UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DESARROLLO WEB CON TECNOLOGÍA .NET

INFORME DE EJERCICIO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:
DAVID ROLANDO BETANCOURT NAVARRETE

ASESOR:
ERNESTO PEÑALOZA ROMERO

MÉXICO 2016

Desarrollo Web con tecnología .NET

INFORME DE EJERCICIO PROFESIONAL

DAVID ROLANDO BETANCOURT NAVARRETE

ÍNDICE

INTRODUCCIÓN

Nunca se deja de adquirir conocimiento, nunca terminamos realmente nuestra educación profesional, el ejercicio de nuestra carrera es una extensión de lo aprendido en la Universidad, en particular a la UNAM debemos agradecer que nos enseña a aprender por nosotros mismos.

El campo laboral de las Tecnologías de la Información es de por sí amplio y se multiplica su alcance al aplicarse a cualquier área de la investigación y el conocimiento.

A lo largo de este informe el lector descubrirá que he enfocado mi carrera profesional al desarrollo y la administración de sistemas en tecnología .NET; Microsoft tiene mucho renombre e inspira confianza en los clientes e igual que cualquier otra tecnología tiene ventajas y desventajas.

En los últimos años la tendencia en el desarrollo de aplicaciones, servicios y productos se ha orientado a la plataforma web, debido a que es accesible desde cualquier dispositivo y/o sistema operativo con acceso a la red, en el informe se apreciará que esta tendencia sigue a la alza y se comprobarán sus ventajas.

1) SECRETARÍA DE GOBERNACIÓN

Mi experiencia en el área de las tecnologías de la información comienza en 2006 realizando prácticas profesionales en la Secretaría de Gobernación. Estando cerca de egresar y con la necesidad de adquirir experiencia, ya que como es muy bien sabido, una vez en el mercado laboral, la experiencia es muy importante y yo (como muchos otros profesionistas), me ofrecí a ejercer medio tiempo por una paga nula a cambio de experiencia en el campo de la informática y las tecnologías de la información. Ingresé al Órgano Interno de Control de la Secretaría. Comencé realizando tareas, que pudieran considerarse sencillas, realizando inventario de equipo de cómputo, reparando algunos problemas de hardware, de conectividad y de sistema operativo, es decir, dando soporte técnico. Al cabo de unas semanas, ya no habiendo nada de equipo informático que inventariar, por fin me asignaron a un proyecto, el cual consistía en desarrollar un sistema en donde se pudiera administrar el inventario que previamente se había realizado. Era el momento de aplicar en conjunto, la investigación y los conocimientos obtenidos en la carrera, sin embargo, no tenía conocimiento alguno en las tecnologías que se proponían para dicho proyecto, estas eran en base de datos SQL 2000 y como lenguaje de programación Visual Basic 6, un lenguaje orientado a objetos con un entorno de desarrollo provisto por Microsoft, ambas orientadas a entornos operativos Windows. Durante mi preparación profesional tomé algunos cursos que abordaron tanto las herramientas como las tecnologías. Si algo brinda la educación otorgada por la Universidad es la capacidad de aprender a aprender. El conocimiento se encuentra en libros, internet, foros, etcétera y la experiencia se adquiere de la aplicación de él en el mundo real.

Un sistema dedicado al control de inventarios es algo relativamente sencillo y considerando que los registros de equipo de cómputo del edificio del OIC eran poco más de 300, realmente no implicaba una demanda de recursos de almacenamiento o ancho de banda. El diseño de la base de datos es relativamente sencillo y estuvo constituido por tres tablas que corresponden al equipo, su clasificación y su propietario. La tabla de equipo tenía dos llaves foráneas que la relacionan con la tabla de clasificación y la de propietarios. Las operaciones sobre las tablas debían permitir agregar, editar y eliminar registros para la tabla de equipos y sólo agregar y editar para las tablas de clasificación y propietario. El requerimiento decía que la aplicación debería funcionar específicamente en un ambiente Windows y sería únicamente instalado en un equipo, mientras que la base de datos sería montada en un servidor remoto. El entorno de desarrollo de Visual Studio 6.0 permite una fácil creación de formularios y establecimiento de la disposición de controles. Basado en Component Object Model permite el empaquetamiento de funcionalidades en componentes. Los formularios estaban compuestos por cuatro pantallas:

- Consulta, baja e impresión del equipo de cómputo. Permitiendo filtrar la información del inventario a través de los filtros de propietario, tipo de equipo, estados de utilidad y/o número de serie, la cual podía ser impresa utilizando integración con Crystal Reports.
- Alta y actualización de equipo al inventario. Permitiendo registrar o actualizar los datos de un equipo de cómputo, así como su estado de utilidad.
- Alta de propietarios. Permitiendo registrar a empleados del OIC que tuvieran equipo a su cargo.
- Parametrización. Permitiendo la modificación de la configuración principal del sistema, como cadena de conexión y los datos de los catálogos.

La programación no tenía una definición semántica de las clases y los métodos, desgraciadamente en esa época mi experiencia no era suficiente para saber que la organización semántica del código así como la documentación y el uso de nomenclaturas ayudarían a extender la vida de un sistema. Las consultas se hicieron directamente en “código duro”, suponiendo que el sistema nunca iba a ser modificado. A pesar de estos inconvenientes, mis superiores quedaron muy satisfechos con el resultado y sólo me pidieron la elaboración del manual de usuario. Quedando concluido así, mi primer sistema con aplicación en el mundo real.

Este proyecto se elaboró aplicando las bases de los sistemas cliente-servidor vistos en la carrera, definitivamente el proyecto tuvo deficiencias en documentación y estándares, como tal no hubo una metodología de trabajo, en parte por la inexperiencia y en parte por lo ajustado del tiempo en el que debía estar listo, esto debido a que los mandos del área no le dieron la importancia al alcance del proyecto, en específico no consideraron futuros cambios al sistema. A mi propia consideración, fue un proyecto escolar con aplicación real.

2) G4 SOLUTIONS

En 2007 me integré a una empresa de seguridad y control de accesos llamada G4 la cual también desarrollaba su propio software e integraba componentes de hardware para aplicaciones de propósito específico, trabajando por outsourcing, donde fui capacitado en la tecnología .NET, el framework de Microsoft para desarrollar aplicaciones, independientemente de la plataforma de hardware. Los principales componentes del framework son:

- El o los lenguajes de programación
- La biblioteca de clases base (BCL).
- El entorno común de ejecución para lenguajes (CLR).

Existe una norma para la infraestructura común de lenguajes (CLI), el desarrollo de lenguajes se facilita, por lo que el entorno de trabajo .NET soporta más de veinte lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje. Adicionalmente se pueden crear aplicaciones con ASP.NET para plataformas web, anteriormente Microsoft utilizaba una tecnología conocida como Active Server Pages (ASP) para la creación de sitios web dinámicos, después con la aparición del Framework .NET Microsoft fusionó las tecnologías denominándolo ahora ASP.NET, permitiendo ahora crear sitios web con páginas dinámicas utilizando cualquier lenguaje de los soportados por el entornos de trabajo de .NET. Las páginas de ASP.NET, conocidas oficialmente como "web forms", son el principal medio de construcción para el desarrollo de aplicaciones web. Los formularios web están contenidos en archivos con una extensión ASPX, estos archivos típicamente contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan del lado del servidor y Controles de Usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web.

ASP.NET sólo funciona sobre el servidor de Microsoft IIS, lo que supone una desventaja respecto a otros lenguajes del lado de servidor, ejecutables sobre otros servidores más populares como Apache.

En G4 se tenía un proyecto denominado “ArenaPremier” el cual consistía en un sistema web de administración de eventos, recintos y venta de boletos, debía estar dividido en componentes:

- Terminal punto de venta.
- Página web.
- Administración de parámetros y condiciones (BackOffice).
- Venta a través de mensajes SMS.

