módulos Java fuesen programados.

La segunda actividad que me fue encargada, fue la construcción de algunos componentes J2EE. En ese entonces no tenía el conocimiento requerido en Java ni en J2EE, pero el apoyo de algunos compañeros de trabajo y la documentación encontrada en Internet fueron de valiosa ayuda. Los elementos J2EE que construí fueron Servlets y EJB's de sesión.

Los Servlets que desarrolle recibían información de la aplicación legada en C vía sockets y enviaban una respuesta a la misma en forma de cadena de datos. Los EJB's de sesión tenían por objetivo ejecutar diferentes procedimientos almacenados (programas almacenados en a una base de datos).

Las herramientas de desarrollo empleadas durante mi participación en Iusacell fueron para Java el Websphere Development Studio for Java, un IDE de IBM que integra una interfaz para codificar clases, interfaces, servlets, ejb's (entre otros) y un servidor de aplicaciones (WAS o WhebSphere Application Server) con contenedores WEB y de EJB's. Para C, utilizaba el compilador de C que estaba disponible en un servidor Linux y a través de la cual tenía acceso a una versión de desarrollo de la aplicación legada.

Figura 4.1 ***Ejemplo de la Clase de un EJB de Sesión***

```
package mx.com.iusacell.ejb;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.SQLException;
…

/**
 * Implementación para la clase Enterprise Bean: EJBAddons
 */
public class EJBAddonsBean implements javax.ejb.SessionBean {

        private javax.ejb.SessionContext mySessionCtx;
        public int res = 1;
```

```
private static final Log log = LogFactory.getLog(EJBAddonsBean.class);

public javax.ejb.SessionContext getSessionContext() {
        return mySessionCtx;
}

public void setSessionContext(javax.ejb.SessionContext ctx) {
        mySessionCtx = ctx;
}

public void ejbCreate() throws javax.ejb.CreateException {
}

public void ejbActivate() {
}

public void ejbPassivate() {
}

public void ejbRemove() {
}

/**  Crea una conexión a la base de datos con un data source **/
private Connection conecta(){
    Connection con = null;
    try{
                ObjectLookup ol = ObjectLookup.getInstance();
                con =ol.getConnection("java:comp/env/AddonsAlias");
                log.info("Conexion a base de datos de los addons establecida");
     }catch(NamingException e){
                log.info(e.toString());
     }catch(SQLException e){
                log.info(e.toString());
     }
     if(con==null)
        log.info("Error al intentar obtener una conexion a la DB, conexión nula");
        return con;
 }

/** Activa o desactiva los servicios topados de un objeto línea */
public int servicioTopado(Linea l, String signo)throws TimeOutException{
    String x;
    ArrayList v = new ArrayList(l.getServicios());
    Iterator it = v.iterator();
    try{
        CallableStatement cs =
                 conecta().prepareCall(("{? = call AprovisionaKBytes(?,?,?)}"));
        while(it.hasNext()){
            cs.registerOutParameter(1, Types.INTEGER);
            cs.setString(2,l.getMDN());
            x=Integer.toHexString(((Servicio)it.next()).id).toUpperCase();
            log.info("Aprovisionando el servicio topado: "+x.trim()+" para el MDN:      "+l.getMDN());
           cs.setString(3,x);
           if(signo.equals("+"))
                cs.setString(4,"A");
           if(signo.equals("-"))
                cs.setString(4,"D");
           cs.execute();
           res = cs.getInt(1);
           log.info("Resultado de la operación a la DB para el servicio: " +x+" :"+res);
        }
        cs.close();

   }catch(SQLException e){
        log.info(e.toString());
```

```
        }
      return res;
    }


    /** Resetea todos los servicios topados de un objeto línea */
    public int reseteaAddon(Linea l) throws SQLException {
          String x;
          ArrayList v = new ArrayList(l.getServicios());
          if(v.size()>0){
                Iterator it = v.iterator();
          try{
                CallableStatement cs =
                conecta().prepareCall(("{? = call APROVISIONAKBYTES(?,?,?)}"));
                while(it.hasNext()){
                            cs.registerOutParameter(1, Types.INTEGER);
                            cs.setString(2,l.getMDN());
                            x=Integer.toHexString(((Servicio)it.next()).id).toUpperCase();
                            log.info("Reseteando el servicio de la linea: "+x.trim()+" para el MDN: "+l.getMDN());
                            cs.setString(3,x);
                            cs.setString(4,"R");
                            cs.execute();
                            res = cs.getInt(1);
                           log.info("Resultado de la operación a la DB para el servicio: " +x+" :"+res);
                }
           }catch(SQLException e){
              log.info(e.toString());
          }
          return res;
          }else return 0;

    }


    /**  Resetea todos los servicios topados de un objeto línea  */
    public int reseteaAddon(String mdn, ArrayList array)throws Exception{
        String x;
        if(array.size()>0 && mdn.equals("")==false ){
                Iterator it = array.iterator();
                try{
                  CallableStatement cs =
    conecta().prepareCall(("{? = call APROVISIONAKBYTES(?,?,?)}"));
                  while(it.hasNext()){
                            cs.registerOutParameter(1, Types.INTEGER);
                            cs.setString(2,mdn);
                            x=((String)it.next()).toUpperCase();
                            log.info("Reseteando el servicio "+x.trim()+" del MDN: "+mdn);
                            cs.setString(3,x);
                            cs.setString(4,"R");
                            cs.execute();
                            res = cs.getInt(1);
                           log.info("Resultado de la operación a la DB para el servicio: " +x+" :"+res);

                  }
                }catch(SQLException e){
                            System.out.println(e.toString());
                }
                return res;
       }else return 0;
    }
}
```

De forma general, durante mi experiencia laboral en Iusacell, aprendí cosas pequeñas como la lectura/ escritura de archivos de texto plano, la implementación de conexiones a bases de datos y de conexiones de componentes vía sockets y vía CORBA (una tecnología robusta empleada en la comunicación de elementos distribuidos y programados en diferentes plataformas).

Algo negativo que encontré durante mi breve estancia en el área de aprovisionamiento de Iusacell, fue la falta de control y organización de las actividades que realizaba el equipo de trabajo. No existían planes de acción ante contingencias o emergencias (caída de servidores, teléfonos móviles sin servicio, recargas de saldo que no llegaban, etc.) ni documentación sobre qué hacer en caso de presentarse problemas en los sistemas.

Las correcciones las ejecutaban únicamente los programadores dueños de las aplicaciones pues nadie más tenía el conocimiento para ello. Cómo es de imaginarse, los horarios de trabajo no se respetaban y las llamadas por la noche para solicitar la presencia física del personal en la oficina eran comunes durante los fines de semana. Esa falta de organización a mi parecer, era el modo de operación con el que se conducían diferentes áreas que interactuaban con la de aprovisionamiento. Tampoco existía una rotación de personas para el monitoreo de eventos: las mismas personas encargadas de desarrollar las aplicaciones eran las mismas encargadas de migrar datos, hacer guardias nocturnas y asistir en la corrección de problemas.

Las pruebas que se programaban en el área de servidores no tenían control alguno; los administradores de los mismos cancelaban continuamente sin aviso previo y ello se traducía generalmente en una pérdida de tiempo importante para mostros los programadores puesto que permanecíamos en la oficina hasta altas horas de la noche, como se podrá ver, en vano.

Lo anterior, me ayudo a formar una percepción personal de organización de mis actividades laborales y de previsión al programar con el fin de evitar que mis desarrollos presentaran fallas en producción.

Fue también durante mi participación en Iusacell que me inicie en forma en la programación con Java y fui conociendo poco a poco y de fondo las características del lenguaje, lo que me preparo para dar un salto a otra empresa de consultoría, IDS Comercial y a otro cliente, Grupo Nacional Provincial.

4.1.2 Programador Analista Java

Cliente: **Grupo Nacional Provincial - GNP (Diciembre 2005 – Marzo 2007)**
Empresa consultora: **IDS Comercial**

IDS Comercial es una empresa mexicana que brinda consultoría TI (Tecnología de la Información). Es la empresa consultora que me brindo mi segunda oportunidad laboral para trabajar con uno de sus clientes más importantes de ese entonces, Grupo Nacional Provincial.

Grupo Nacional Provincial es una empresa aseguradora que proporciona seguros de vida, seguros de gastos mayores, seguros de autos, seguros de casa/habitación y seguros

empresariales. Al inicio, fui asignada a la revisión del código fuente de una aplicación gráfica llamada INFO cuyo objetivo era la administración y seguimiento de los siniestros originados por pólizas de seguros de vida. El objetivo del análisis de dicho código era obtener la lógica de negocio empleada y que sería traslada a una nueva aplicación sustituta, hecha con un *workflow*.

Se define como workflow a la automatización de un proceso del negocio, entero o en partes durante el cual los documentos, la información o las tareas se pasan de un recurso (ser humano o máquina) a otro para una acción específica en función de un sistema de reglas procesales.

Sin embargo, el proyecto más importante en el que participe dentro de GNP fue NUEVO SERBE (Nuevo Servicio a Beneficiarios), un desarrollo que inicio desde cero (más no así mi participación en el) para la línea de seguros de vida.

NUEVO SERBE es hoy en día una aplicación WEB que extrae la información de los asegurados (así como de sus pólizas) desde aplicaciones legadas (mainframe) o en Visual Basic (otro lenguaje de programación de Microsoft) para presentarla en pantallas HTML pero definidas en archivos XML (una innovación tecnológica según los diseñadores del sistema).