El avance del proyecto al momento de mi llegada era únicamente de la estructura de la base de datos, la cual se había elaborado en base a un proyecto anterior del mismo giro, de hecho, el proyecto pretendía ser la versión 2.0 de un sistema ya existente y cuya capacidad y extensibilidad ya habían sido superadas.

El primer paso consistió en documentar la base de datos, esta tarea incluía hacer casos de uso y diagramas de flujo para revisar y mostrar cómo se movería la información a través de las diferentes tablas al insertar, modificar y eliminar información de la misma. La base de datos tenía decenas de tablas, dichas tablas se agrupaban de manera semántica en:

- Tablas de usuarios, perfiles y permisos. Contenían la información sobre los usuarios que podían operar el sistema de BackOffice, aquellos que operaban taquillas y cualquier usuario en general. El usuario tenía asignado un perfil y además una serie de permisos, y cada perfil tenía una serie de permisos, de tal manera que a cada usuario en particular se le podían asignar y restringir características a pesar de su perfil. Los permisos consistían en una cadena de 1000 caracteres seccionada de acuerdo a grupos de permisos y cuyos valores sólo podían tener valores 0 o 1, por ejemplo, los permisos de la pantalla de usuarios eran una sección de cinco dígitos, la primera posición indicaba que la característica podía ser accedida por el usuario, en la segunda posición si este podía agregar nuevos usuarios, en la tercera si podía editar la información de usuarios existentes, en la cuarta si podía eliminar usuarios y en la quinta si podía imprimir los registros en un reporte.

- Tablas de parametrización. Estas tablas contenían la información sobre los parámetros de operación del sistema, el idioma de presentación del sistema, campos personalizados, etcétera.

- Tablas de franquicias. Se contenía la información relativa a las franquicias, es decir, los usuarios finales del sistema de BackOffice, los cuales a su vez podían controlar franquicias dependientes y de esta manera administrar todo lo relacionado a sus eventos, recintos, usuarios, permisos, perfiles, parametrización, etcétera.

- Tablas de eventos. En ellas se guardaba la información relacionada a los eventos, los detalles de los mismos, así como las imágenes asociadas al mismo, cada evento debía vincularse con un presentador/artista, con un recinto además de la configuración del recinto y la configuración de venta. También se guardaba en estas tablas la información relativa a la configuración de los mismos.

- Tablas de recintos. Contenían la información de los recintos disponibles para un evento, así como datos generales de los mismos, imágenes, sitios relacionados, etcétera. Estaba estrechamente relacionada con la configuración de los recintos.

- Tablas de configuración de recintos. Estas tablas almacenaban la información de las secciones físicas y lógicas de los recintos, es decir la disponibilidad de cada sección en cuanto a capacidad y numerabilidad en función de los probables eventos a llevarse a cabo.

- Tablas de catálogos de artistas y presentadores. En ellas se almacenaba la información relativa a los artistas, presentadores, equipos deportivos, así como su clasificación y demás información general.

- Tablas de operación de venta. Contenían toda la información de la configuración de ventas, el inicio y término de las mismas, la lógica de venta de los boletos, costos de los mismos, tipos de boleto, secciones de los recintos, preventas, las taquillas, tipos de taquillas, promociones y descuentos.

- Tablas de usuarios de página web y clientes preferenciales. Aquí se guardaba la información relativa a los usuarios de la página web del sistema y clientes frecuentes, se almacenaban sus preferencias y sus datos para que el usuario final pudiera permanecer en contacto y notificando a sus clientes sobre próximos eventos, descuentos, etcétera.

- Tablas de catálogos generales. Estas tablas contenían de manera centralizada, todas las imágenes, direcciones, teléfonos, y links utilizados en el sistema.

El diseño de la base de datos era bastante robusto y contemplaba extensibilidad sin embargo no estaba completo ni totalmente bien elaborado de acuerdo de acuerdo a las especificaciones de nomenclatura y eliminación en cascada del esquema. Una parte del tiempo se destinó a validar que las relaciones entre las tablas y las consecuentes eliminaciones en cascada, se estuvieran de acuerdo a lo planeado.

Al realizar este desarrollo simultáneamente se pretendía obtener la certificación nivel 2 de CMMI (Capability Maturity Model Integration), el cual es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

A través de la calificación del nivel de madurez en los procesos para el desarrollo del software, primero para determinar en qué nivel se encuentran los procesos y luego para definir las estrategias de mejora. El modelo está dividido en niveles del uno al cinco, siendo el uno el más bajo y donde no se ha implementado ningún proceso ni mejora. Cada nivel tiene bien definidas áreas de proceso para el manejo de cada parte de los mismos.
En el nivel dos, que se deseaba alcanzar, los roles son los siguientes:

- CM - Configuration Management (Gestión de la configuración). Su objetivo es establecer y mantener la integridad de los elementos de trabajo, identificando, controlando y auditando dichos elementos mediante:

    - La identificación de los elementos de trabajo que componen una línea base, es decir un conjunto de especificaciones o productos de trabajo.
    - Controlando los cambios y versiones de dichos elementos.

- Proporcionando formas de construir los elementos de trabajo a partir del sistema de control de la configuración.
- Mantener la integridad de las líneas base.
- Proporcionar información precisa de los datos de la configuración a desarrolladores y clientes.

- MA - Measurement and Analysis (Medición y Análisis). Tiene como objetivo desarrollar y sostener una capacidad de medición que sea usada para ayudar a las necesidades de información de la gerencia o dirección.
  Los datos tomados para la medición deben estar alineados con los objetivos de la empresa para proporcionar información útil a la misma.
  Se ha de implantar un mecanismo de recolección de datos, almacenamiento y análisis de los mismos de forma que las decisiones que se tomen puedan estar basadas en estos datos.
  Este sistema tiene que permitir además:

  - Planificación y estimación objetiva.
  - Comparar el rendimiento actual contra el rendimiento esperado en el plan.
  - Identificar y resolver problemas relacionados con los procesos.
  - Proporcionar una base para añadir métricas en procesos futuros.

- PMC - Project Monitoring and Control (Monitorización y Control de Proyectos). El objetivo de la monitorización y control de proyectos es proporcionar una compresión del estado del proyecto para que se puedan tomar acciones correctivas cuando la ejecución de proyecto se desvíe del plan.
  El documento del plan de proyecto es la base para monitorizar las actividades, comunicar el estado y tomar acciones correctivas. El progreso se determina comparando los actuales elementos de trabajo: tareas, horas realizadas, coste y calendario actual, con los estimados en el plan de proyecto. Una apropiada visibilidad permite tomar acciones correctivas antes de que el trabajo real se desvíe mucho del plan. Las acciones que se tomen, harán que tengamos que rehacer/ajustar nuestro plan de proyectos.

- PP - Project Planning (Planificación de Proyectos). Tiene como objetivo establecer y mantener planes definidos en las actividades del proyecto.
  Las tareas implicadas en la planificación de proyectos son:

- Desarrollar un plan inicial del proyecto.
- Establecer una relación adecuada con todas las personas involucradas en el proyecto.
- Obtener compromiso con el plan.
- Mantener el plan durante el desarrollo del proyecto.

  El plan incluye estimación de los elementos de trabajo y tareas, recursos necesarios, negociación de compromisos, establecimiento de un calendario, e identificación y análisis de los posibles riesgos que pueda tener el proyecto.
  El plan de proyectos es un herramienta de trabajo “viva” que se debe de actualizar con mucha frecuencia ya que los requisitos cambian constantemente, habiendo entonces que rehacer la estimación, algunos riesgos desaparecerán y surgirán nuevos, por lo tanto, habrá que tomar acciones correctivas.