Ya con más conocimientos en Java y J2EE, los módulos a los que fui asignada fueron los desacopladores a dos sistemas legados en mainframe: al Sistema LIFE70 y al Sistema SISI.

En informática, el término desacoplar se emplea para identificar la separación de bloques de software que no deben tener dependencia alguna entre ellos. En NUEVO SERBE, los desacopladores a los sistemas LIFE70 y SISI eran componentes que recibían peticiones a través de mensajes depositados en colas de mensajes MQ (Message Queue), procesan la información que obtienen desde las pantallas del mainframe y entregan la respuesta a dichas solicitudes en otro mensaje MQ. Los desacopladores eran componentes que todo el tiempo estaban ejecutándose pues debían atender las solicitudes encoladas en todo momento.

La extracción de la información de las aplicaciones en mainframe se hizo a través de la ejecución de macros con una herramienta de IBM llamada HATS. Una macro es el almacenamiento cronológico de pulsaciones de teclas, acciones de comandos, instrucciones, e incluso movimientos del mouse, con el fin de automatizar o economizar procedimientos. En este caso en particular, las macros representan una serie de pantallas capturadas que se ejecutaban para la extracción de datos precisos y útiles al negocio del NUEVO SERBE.

Para la manipulación de la información a enviar o recibir de las macros, HATS proporciona a su vez una serie de objetos Java llamados *objetos de integración*, particulares a cada macro HATS. De esta manera, por medio de los objetos de integración, se pueden enviar datos como usuario, contraseña y número de póliza a una macro HATS que devolverá a su vez la información de uno o más siniestros de dicha póliza a través de los objetos Java, por ejemplo.

Figura 4.2 ***Interacción entre el desacoplador LIFE70, HATS y la aplicación mainframe LIFE70***

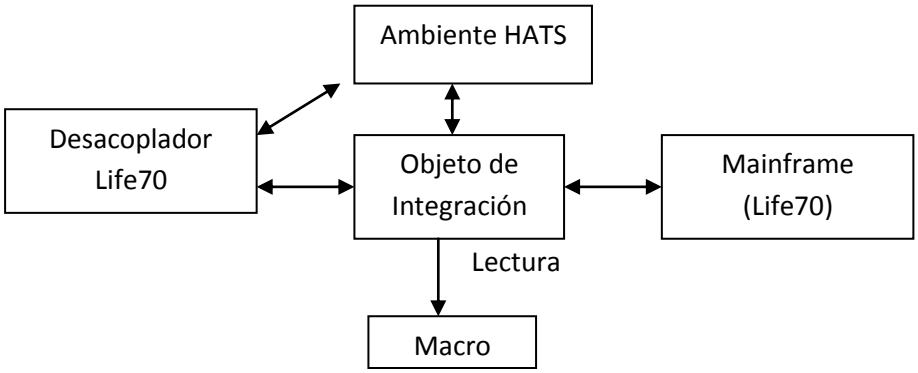


Figura 4.3 ***Ejemplo del inicio de grabación de una macro en HATS***

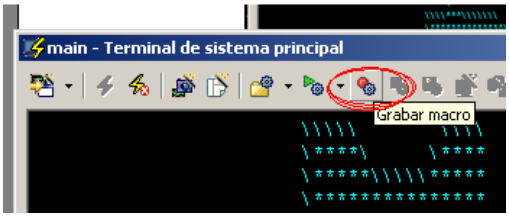


Figura 4.4 ***Ejemplo de una pantalla de la aplicación legada LIFE70***

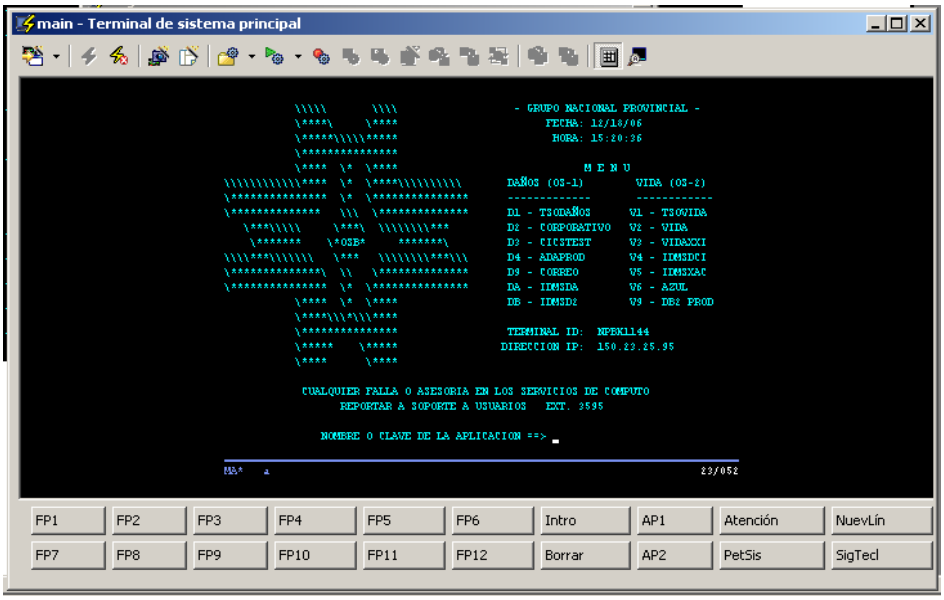
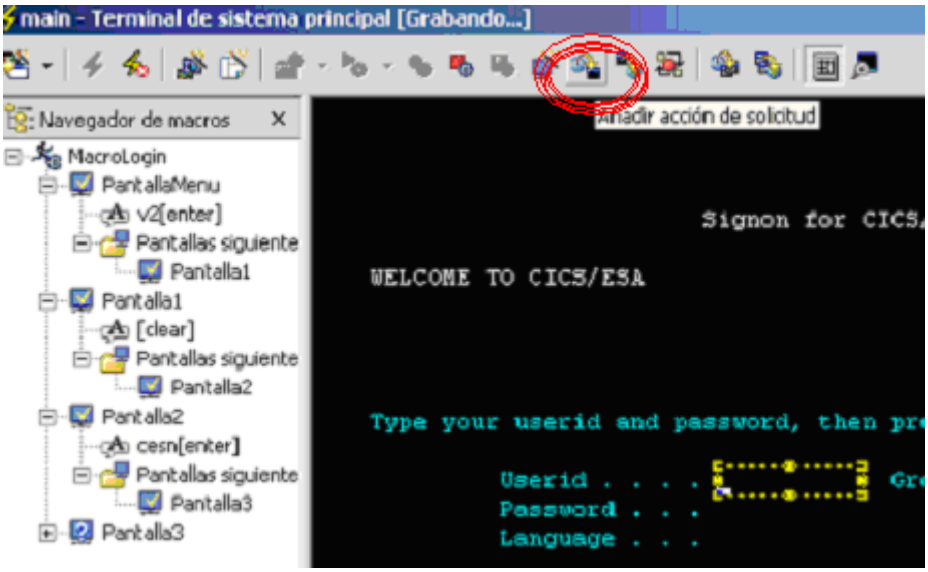


Figura 4.5 ***Ejemplo de la definición de un parámetro de entrada a enviar a la macro***



Dentro de los desacopladores, la ejecución de las macros, por medio de los objetos de integración, era controlada por una serie de servicios que no eran más que unas clases que implementan el patrón de diseño Singleton (ver capítulo 2). Casi todos los servicios realizaban en forma más o menos uniforme, las siguientes tareas:

- Recibir los mensajes de solicitud para extraer la información que las macros necesitaban para poder funcionar correctamente.
- Ejecutar las macros vía los objetos de integración.
- Recibir la información devuelta en las macros para almacenarla en un mensaje de respuesta (dicha información podía estar compuesta por una serie de parámetros o una lista de objetos).
- Depositar el mensaje de respuesta en la cola de mensaje de salida.

Las herramientas utilizadas en el proyecto para construir los desacopladores fueron nuevamente el IDE IBM WebShepre Development Studio, IBM MQ Series, IBM Hats y XML.

El desarrollo de los desacopladores LIFE70 y SISI consistió por mi parte en la programación y orquestación de todos los servicios que los componían así como la extracción de información de las macros necesarias, pues el módulo de conexión a las colas de mensajes era ya un componente programado.

Cuando llegó el momento de integrar los desacopladores a NUEVO SERBE, estuve encargada de solicitar las instalaciones en los servidores de desarrollo (actividad que no siempre resulto en tiempo y forma por el retraso en la atención que había por parte del área de infraestructura de GNP), dar seguimiento y solución a las fallas detectadas en los servicios en las pruebas de estrés y de concurrencia y en la revisión general de calidad en la

programación del software programado (de estas últimas pruebas aprendí algunas de las convenciones de codificación de código fuente en Java).

La ejecución de las pruebas anteriores requirió una anticipada planeación con el área de servidores de GNP, ya que era la única dentro de la institución que contaba con el software adecuado para realizar dichas evaluaciones. En esta etapa la interacción con el área de servidores fue más amplia pero a su vez más ágil debido a que la aplicación de NUEVO SERBE iba a liberarse a producción y por ende, su atención era más importante.

Con el tiempo, he aprendido mejores prácticas de programación y haciendo hoy en día una retrospectiva en el pasado, puedo detectar algunas áreas de oportunidad para los desacopladores LIFE70 y SISI:

- Uso de mensajes en archivos de propiedades y no en código duro
- Mayor atomización en los métodos que componen las clases
- Métodos que hagan una sola cosa a la vez

Hay que mencionar que el uso de las licencias de IBM HATS resulto ser muy costoso, pero afortunadamente GNP contó con la disposición y los recursos económicos para emplear de manera legal el software y de paso, nos permitió contar con el soporte que requeríamos ya que en México hasta ese momento, no se había utilizado HATS.