- PPQA - Process and Product Quality Assurance (Aseguramiento de la Calidad). Proporciona personas y gestión con el objetivo de que los procesos y los elementos de trabajo cumplan los procesos. Cabe destacar que es más importante cumplir el proceso que la funcionalidad.
  Esto se consigue mediante:

    - Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios contra las descripciones de procesos, estándares y procedimientos.
    - Identificar y documentar los elementos no conformes.
    - Proporcionar información a las personas que están usando los procesos y a los gestores, de los resultados de las actividades del aseguramiento de la calidad.
    - Asegurar de que los elementos no conformes son arreglados.

  No suele darse mucha importancia a esta área del proceso, sin embargo un modelo de calidad no puede ser aplicado sin considerarla.

- REQM - Requirements Management (Gestión de Requerimientos). Consiste en administrar los requisitos de los elementos del proyecto y sus componentes e identificar inconsistencias entre estos requisitos, el plan de proyectos y los elementos de trabajo.
  En este proceso se deben de administrar todos los requisitos del proyecto, tanto los requisitos técnicos como los requisitos no técnicos.
  Estos requisitos han de ser revisados conjuntamente con la fuente de los mismos, así como con las personas que se encargarán del desarrollo posterior.

El equipo de desarrollo para el proyecto contaba solamente con cinco personas, siendo:

- Líder de proyecto
- Diseñador
- Desarrolladores (2)
- Documentador.

El proyecto llevaba un año de desarrollo solamente en el modelado de la base de datos, para cualquier empresa el costo del desarrollo, mantenimiento y soporte de un producto debería ser el óptimo, específicamente en un sistema tan robusto un error en alguna fase implicaría un alto costo de tiempo, dinero y recursos, por esta razón administrar los procesos de desarrollo y llevarlos por una mejora continua determinará la sustentabilidad y rentabilidad del proyecto.

Con base en el modelo de calidad se especificó una serie de normativas a seguir durante el desarrollo, con la finalidad de mantener un estándar que permitiera a futuros desarrolladores dar soporte y extensibilidad al sistema. De esta manera se creó una especificación de la nomenclatura a seguir a lo largo del desarrollo para todas las variables, constantes, enumeraciones, funciones, métodos, clases, “stored procedures”, vistas, etcétera. Las reglas eran las siguientes:

- Notación pascal.
- Hacer referencia al tipo de dato con los tres primeros caracteres en el caso de variables.
- Hacer referencia al tipo de dato devuelto con los tres primeros caracteres en el caso de funciones.
- Utilizar nombres descriptivos en idioma Inglés.

Posteriormente se elaboró un diccionario de la base de datos, divido en dos partes. La primera especificando de manera alfabética el detalle de cada tabla, indicando los campos que la componían y de estos el tipo de dato, nulidad, longitud en caso de aplicar y su función, mientras que la segunda parte contenía diagramas de los diferentes grupos de tablas y sus relaciones indicando la interacción con los demás grupos de tablas.

El lenguaje de programación que se utilizaría a lo largo del proyecto estaba constituido por dos opciones, Visual Basic .NET y C#, aunque ambos lenguajes tienen diferencias menores en el entorno de desarrollo de Visual Studio, la afinidad que tenía el equipo de trabajo con VB.NET y características específicas tales como que no es sensible entre mayúsculas y minúsculas, enlace automático de los eventos a sus métodos manejadores, entre otros. Determinaron que sería mejor utilizar VB.NET. Se especificó que para el desarrollo debía utilizarse lenguaje Visual Basic en el marco de trabajo del Framework .NET 2.0 de Microsoft y como gestor de versiones se utilizaría, Microsoft Visual SourceSafe.

Visual Basic .NET es un lenguaje de programación orientado a objetos, que deriva de las versiones antiguas de Visual Basic y que no estaban integradas al Framework .NET, el lenguaje tiene varias características que facilitan la programación como el enlace automático de los eventos a través de la palabra reservada Handles, las variables pueden ser declaradas utilizando el constructor WithEvents y escoger fácilmente el método al cual se relacionara la variable, Visual Basic no es sensible entre mayúsculas y minúsculas, entre otras.