El proyecto termino satisfactoriamente la fase de construcción, pero no me fue posible permanecer en el mismo para ver la implementación en producción del NUEVO SERBE, el presupuesto de GNP para mí y otros consultores se había agotado e IDS Comercial decidió reasignarme a otro cliente, esta vez del ámbito gubernamental: el Instituto del Fondo Nacional para la Vivienda de los Trabajadores.

4.2 Experiencia laboral con Java y J2EE en el Sector Público

4.2.1 Ingeniero de Sistemas (Programador Java *Senior*)

Cliente: **Instituto del Fondo Nacional para la Vivienda de los Trabajadores – INFONAVIT (Marzo 2007 – Enero 2009)**
Empresa consultora: **IDS Comercial**

El Instituto del Fondo Nacional para la Vivienda de los Trabajadores (INFONAVIT) es la institución encargada de otorgar financiamiento a los trabajadores derechohabientes y acreditados que están en búsqueda de créditos para la adquisición de alguna vivienda o la construcción de la misma.

El INFONAVIT ha formalizado 245,432 créditos en lo que va del 2010[9], lo que implica un manejo importante de volumen de datos dentro del instituto.

[9] Avance semanal de créditos otorgados al 25 de julio de 2010, INFONAVIT

Mi experiencia profesional dentro del instituto se centro, en una primera etapa, en el proyecto *Mejoras al Sistema ARC Cobranza* (donde fui el único consultor en Java) y en una segunda etapa, en el proyecto *Mantenimiento y actualizaciones del Sistema SARA*, cuyo equipo de trabajo estaba conformados por varios programadores Java/J2EE.

4.2.1.1 Mejoras al Sistema ARC Cobranza

El Sistema ARC Cobranza es una aplicación del área de cobranza del INFONAVIT desarrollada en *People Soft*, un software de Planificación de Recursos Empresariales (E.R.P. o *Enterprise Resource Planning*), empleado en la gestión de Recursos Humanos , en la gestión de las Relaciones con los Clientes (CRM*, Customer Relationship Management*) y en la gestión de Nómina a grandes empresas.

Cuando los trabajadores se ven imposibilitados por diversas razones y motivos para pagar sus créditos de vivienda, pueden solicitar una reestructuración de su adeudo con el instituto a través de un de convenio. En un inicio, dichos convenios se creaban de manera masiva para todos los créditos existentes en el sistema, aún cuando los acreditados no hubiesen solicitado documento alguno. Como es de imaginarse, esta actividad representaba un gasto de recursos innecesarios mes a mes, por lo que se buscó la manera de generar los convenios en línea y a demanda. Una aplicación WEB J2EE fue la solución.

Así, programé una pequeña aplicación WEB nombrada *ARCConvenios* que genera dinámicamente un convenio en formato PDF y que se visualiza en un navegador de internet para que sea manipulado por el usuario como más le convenga. De esta forma, se generan únicamente los convenios que se requieren y lo mejor, en línea. La aplicación fue diseñada y programada para recibir de las solicitudes del sistema ARC Cobranza (People Soft solo puede invocar direcciones vía GET) toda la información a vaciar en el convenio referente al acreditado y su número de crédito.

Como herramientas de desarrollo emplee nuevamente el IDE Websphere Development Studio y una librería de software libre llamada Jasper Reports. Jasper Reports es una herramienta muy versátil que permite crear reportes y documentos en diferentes formatos (Word, PDF, Excel, CVS) de forma sencilla, ya que las especificaciones de los reportes se realizan a través de plantillas en XML.

La interacción entre el ARCCobranza en PeopleSoft y el ARCConvenios en J2EE se efectuaba de la siguiente manera:

- El usuario seleccionaba desde ARCCobranza el acreditado que está solicitando un convenio en particular.

- A su vez, ARCCobranza se encargaba de extraer toda la información a vaciar en el convenio y hecho esto, enviaba vía GET una solicitud a ARCConvenios con el nombre del convenio y una cadena de datos con la información del crédito y el acreditado.

- ARCConvenios recibía esta información y procedía a almacenar los datos del crédito y del acreditado en un objeto Java llamado Map.

- Con el nombre del convenio, ARCConvenios buscaba el archivo de propiedades *.properties* que contenía todo el texto del convenio en cuestión y junto con la información contenida en el objeto Map, vaciaba todos los datos en la correspondiente plantilla jrxml (jasper report xml). Cada convenio tenía su propia plantilla y su propio archivo de propiedades.

- Hecho esto, ARCConvenios terminaba la solicictud de ARCCobranza presentando el convenio en el navegador de Internet en formato PDF.

Figura 4.6 ***Ejemplo de un archivo de propiedades para convenios – Convenio RLDFP070502.properties***

```
encabezado = CONVENIO MODIFICATORIO DE CONTRATO DE APERTURA DE CRÉDITO SIMPLE
tituloConvenio = CON QUITA CONDICIONADA C-RLDF-001
deQuien = POR "EL INFONAVIT"
paraQuien = "EL ACREDITADO"
repInfonavit = JEFE DEL AREA DE COBRANZA
plantilla = TEMPLATE
tipoPlantilla = simple
########################################################################
# Texto del convenio RLDF Privado
########################################################################
text.paragraph = |3|CONVENIO MODIFICATORIO DE CONTRATO DE APERTURA DE CRÉDITO SIMPLE CON GARANTÍA HIPOTECARIA QUE CELEBRAN POR UNA PARTE, EL INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES, EN LO SUCESIVO EL "INFONAVIT" REPRESENTADO POR {Variable3}, JEFE DE COBRANZA DE LA DELEGACIÓN REGIONAL {Variable63}, Y POR LA OTRA PARTE, EL(LA) SEÑOR(A) EN LO SUCESIVO EL "TRABAJADOR" {Variable4}, DE CONFORMIDAD CON LOS SIGUIENTES ANTECEDENTES, DECLARACIONES Y CLÁUSULAS:
```

Figura 4.7 ***Formato PDF para "CONVENIO MODIFICATORIO DE CONTRATO DE APERTURA DE CRÉDITO SIMPLE CON QUITA CONDICIONADA"***

INFONAVIT

INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES
CONVENIO MODIFICATORIO DE CONTRATO DE APERTURA DE CRÉDITO SIMPLE
CON QUITA CONDICIONADA C-RLDF-001

No. De Crédito: 1300006437 No. De Folio: 288593

CONVENIO MODIFICATORIO DE CONTRATO DE APERTURA DE CRÉDITO SIMPLE CON GARANTÍA HIPOTECARIA QUE CELEBRAN POR UNA PARTE, EL INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES, EN LO SUCESIVO EL "INFONAVIT" REPRESENTADO POR CLAUDIA SPINOLA SESE-A, JEFE DE COBRANZA DE LA DELEGACIÓN REGIONAL HIDALGO, Y POR LA OTRA PARTE, EL(LA) SEÑOR(A) EN LO SUCESIVO EL "TRABAJADOR" MONZALVO JARN MA JOSEFINA, DE CONFORMIDAD CON LOS SIGUIENTES ANTECEDENTES, DECLARACIONES Y CLÁUSULAS:

ANTECEDENTES

1.-PRIMERO. APERTURA DE CRÉDITO. CON FECHA 17 DE MARZO DE 2000, SE HIZO CONSTAR, ENTRE OTROS ACTOS, EL CONTRATO DE APERTURA DE CRÉDITO SIMPLE CON GARANTÍA HIPOTECARIA, EN LO SUCESIVO "EL CONTRATO DE CRÉDITO", POR EL CUAL EL "INFONAVIT" OTORGÓ A EL "TRABAJADOR" UN CRÉDITO SIMPLE POR LA CANTIDAD DE $207,388.80 PESOS MONEDA NACIONAL, EQUIVALENTE EN LA FECHA DEL OTORGAMIENTO DEL CRÉDITO A 180.0000, VECES SALARIO MÍNIMO MENSUAL DEL DISTRITO FEDERAL, EN ADELANTE VSMM, PARA QUE FUERA DESTINADO A LA ADQUISICION DEL INMUEBLE UBICADO EN JUAN DE DIOS PESA 117 CUAUHTEMOC PACHUCA DE SOTO. EL CRÉDITO OTORGADO SE IDENTIFICA POR EL "INFONAVIT" CON EL NÚMERO DE CRÉDITO 1300006437.

2.-SEGUNDO. HIPOTECA. PARA GARANTIZAR EL PAGO Y CUMPLIMIENTO DE LAS OBLIGACIONES DERIVADAS DEL CONTRATO DE CRÉDITO Y MEDIANTE EL MISMO INSTRUMENTO REFERIDO EN EL ANTECEDENTE PRIMERO, EL "TRABAJADOR" CONSTITUYÓ HIPOTECA EN PRIMER LUGAR Y GRADO A FAVOR DEL "INFONAVIT", SOBRE EL INMUEBLE REFERIDO EN EL MISMO ANTECEDENTE PRIMERO.

DECLARACIONES

I.- DECLARA EL "INFONAVIT" A TRAVÉS DE SU REPRESENTANTE, QUE:

a) EN CONSIDERACIÓN A LA SOLICITUD DEL "TRABAJADOR", HA SIDO APROBADA SU SOLICITUD DE MODIFICACIÓN AL CONTRATO DE CRÉDITO, POR LO QUE POR ESTE INSTRUMENTO SE FORMALIZAN LAS MODIFICACIONES QUE PACTAN LAS PARTES RESPECTO DEL CONTRATO DE CRÉDITO, MISMAS QUE SE LIMITAN A LOS ASPECTOS QUE SE CONTIENEN EN EL PRESENTE CONVENIO MODIFICATORIO.

b) QUE EN CUMPLIMIENTO AL ACUERDO NÚMERO RCA-0339-08/02, DE FECHA 28 DE AGOSTO DEL 2002 DE SU CONSEJO DE ADMINISTRACIÓN, HA APROBADO LA REESTRUCTURACIÓN DEL CRÉDITO NÚMERO 1300006437.

II.- DECLARA EL "TRABAJADOR", BAJO PROTESTA DE DECIR VERDAD, QUE:

a) PARA CUMPLIR CON SUS OBLIGACIONES DE PAGO DEL SALDO INSOLUTO DEL CRÉDITO, DE LOS INTERESES DEVENGADOS Y DE LOS DEMÁS ACCESORIOS QUE ADEUDA AL "INFONAVIT" CONFORME AL CONTRATO DE CRÉDITO, SOLICITÓ LA MODIFICACIÓN DEL CONTRATO DE CRÉDITO, EN LOS TÉRMINOS QUE SE PACTAN EN EL PRESENTE CONVENIO, CON EL OBJETO DE OBTENER UNA QUITA EN EL MONTO TOTAL ADEUDADO, Y MANIFIESTA EXPRESAMENTE QUE CONOCE Y COMPRENDE LOS TÉRMINOS Y CONDICIONES DE ESTE CONVENIO, TODA VEZ QUE LE HAN SIDO EXPLICADOS A SU ENTERA SATISFACCIÓN, ASÍ COMO TODOS LOS TÉRMINOS JURÍDICOS Y FINANCIEROS QUE SE USAN EN EL MISMO, POR LO QUE, HABIENDO ANALIZADO Y COMPRENDIDO LOS ALCANCES Y CONSECUENCIAS DE ESTE CONVENIO, MANIFIESTA QUE ES EL ADECUADO Y QUE PUEDE CUMPLIRLO SATISFACTORIAMENTE.

b) LOS DATOS PROPORCIONADOS EN LA SOLICITUD Y DEMAS DOCUMENTOS PRESENTADOS AL "INFONAVIT" PARA EL OTORGAMIENTO DEL PRESENTE CONVENIO SON VERDADEROS Y AUTÉNTICOS.

Los certificados de adeudo son documentos empleados por el área jurídica del INFONAVIT para revisar todos los pagos realizados por el acreditado y que el instituto tiene registrados en su sistema. Para la generación de estos documentos, se creó una aplicación WEB llamada *ARCCertificados*, que al igual que ARCConvenios, interactúa a través de solicitudes vía GET con ARCCobranza.

La diferencia entre ARCConvenios y ARCCertificados es que la primera no hace búsqueda alguna en base de datos para obtener la información que necesita; ARCCertificados si accede al servidor de la base de datos SQLServer para obtener todos los pagos que ha realizado un acreditado a partir del número de crédito que le envía ARCConvenios.

Figura 4.8 ***Ejemplo de Certificado de Adeudo***

INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES

CERTIFICACIÓN DE ADEUDOS

CRUZ ROSADO CARLOS

EL PRESENTE DOCUMENTO LO FIRMA EL(LA) LICENCIADO JOSE ANTONIO MARTINEZ GUIZAR, SUBGERENTE DE AREA DE COBRANZA DE LA DELEGACIÓN REGIONAL QUINTANA ROO DEL INFONAVIT, CON FUNDAMENTO EN LO DISPUESTO EN LOS ARTICULOS 23 FRACCION I DE LA LEY DEL INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES; 1°, 3°, FRACCION VI, 4°, FRACCIÓN XVIII Y 19°, DEL REGLAMENTO INTERIOR DEL INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES EN MATERIA DE FACULTADES COMO ORGANISMO FISCAL AUTONOMO, PUBLICADO EN EL DIARIO OFICIAL DE LA FEDERACION EL 20 DE JUNIO DE 2008, DISPOSICIONES QUE LE FACULTAN PARA CERTIFICAR QUE TODOS LOS DATOS QUE SE IDENTIFICAN EN ESTE CERTIFICADO DE ADEUDOS, COINCIDEN FIELMENTE CON LOS REGISTROS QUE OBRAN EN EL INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES, DE LOS CUALES SE DESPRENDE QUE A LA FECHA DE EMISION DE ESTE DOCUMENTO, EL DEUDOR CRUZ ROSADO CARLOS MANTIENE UN ADEUDO CON EL INFONAVIT, DERIVADO DEL CRÉDITO 2302039593 QUE LE FUE OTORGADO, EL CUAL SE INTEGRA DE LA SIGUIENTE MANERA. ESTE CERTIFICADO CONSTA DE 4 FOJAS.

CREDITO
2302039593
NSS
00080887084711
RFC
CURC740505H70
CURP
-

TIPO DE CRÉDITO	L-III 30 AÑOS VSM
MONTO DE OTORGAMIENTO	158.0000 VSM

FECHA DE EMISIÓN	03 DE AGOSTO DE 2010

DEUDA ACTUAL

SALDO AL 30/06/2010	VSM	PESOS
SALDO FINAL DEL PERIODO	234.9560	410,417.38
CAPITAL	181.0870	317,891.97
INTERESES	51.5690	90,114.83

PRORROGAS

FECHA INICIO	FECHA FIN
16/04/2003	01/05/2004

MOVIMIENTOS EN VSM (Veces Salario Mínimo Mensual)

FECHA	TRANSACCIÓN Y CONCEPTO	ORIGEN	MONTO DE LA TRANSACCIÓN	PAGO A SEGURO	PAGO A INTERESES	PAGO A MORATORIOS	PAGO A CAPITAL	SALDO A CAPITAL
31/10/2003	8892 TRAS.SUBCTA.VIV	-	8.0975	0.0000	0.0000	0.0000	-8.0975	157.2397
31/01/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
29/02/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
31/03/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
30/04/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
31/05/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
30/06/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
31/07/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
31/08/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
17/09/2004	8850 PAGO REGULAR	0329	0.1350	0.0000	-0.1350	0.0000	0.0000	169.5990
17/09/2004	8880 PAGO REGULAR	329	1.8650	0.0000	-1.8650	0.0000	0.0000	169.5990
31/10/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
30/11/2004	SIN MOVIMIENTO	-	-	-	-	-	-	-
17/12/2004	8850 PAGO REGULAR	0329	0.1330	0.0000	-0.1330	0.0000	0.0000	169.2850
17/12/2004	8850 PAGO REGULAR	329	1.7370	0.0000	-1.7370	0.0000	0.0000	169.2850
31/01/2005	SIN MOVIMIENTO	-	-	-	-	-	-	-
28/02/2005	SIN MOVIMIENTO	-	-	-	-	-	-	-

El principio de funcionamiento es el mismo: el sistema ARCCobranza solicita la generación de certificados en forma masiva a través de un Servlet en específico; éste genera todos los certificados de adeudo de los números de crédito que existen en la base de datos en

base a una tabla exclusiva para ello y deposita en una ruta definida en el archivo de configuración web.xml de la aplicación WEB todos los convenios que genero físicamente de manera local (la transferencia de los convenios en formato PDF la realiza por medio de protocolo FTP o File Transfer Protocol).

Por el contrario, si el sistema ARCCobranza solicita la impresión del certificado en forma individual, entonces envía el número de crédito para que ARCCertificados genere el certificado de adeudo a presentar en línea en formato PDF dentro de un navegador de Internet.

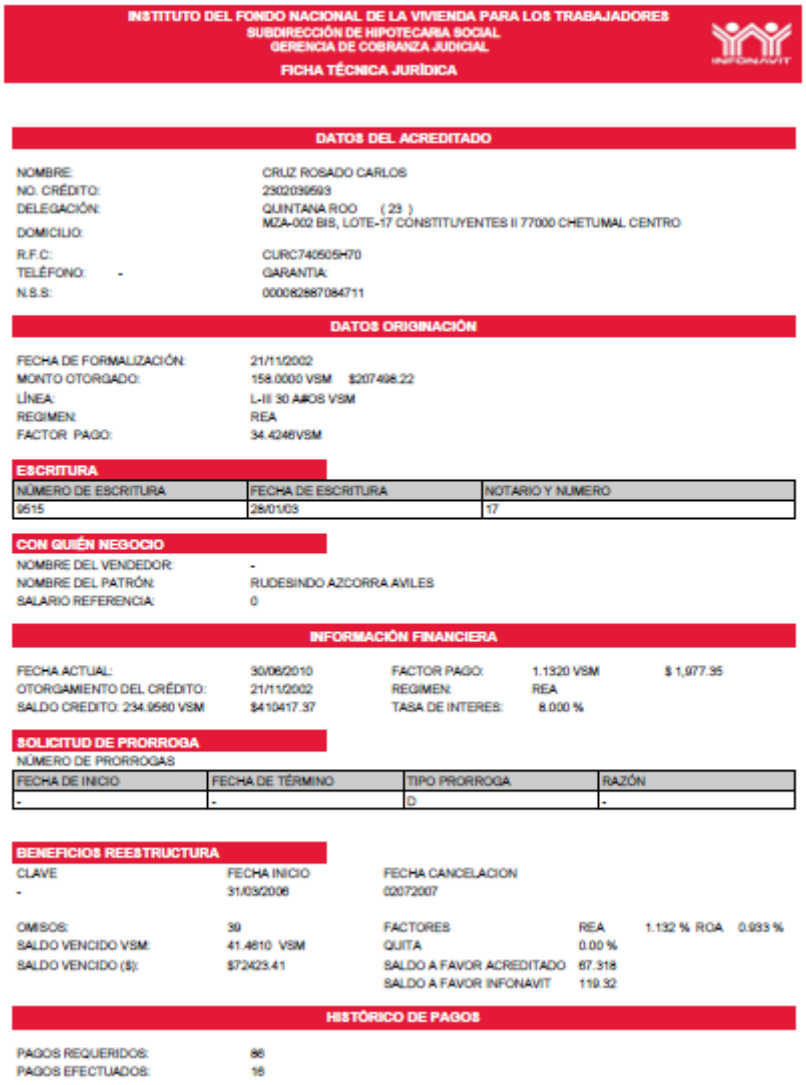
Las plantillas de ARCCertificados poseen otra diferencia contra aquellas definidas en ARCConvenios: la definición de una fuente de datos o data source dentro del archivo jrxml. Dicha fuente de datos fue previamente definida dentro del servidor de aplicaciones; lo interesante de JasperReports es que se hace cargo del manejo de las fuentes de datos especificando únicamente el nombre de la misma dentro de la plantilla jrxml.