Microsoft Visual SourceSafe es un programa que gestiona versiones de los elementos un proyecto. Entre las principales y deseables características de un “versionador” se encuentran:

- Gestionar un historial para cada documento del sistema, permitiendo con esto recuperar versiones estables de cualquier característica de la funcionalidad.
- Permitir al administrador del proyecto hacer visibles sólo determinados archivos, documentos y/o carpetas a sus recursos dependiendo el perfil o las tareas que tengan los mismos.
- Incluir un analizador de documentos que permita integrar cambios cuando se haya trabajado de manera “offline” o por cuando por alguna razón dos o más recursos trabajaron simultáneamente sobre el mismo archivo o documento integrando diferentes características de la funcionalidad.

A lo largo del desarrollo de un sistema el control de versiones se torna importante: la liberación de una versión estable de un sistema no implica que carezca de errores, adicionalmente resulta muy usual que el cliente decida hacer cambios al diseño o a las funcionalidades del sistema, es por estas razones que tener un historial de cada versión resulta de utilidad. Es muy común también que el cliente al ver el cambio prefiera regresar a lo anterior, tener una herramienta de control de versiones evita el retrabajo y sobre todo permite la gestión de los componentes del proyecto de manera cronológica. En ocasiones se integran o modifican componentes del sistema de los cuales no se puede saber con certeza la manera en que afectarán a otros elementos del mismo, ante la incapacidad de simular el ambiente productivo en prueba, por ejemplo el “performance”, aventurarse a cambiar el código sin respaldarlo muy probablemente desemboque en dolores de cabeza y retrabajo, de hacerse una copia de seguridad el no tener un repositorio de las mismas podría terminar en el extravío del código, es en escenarios como estos en donde un software gestor de versiones que centraliza la gestión del código y el diseño no solamente facilita el desarrollo sino que se vuelve una pieza fundamental del mismo.

Finalmente, para desarrollar también se necesita una metodología, es decir una estructura en la manera de planificar, desarrollar y verificar los objetivos del proyecto. Se decidió utilizar el modelo en espiral porque algunos requerimientos no estaban definidos plenamente y sería necesario en algún momento redefinirlos, el modelo en espiral es iterativo, combinado con las características sistemáticas del modelo en cascada permite hacer liberaciones de versiones del mismo producto y un refinamiento por cada iteración. También permite manejar los riesgos tanto técnicos como de administración del proyecto.

El modelo está basado en el continuo refinamiento iterando a través de la definición y análisis de requerimientos, el diseño del sistema y de software y la implementación. En cada iteración del ciclo estos sistemas se convierten es una extensión de sus antecesores. Este modelo utiliza mucho de las mismas fases del modelo en cascada, en esencia el mismo orden, separado por planeación, manejo de riesgos y la construcción de prototipos y pruebas.

Los documentos son producidos cuando son requeridos y el contenido refleja la información necesaria de cada punto en el proceso.

La idea es tener una línea continua de productos listos y disponibles para la revisión por parte del usuario. El ciclo de vida en espiral permite agregar elementos al sistema cuando estos estén disponibles, de esta manera se asegura que no haya conflicto con requerimientos anteriores, también obliga al usuario a envolverse en el desarrollo del sistema desde etapas tempranas.

- Cada iteración gira alrededor de una serie de áreas:
- Determinar objetivos, alternativas y restricciones de la nueva iteración
- Evaluar alternativas e identificar y resolver posibles riesgos
- Desarrollar y verificar el producto para la iteración actual
- Planear la siguiente iteración

El desarrollo comenzó utilizando el modelo de programación por capas, utilizando como base las siguientes:

- Capa de datos. Conformada por la base de datos, es decir, las entidades que permiten el almacenamiento, además de las entidades que permiten administrar, procesar y en general manipular la información, como Stored Procedures, Triggers, Vistas, etcétera.
- Capa de negocio. Se refiere a todos los componentes que llevan la lógica del sistema, es decir, a la abstracción de los objetos y sus interacciones. Esta a su vez está estructurada por dos capas, donde una era representada por una única clase base la cual contenía los métodos y funciones necesarios para la interacción con la capa de datos. A partir de esta clase, se heredaban todas las otras representaciones abstractas de las entidades del sistema.
- Capa de presentación. Es la interfaz con la que usuario interactúa, así como las salidas que el usuario final obtendrá y donde la información debe acoplarse para que sea entendible para él mismo.