Figura 4.9 ***Plantilla en Jasper Reports - Certificado.jrxml***

```
<!-- CUERPO DEL REPORTE -->
<detail>
      <band height="10">
         <!-- CABECERO DEL CERTIFICADO DE ADEUDO -->
         <subreport>
           <reportElement positionType="Float" x="0" y="0" width="450" height="10"
isRemoveLineWhenBlank="true"/>
           <parametersMapExpression> <![CDATA[$P{REPORT_PARAMETERS_MAP}]]>
</parametersMapExpression>
          <subreportParameter name="NEC_NUM_CREDITO">
            <subreportParameterExpression><![CDATA[$P{NEC_NUM_CREDITO}]]></subreportParameterExpression>
          </subreportParameter>
          <subreportParameter name="NEC_NOMBRE">
            <subreportParameterExpression><![CDATA[$F{NEC_NOMBRE}]]></subreportParameterExpression>
          </subreportParameter>
          <subreportParameter name="CB_DESCRIP">
            <subreportParameterExpression><![CDATA[$F{CB_DESCRIP}]]></subreportParameterExpression>
          </subreportParameter>
          <subreportParameter name="CB_TITULO">
            <subreportParameterExpression><![CDATA[$F{CB_TITULO}]]></subreportParameterExpression>
          </subreportParameter>
          <subreportParameter name="CB_NOMBRE_JEFE_JUR">
            <subreportParameterExpression><![CDATA[$F{CB_NOMBR_JEF_JUR}]]></subreportParameterExpression>
          </subreportParameter>
          <subreportParameter name="CB_PUESTO">
            <subreportParameterExpression><![CDATA[$F{CB_PUESTO}]]></subreportParameterExpression>
          </subreportParameter>
          <connectionExpression><![CDATA[$P{REPORT_CONNECTION}]]></connectionExpression>
          <subreportExpression class="java.lang.String">
                    <![CDATA["WEB-INF/jasper/SUBCERTCABECERO.jasper"]]>
          </subreportExpression>
          </subreport>
      </band>
</detail>
```

La ficha técnica es también un documento empleado por el área jurídica del instituto; su finalidad es presentar una “radiografía” de todo el historial del crédito de un trabajador: nombre completo, domicilio, tipo de crédito, saldos, si ha solicitado convenios o reestructuras, etc.

Figura 4.10 ***Ejemplo de Ficha Técnica***



INSTITUTO DEL FONDO NACIONAL DE LA VIVIENDA PARA LOS TRABAJADORES
SUBDIRECCIÓN DE HIPOTECARIA SOCIAL
GERENCIA DE COBRANZA JUDICIAL
FICHA TÉCNICA JURÍDICA

DATOS DEL ACREDITADO

NOMBRE: CRUZ ROSADO CARLOS
NO. CRÉDITO: 2302039593
DELEGACIÓN: QUINTANA ROO (23)
DOMICILIO: MZA-002 BIS, LOTE-17 CONSTITUYENTES II 77000 CHETUMAL CENTRO
R.F.C.: CURC740505H70
TELÉFONO: - GARANTIA:
N.S.S: 00008288704711

DATOS ORIGINACIÓN

FECHA DE FORMALIZACIÓN: 21/11/2002
MONTO OTORGADO: 158.0000 VSM $207498.22
LÍNEA: L-III 30 AÑOS VSM
REGIMEN: REA
FACTOR PAGO: 34.4246VSM

ESCRITURA

NÚMERO DE ESCRITURA	FECHA DE ESCRITURA	NOTARIO Y NUMERO
9515	28/01/03	17

CON QUIÉN NEGOCIO

NOMBRE DEL VENDEDOR: -
NOMBRE DEL PATRÓN: RUDESINDO AZCORRA AVILES
SALARIO REFERENCIA: 0

INFORMACIÓN FINANCIERA

FECHA ACTUAL: 30/06/2010 FACTOR PAGO: 1.1320 VSM $ 1,977.35
OTORGAMIENTO DEL CRÉDITO: 21/11/2002 REGIMEN: REA
SALDO CREDITO: 234.9560 VSM $410417.37 TASA DE INTERES: 8.000 %

SOLICITUD DE PRORROGA

NÚMERO DE PRORROGAS

FECHA DE INICIO	FECHA DE TÉRMINO	TIPO PRORROGA	RAZÓN
-	-	D	-

BENEFICIOS REESTRUCTURA

CLAVE	FECHA INICIO	FECHA CANCELACION			
-	31/03/2006	02072007			
OMISOS:	59	FACTORES	REA	1.132 % ROA	0.933 %
SALDO VENCIDO VSM:	41.4610 VSM	QUITA	0.00 %		
SALDO VENCIDO ($):	$72423.41	SALDO A FAVOR ACREDITADO	67.318		
		SALDO A FAVOR INFONAVIT	119.32		

HISTÓRICO DE PAGOS

PAGOS REQUERIDOS: 86
PAGOS EFECTUADOS: 16

Se creó entonces otra aplicación WEB denominada *ARCFichaTecnica*, cuya finalidad era mantener separados los certificados de adeudo de las fichas técnicas. La ficha técnica es un documento mucho más complicado que los certificados y los convenios ya que está formado por más de 10 platillas jrxml (cada una representando una sección del documento).

De igual forma, ARCFichaTecnica cuenta con la capacidad para generar fichas técnicas en formato PDF de forma individual y en línea o de forma masiva, transfiriendo dichos documentos vía FTP a una ubicación ya predefinida en el archivo web.xml.

ARCCertificados y ARCFichaTécnica no son aplicaciones tan dinámicas como ARCConvenios ya que en ellas solo las plantillas jrxml podían ser modificadas sin necesidad de

recompilar o detener el servidor de aplicaciones, sin embargo, permitieron a ARCCobranza contar con un módulo que generaba documentos PDF de forma rápida, segura y sin una carga excesiva sobre su propio servidor de aplicaciones.

Paralelamente a los tres desarrollos de ARCConvenios, ARCCertificados y ARCFichaTécnica, programe otros módulos independientes y mucho más pequeños como parte del mantenimiento también a ARCCobranza como por ejemplo, la construcción de un componente que permitía el intercambio de información entre los sistemas IVR y ARCCobranza a través de un estándar de comunicación propio del INFONAVIT llamado "TAXME" (comunicación con mensajes XML).

Asimismo programé algunas utilerías principalmente para la validación de archivos, formateo de archivos, transferencia de archivos vía FTP así como cargas y descargas desde sitios web de archivos, empleados de forma unitaria por otros integrantes del equipo de desarrollo para facilitar las tareas que les eran asignadas.

4.2.1.2 Mantenimiento y actualizaciones del Sistema SARA

En el proyecto Mantenimiento y actualizaciones del Sistema SARA (*Sistema de Administración de Recursos Automatizado*) las tareas desempeñadas fueron principalmente de mantenimiento a diversos sistemas productivos desarrollados con Java y J2EE. Curiosamente al sistema productivo SARA no le realice mantenimiento alguno y sin embargo, así se denominaba el proyecto al que fui reasignada una vez que mi asignación en ARCCobranza termino.

El sistema productivo *Administración de Medios Alternos de Pagos* (ADMINMAP) está encargado de recibir una serie de pagos de créditos que los acreditados realizaban a través de supermercados como Soriana, Chedrahui, WalMart o por medio de otras instituciones autorizadas por el INFONAVIT, ya sea por tarjeta de crédito o por transferencia bancaria. La recepción de pagos se realizaba por medio de archivos de texto plano que dichas instituciones enviaban al INFONAVIT vía FTP a una ruta destino conocida por ADMINMAP.

Para dicha aplicación, desarrollé un módulo de carga automática de dichos pagos de los centros comerciales así como de otro módulo de consulta del historial de archivos procesados por el sistema ADMINMAP. Para estos dos componentes emplee como herramientas de desarrollo la plataforma Struts (una herramienta J2EE para construir aplicaciones WEB), el IDE WebSphere Developer Studio, el servido de aplicaciones WebSphere 6.0, FTP y algunas otras utilerías como Cronos (administración de hilos de ejecución o *threads*).

Con el tiempo, agregué un pequeño módulo a ADMINMAP para que notificara a los responsables de las instituciones autorizadas para recibir los pagos de los acreditados vía email, el procesamiento exitoso o no exitoso de sus archivos. Para ello empleo Java Mail, una API Java para el manejo de emails.

Tuve otras responsabilidades menores con otro sistema productivo, SISTEC, en el cual únicamente yo revisaba algunos procedimientos almacenados y efectuaba modificaciones muy pequeñas a algunas clases. Sinceramente ya mi participación en éste último proyecto con el INFONAVIT fue breve y limitada.

Para finalizar, una limitante constante dentro el instituto era el acceso restringido a las descargas de software desde Internet, casi todo el software libre no podía ser accedido de para su descarga porque el área de infraestructura no lo permitía aún siquiera al área de sistemas. Eso siempre represento un desafío porque la espera para obtener el software a emplear era considerablemente larga.

Mi ciclo profesional dentro de IDS Comercial había terminado y comenzó uno nuevo en otro cliente del ámbito gubernamental, el Sistema de Administración Tributaria SAT de la Secretaría de Hacienda y Crédito Público.

4.2.2 Information Specialist (Programador Java *Senior*)

Cliente: **Hewlett Packard (HP) – Servicio de Administración Tributaria (SAT)**
(Enero 2009 - a la fecha)
Empresa consultora: **Randstad México**

Randstad es una empresa transnacional de origen holandés dedicada principalmente al reclutamiento de recursos humanos para diferentes actividades empresariales, entre ellas la consultoría en tecnología informática. En este caso, el cliente al que le está brindando consultoría IT externa es a Hewlett Packard de México, que es una empresa trasnacional estadounidense dedicada entre otras cosas a los servicios empresariales IT (*HP Enterprise Serivices*).

Actualmente, HP tiene entre sus cliente al Servicio de Administración Tributaria (SAT)[10], un órgano descentralizado de la Secretaría de Hacienda y Crédito Público cuya misión principal es recaudar las contribuciones federales y controlar la entrada y salida de mercancías del territorio nacional a través de las aduanas.

HP, por medio de su división de Enterprise Services, está proporcionando al SAT una modernización tecnológica de todos sus sistemas aduanales. A éste proyecto se le conoce como *Modernización Tecnológica de Aduanas* o *MTA*.

MTA es un proyecto dividido en diferentes olas (también conocidas como fases); dichas olas o etapas tienen por objetivo la migración de las diferentes aplicaciones empresariales del SAT con las que opera actualmente en todas las aduanas del país.

[10] Sistema de Administración Tributaria. Ver bibliografía al final del documento.

Yo fui asignada para la etapa inicial, Ola 1, cuyo objetivo es la migración tecnológica de tres sistemas: **SICOCA** (*Sistema de Control de Catálogos* y programado en Visual Basic), **SCAAA** (*Sistema de Control de Agentes y Apoderados Aduanales* y programado también en Visual Basic) y **SOIA** (*Sistema de Operación Integral Aduanera*, programado en .NET).

Con el proyecto, se desea que toda la lógica de negocios de las aplicaciones SCAAA, SICOCA y SOIA se traslade servicios Web en Java y que los flujos de trabajo en Documentum reflejen todas las actividades que los usuarios de los sistemas legados han venido desempeñando con los mismos.

Inicialmente me integre al equipo de construcción de Ola 1, pero debido al atraso en que se encontraba el equipo de diseño, todos los programadores destinados al desarrollo se volcaron a terminar de generar los artefactos de diseño requeridos tales como: diagramas de clases y de secuencia; generación de objetos Java como Fachadas, Comandos, DAO's; creación de documentos con las especificaciones de interfaz de usuario (EIU) y generación de documentos de revisión de calidad como los WPR (Work Process Review).

Las plataformas empleadas en la migración tecnológica de Ola 1 son las siguientes:

Documentum

Es un administrador empresarial de contenidos, quien concentra las tecnologías, estrategias, los métodos y las herramientas utilizadas para capturar, administrar, almacenar, preservar y entregar el contenido de los documentos relacionados a alguna organización. Cuenta con un modulo BPM (Bussiness Process Management) pero no lo suficientemente robusto como para considerarlo un verdadero workflow.

Message Broker

Es un componente de software que actúa como un hubo concentrador central que puede recibir mensajes de diferentes destinatarios, determinar el destino de cada mensaje y reenviar el mensaje al receptor indicado.

Axis2

Es un motor SOAP (protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP y basado en XML) / WSDL (Web Services Description Language - *Lenguaje de Descripción de Servicios Web basado en XML*) para definir y construir Servicios Web.

Ibatis

Es un medio de trabajo, de código abierto que se ocupa de la capa de Persistencia (se sitúa entre la lógica de Negocio y la capa de la Base de Datos). Puede ser implementado en Java.

Spring

Es un medio de trabajo liviano basado en la técnica Inversión de Control (IoC) y en una implementación de desarrollo según el paradigma de Orientación a Aspectos (AOP). Es un método de trabajo o framework porque define la forma de desarrollar aplicaciones J2EE, dando soporte y simplificando complejidad propia del software corporativo. Spring emplea la técnica de Inversión de Control (IoC) porque promueve el bajo acoplamiento a partir de la inyección de dependencias (DI) entre los objetos (relaciones). Y finalmente está orientado a Aspectos (AOP) porque presenta una estructura simplificada para el desarrollo y utilización de aspectos.

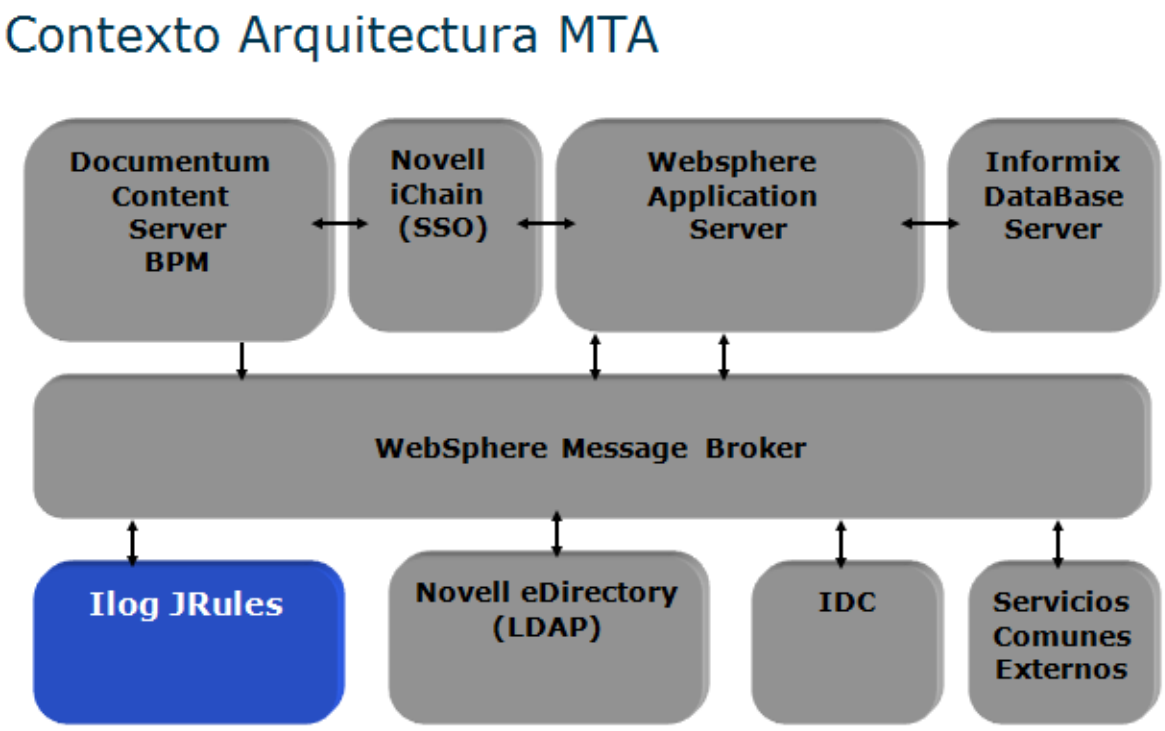
ILog

Es un sistema de administración de reglas de negocio

Cognos

Es un software conocido como Business Intelligence (BI) o Inteligencia de Negocios quien entre sus muy diversas bondades, permite la generación de reportes en formatos PDF, Excel y texto plano, entre otros.

Figura 4.11 La *Arquitectura para MTA OLA 1*



Como herramientas de desarrollo se están empleando:

- **Servicios Web**
  IDE Eclipse, Axis2, servidores de aplicaciones Tomcat y JBoss
- **Frameworks**
  Spring, IBatis
- **Message Broker**
  IBM MessageBroker
- **Reglas de Negocio**
  IBM Ilog
- **Flujos y Formas**
  Documentum Process Builder y Documentum Forms Builder
- **Reportes**
  IBM Cognos
- **Acceso a las aplicaciones legadas SCAAA y SICOCA**
  VMWare Player y Visual Basic 6.0

Una vez terminada la etapa de diseño, se inicio con la fase de construcción de servicios Web así como con los flujos y formas en Documentum.

El desarrollo de los servicios para el Message Broker no presento mayores problemas ni tampoco la definición de reglas de negocio con ILog.

El desarrollo de los servicios Web se llevo a cabo sin tantos contratiempos tecnológicos pero si de negocio pues el análisis y el diseño que se hizo de los servicios no fue correcto ni completo. Se ha tenido que invertir tiempo extra para analizar el código fuente de las aplicaciones legadas así como las reglas de negocio junto con los expertos del mismo para definir con mayor precisión, el comportamiento que deben seguir cada una de las operaciones realizadas por los servicios Web.

Figura 4.12 ***Comando ConsultaEstadosRepublicaCommand, empleado en la operación consultarEstadosRepublica del servicio Web Catalogo***

```
package mx.gob.sat.aduanas.serviciosnegocio.comunes.catalogo.command;

import java.util.ArrayList;
import java.util.List;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.ICommand;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.catalogo.dao.ICatalogoDAO;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.catalogo.dto.ConsultaEstadosRepublicaRequest;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.catalogo.dto.ConsultaEstadosRepublicaResponse;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.utilerias.DatabaseContextHolder;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.utilerias.DatabaseNames;
import mx.gob.sat.aduanas.serviciosnegocio.comunes.utilerias.assemblers.DTOAssembler;
import mx.gob.sat.aduanas.serviciosnegocio.entidad.figuraaduanal.comunes.EntidadFederativa;
import mx.gob.sat.aduanas.util.exception.GenericException;

public class ConsultaEstadosRepublicaCommand implements ICommand {

    /** Inyección del DAO vía Spring*/
    private ICatalogoDAO consultaEstadosRepublicaDAO;
```

```
	private DatabaseNames targetDatabase;

	/**
	 * Define el valor de la propiedad <code>targetDatabase</code>.
	 *
	 * @param targetDatabase valor a asignar a la propiedad <code>targetDatabase</code>.
	 */
	public void setTargetDatabase(DatabaseNames targetDatabase) {
		this.targetDatabase = targetDatabase;
	}

	/**
	 * Método para que Spring inyecte un valor a consultaEstadosRepublicaDAO
	 *
	 * @param consultaEstadosRepublicaDAO Dao a inyectar
	 */
	public void setConsultaEstadosRepublicaDAO(ICatalogoDAO consultaEstadosRepublicaDAO) {
		this.consultaEstadosRepublicaDAO = consultaEstadosRepublicaDAO;
	}

	/**
	 * Método que regresa en un mensaje las entidades federativas
	 * @param request Mensaje con la clave del estado de la republica a buscar.
	 * @return ConsultaEstadosRepublicaResponse con el arreglo de entidades federativas
	 * @throws GenericException Propaga una excepcion si ocurre un error de Sistema o de
Aplicacion
	 */
	public Object execute(Object request) throws GenericException {
		final ConsultaEstadosRepublicaResponse response =
			new ConsultaEstadosRepublicaResponse();
		String clave = ((ConsultaEstadosRepublicaRequest) request).getClave();

		final EntidadFederativa entidadFed = new EntidadFederativa();
		entidadFed.setClave(clave);
		entidadFed.setEstado(true);

		DatabaseContextHolder.setDatabaseName(targetDatabase);

		final List<EntidadFederativa> entidadesFed =
		consultaEstadosRepublicaDAO.consultarEstadosRepublica(entidadFed);
		final List<mx.gob.sat.aduanas.mdc.catalogos.EntidadFederativa> entidadesFedMDC =
			new ArrayList<mx.gob.sat.aduanas.mdc.catalogos.EntidadFederativa>();

		if(entidadesFed != null && entidadesFed.size() > 0){
			DTOAssembler.copiarLista(entidadesFed, entidadesFedMDC,
			mx.gob.sat.aduanas.mdc.catalogos.EntidadFederativa.class);
			// Construir el reponse de acuerdo al objeto mdc
			response.setEstados(entidadesFedMDC.toArray(new
			mx.gob.sat.aduanas.mdc.catalogos.EntidadFederativa[entidadesFedMDC.size
			()]));

		}else{
			// Construir el reponse de acuerdo al objeto mdc
			response.setEstados(null);
		}

		return response;
	}
}
```

Como se puede observar, esta clase implementa el patrón de diseño Command (ver que implementa la interfaz *ICommand* y el método *execute;* recibe un objeto *request* y regresa una respuesta dentro de un objeto *response*). Emplea otro objeto que a su vez implementa otro patrón de diseño para acceder a los datos de la base de datos: el DAO *CatalogoDAO*. Este DAO se inyecta a través de Spring y se configura ésta acción en el archivo *applicationContextCommandCatalogo.xml*. Ver la figura a continuación:

Figura 4.13 ***Archivo applicationContextCommandCatalogo.xml***

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
"http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans default-lazy-init="true">
	<!-- Beans de Catalogos -->
	<bean id="consultaEstadosRepublicaCommand"
	class="mx.gob.sat.aduanas.serviciosnegocio.comunes.catalogo.command.ConsultaEstadosRepu
blicaCommand">
		<property name="consultaEstadosRepublicaDAO" ref="catalogoDAO" />
		<property name="targetDatabase" value="SICOCA"/>
	</bean>
</beans>
```

La forma en que trabaja la operación *consultarEstadosRepublica* y en general todas las operaciones de los servicios Web empleados en MTA se puede observar con la siguiente descripción:

1. Se invoca al método **consultarEntidadFederativa** de **ICatalogoService**, enviando como parámetro un objeto de tipo **ConsultaEstadosRepublicaRequest**.
2. Se invoca el método **consultarEntidadFederativa** de **CatalogoServiceSkeleton** enviando como parámetro un objeto de tipo **ConsultaEstadosRepublicaRequest**.
3. Se invoca el método execute de **ConsultaEstadosRepublicaCommand** enviando como parámetro un objeto de tipo **ConsultaEstadosRepublicaRequest**.
4. Se obtiene una instancia de **CatalogoDAO**
5. Se obtiene de **ConsultaEstadosRepublicaRequest** la clave de la entidad federativa a consultar.
6. Se invoca al método **consultarEstadosRepublica** de la instancia obtenida de **CatalogoDAO** enviando los parámetros obtenidos. Dicho método retorna un objeto de entidad de **EntidadFederativa**.
7. Se convierte el objeto de **EntidadFederativa** obtenido, a un objeto de **EntidadFederativa** del modelo de datos canónico.
8. Se crea el **ConsultaEstadosRepublicaResponse**
9. Se establece el array convertido.
10. Posteriormente el **ConsultaEstadosRepublicaResponse** es regresado hasta el cliente.

El problema mayor en el proyecto se está presentando en Documentum, pues no es una herramienta flexible que permita dotar a los flujos o formas de comportamientos robusto y dinámico que si brindaría cualquier otra plataforma destinada a la capa Web de una aplicación empresarial. Tanto las formas como los flujos en Documentum pueden corromperse en cualquier momento, y es en ocasiones difícil recuperarlos, por lo que no en pocas ocasiones se han tenido que volver a construir dichos componentes.

Documentum es una herramienta limitada que no permite agregar comportamientos adicionales a sus pantallas con Javascript; por ejemplo, no se pueden agregar ventanas pop up, no se pueden crear nuevos botones, no se pueden atrapar los eventos de cambio de foco ni se pueden agregar hojas de estilo en cascada. Como se pueden imaginar, la presentación y la navegación a través de las formas de Documentum, no presenta una experiencia visual que dé la impresión de rapidez y dinamismo que se espera siempre de una aplicación web.

Figura 4.14 ***Ejemplo de una Forma en Documentum***

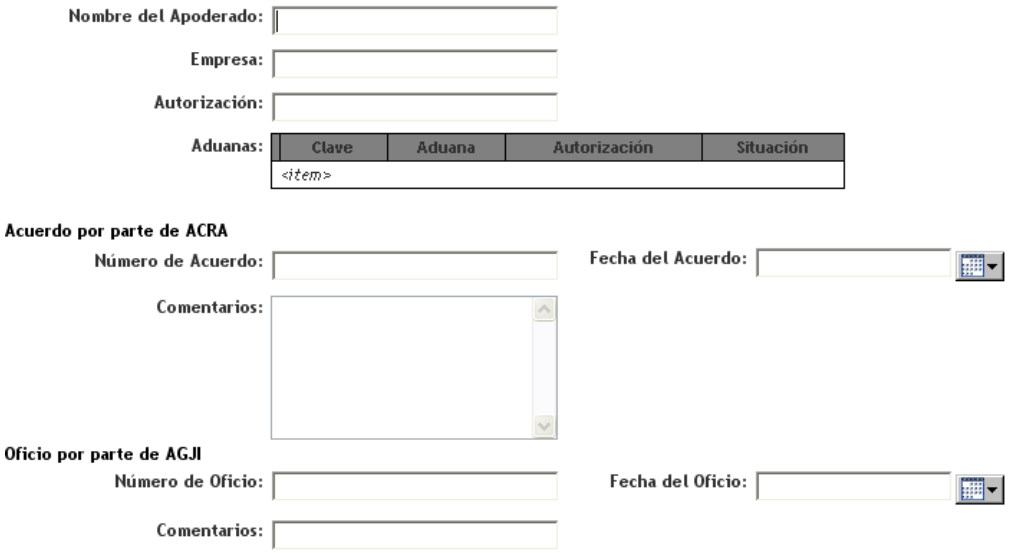
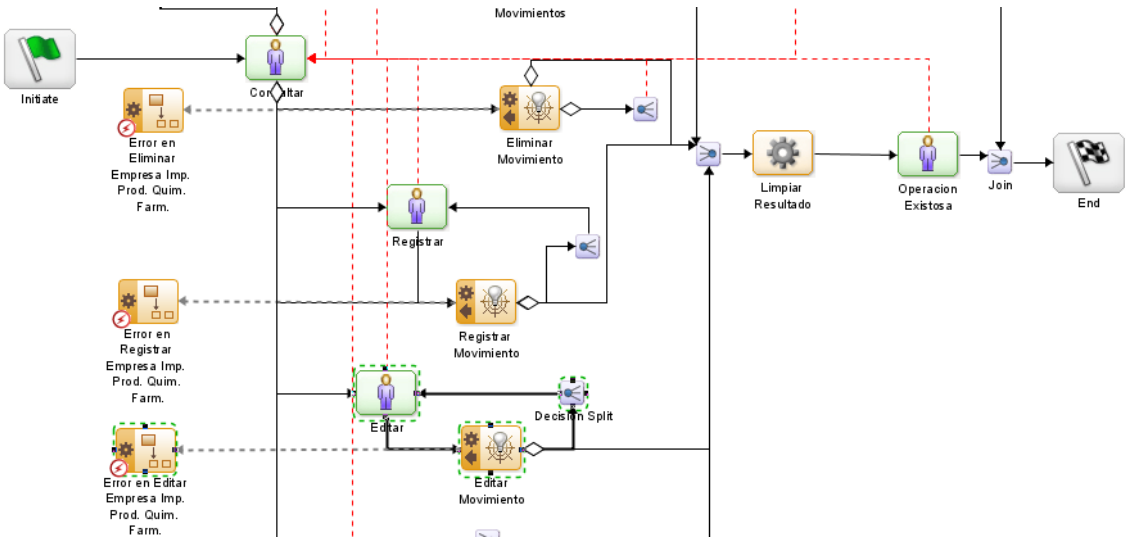


Figura 4.14 ***Ejemplo de un flujo en Documentum***



Otra desventaja que se tiene con los flujos es la invocación de los servicios Web. Si las definiciones de los mensajes de solicitud (request) de una operación tienen una profundidad mayor a cinco, Documentum simplemente no reconoce el mapeo de datos.

Para solucionar algunos de estos problemas con Documentum, el equipo de desarrollo ha creado componentes Java que permiten mapear los request de los servicios Web sin importar la profundidad que tengan; también ha creado componentes Java que se ejecutan cuando se envían los datos de la forma, como validaciones de datos o de disponibilidad de servicios. Sin embargo, ni con todos los componentes anteriores el dinamismo de las formas en Documentum es insuficiente.

A mi parecer, la decisión de utilizar Documentum como la interfaz de usuario en Ola1 fue equivocada, ya que está implicando desde hace meses atrás una cantidad de esfuerzo técnico y humano sin precedentes. Se invierte tiempo en resolver detalles que con otras tecnologías ni siquiera se presentarían.

Como programador en este proyecto, me he enfrentado a diferentes desafíos (principalmente con Documentum y con los servicios Web) y como lo he mencionado reiteradamente a lo largo de del capítulo tres, el retrabajo causado por imposiciones tecnológicas, deficiencias en el análisis y diseño del sistema así como la mala planeación de actividades para el equipo de trabajo, nunca ha sido agradable.

El aprendizaje tecnológico en Ola 1 ha sido grande porque hasta ahora no había empleado la mayoría de las tecnologías aquí utilizadas. El conocimiento adquirido en MTA me será de gran utilidad en futuros proyectos que requieran utilizar dichas herramientas.

En noviembre del 2009, tome a mi cargo uno de los equipo de desarrollo de Ola 1. Desde el inicio, mis actividades cambiaron de la programación de componentes a la administración de pruebas de software y a la organización de actividades del equipo de trabajo.

En resumen, las actividades que he desempeñado desde entonces, pueden listarse de la manera siguiente:

- Conocer a fondo los acuerdos y lineamientos de diseño y construcción definidos para el proyecto.
- Conocer las reglas de negocio determinadas para los flujos pertenecientes al equipo.
- Conocer la operación de cada uno de los flujos del equipo tanto en las aplicaciones legadas como en los casos de uso.
- Comunicar al equipo de trabajo los acuerdos a los que llegan en las juntas o reuniones de los líderes del proyecto (planes de acción, prioridades, etc.).
- Conocer los tiempos de entrega de los componentes en desarrollo al ambiente de pruebas para su evaluación.

- Delegar responsabilidades a los miembros del equipo para que desarrollen o realicen correcciones de defectos, según corresponda.

- Vigilar que las responsabilidades delegadas se cumplan en tiempo y forma por los elementos del equipo.

- Verificar que los componentes entregados por el equipo cumplan con los requerimientos solicitados.

- Notificar a los jefes inmediatos cualquier contratiempo que se presente en las actividades desempeñadas por el equipo (dificultades técnicas, tecnológicas, de infraestructura o de negocio).

- Escalar a los líderes del proyecto los comentarios e inconformidades expresadas dentro del equipo de trabajo.

- Canalizar a los jefes inmediatos todas las actitudes negativas presentes en los miembros de trabajo que no tienen disponibilidad para cooperar en las actividades asignadas.

- Negociar con los miembros del equipo de vez en cuando permisos a cambio de tiempo o esfuerzo extras para cumplir con los objetivos planteados en el proyecto.

Personalmente, la gestión de los recursos humanos es el aspecto más difícil que me ha tocado enfrentar ahora como líder de un equipo de trabajo. Cada persona posee una personalidad y una forma de trabajo diferentes y en ocasiones, es difícil manejar todas las diferencias que se presentan en el equipo de forma adecuada.

Y finalmente puedo comprender el sentir de mi equipo de trabajo porque el proyecto con sus deficiencias no representa más, una experiencia enriquecedora para su crecimiento laboral y si más una experiencia agotadora y aburrida.

Ser líder no es una tarea fácil para mí, sin embargo, ha resultado ser una experiencia que me ha brindado un gran aprendizaje y me ha hecho crecer profesionalmente no solo como programador sino también como persona.

CONCLUSIONES

El lenguaje de programación Java junto con la plataforma J2EE son dos de las herramientas más empleadas en el desarrollo de aplicaciones empresariales que deben cumplir hoy en día con las exigencias de seguridad, confiabilidad, escalabilidad y robustez que el manejo de información de las instituciones requiere. Las tecnologías Java y J2EE son ampliamente utilizadas en el desarrollo de software para diversas instituciones en el mundo entero y cuentan con un amplio número de seguidores, por lo que es mucho más fácil encontrar soluciones a diversos problemas a lo largo de Internet.

El uso de buenas prácticas de programación tanto en Java como en J2EE permiten desarrollar y entregar componentes de software bien programados, escalables y de fácil mantenimiento, características que resultan ser de gran ayuda para todos aquellos programadores que heredarán los componentes que una programa de forma inicial pero que por diversas razones no continua dentro del proceso de mantenimiento de los mismos. La programación bajo estándares como los patrones de diseño permite implementar soluciones ya probadas a los nuevos componentes desarrollados.

Durante toda mi experiencia profesional, he enfrentado desafíos como programador y en los últimos tiempos, como líder. Todas esas dificultades me han hecho crecer profesionalmente y han puesto a prueba mis habilidades técnicas y administrativas para encontrar soluciones satisfactorias acordes a las necesidades de los proyectos. Cabe mencionar que la administración de recursos humanos ha sido el desafío más complicado que he tenido en mi trayectoria laboral por la gran diversidad de personalidades que se reúnen dentro de un equipo de trabajo. Sin embargo, ha sido una experiencia enriquecedora que ha complementado mi visión dentro de la administración de proyectos de desarrollo de software.

Así, cada proyecto dentro de mi experiencia laboral ha resultado siempre en un aprendizaje técnico y en los últimos tiempos, en un aprendizaje administrativo (como lo mencione anteriormente); todos han sido de una u otra manera exitosos para mí por todas las enseñanzas que me han dejado ya fuese como programador o como líder. Me han permitido mejorar mis conocimientos de programación, de pruebas de software y de administración de proyectos.

Personalmente, resumiendo toda la experiencia profesional adquirida a través de estos cinco años en el sector público y privado y en los diversos proyectos mencionados en el último capítulo, me ha mostrado en repetidas ocasiones que el seguimiento de un proyecto tiene que efectuarse tanto en el aspecto técnico como en el administrativo, por lo que es de suma importancia definir correctamente con el usuario desde el inicio los alcances del mismo con el fin de evitar re trabajos y retrasos en las entregas finales, así como conflictos personales dentro de los equipos de trabajo debidos al hartazgo de los miembros que lo integran. Puedo concluir que definitivamente adquirí una visión global del desarrollo de software en el ámbito profesional así como una panorámica de que hacer y que dentro de un nuevo proyecto.

BIBLIOGRAFÍA

Trail: Getting Started
http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

White Paper, The Java Language Environment
http://java.sun.com/docs/white/langenv/

Understanding Java Platform, Enterprise Edition
http://java.sun.com/javaee/6/docs/firstcup/doc/p4.html

Code Conventions for the JavaTM Programming Language
http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Ingeniería de Software
http://cysingsoft.wordpress.com/2009/06/23/modularidad-cohesion-y-acoplamiento-carlos-fontela/

Diseño de Software Orientado a Objetos
http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

Core J2EE Patterns: Patterns index page
http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html

Sistema de Administración Tributaria
http://www.sat.gob.mx/

EMC Documentum
http://www.emc.com/domains/documentum/index.htm