La idea del avance del proyecto estaba enfocada en desarrollar paralelamente (por parte del diseñador) la interfaz gráfica con las funcionalidades para moverse a través de las pantallas y de nuestra parte (los desarrolladores), elaborar la integración de las capas de datos y de negocios, para que al finalizar éste, se desarrollara solamente la integración con la interfaz.

Consideré que la mejor opción para el manejo de los objetos era a través de una única clase que mantuviera el control de todos los accesos a la base de datos, esto fue debido a que usualmente al programar, se suele reutilizar código, sobretodo tratándose de instanciar objetos, abrir conexión, ejecutar sentencias, cerrar conexión, etcétera.

Al generar componentes no sólo facilitamos el desarrollo y reducimos el número de líneas de código, sino que además se empaquetan recursos de software que pueden conectarse y/o utilizarse en otros proyectos.

La integración de las capas de datos y de negocio se dividió en dos partes, se desarrolló una clase base que era el medio de comunicación con la base de datos y a partir de la cual todas las demás debían heredar. Esta clase llamada dbf (DataBaseFramework) contenía los métodos para interactuar con la base de datos, de manera que en las capas superiores no fuera necesario repetir código.

- Se diseñó para crear instancias usando como parámetros la cadena de conexión como string, un objeto de tipo System.Data.SqlClient.SqlConnection y cada una de las formas con una sobrecarga para indicar con un valor booleano si iniciar junto con la instancia una nueva transacción.
- Contenía funciones y métodos con los que se podía ejecutar sentencias SQL directamente, así como Stored Procedures y opcionalmente se podían asignar los parámetros SQL a través de una colección.
- Permitía manejar transacciones desde la capa de negocio.
- Integraba la interfaz IDisposable con la que se podían manualmente liberar de memoria las instancias creadas.

Con la finalidad de mantener una estructura organizada de las clases y que de cierta manera se asemejase lo más posible a las entidades de la base de datos que les proporcionarían la información se elaboró una distribución semántica de las clases en namespaces, esto sería muy práctico al momento de manejar excepciones, realizar modificaciones y documentar. Posteriormente se desarrolló el plan de proyecto, y al cabo de unas semanas las primeras pantallas estaban listas para ser integradas con la capa de negocio. También era necesario realizar cambios a la base de datos y crear nuevos objetos pues conforme fue avanzando el desarrollo surgieron nuevos problemas que no se habían contemplado en el análisis inicial. Aquí ocurrió el primer retraso y por consecuencia el primer ajuste al plan de proyecto, puesto que en su versión inicial no se había contemplado un tiempo definido para el desarrollo del código necesario para la integración de la capa de negocio con la de presentación.

El desarrollo continuó con los tiempos más ajustados, como casi todo proyecto de desarrollo, y se fueron integrando las pantallas, la parte del proyecto que representó el mayor reto fue la programación de la lógica de negocio sobre las configuraciones de ventas. La solución a este problema consistió en crear una tabla que fungía como una matriz multidimensional y donde la precedencia era la siguiente:

- Sección a la venta, tipo de boleto, impuestos/cargos, promociones, descuentos.

De esta manera se cubrían todas las combinaciones posibles. En la capa de negocio se manejaba como un objeto DataTable, el verdadero problema se presentó al integrar la interfaz gráfica, la cual no había contemplado la estructura, y ésta se desarrolló a través de un GridView donde por cada renglón se tenía una combinación de todas las posibles y el usuario mediante controles checkbox decidía si esa aplicaba o no aplicaba, sin embargo, para recintos con muchas secciones a la venta y tipos de boleto, la selección se llegaba a ser tediosa para el usuario además de que el rendimiento se vea afectado al momento de guardar la información. Posteriormente se presentó la pantalla en una de las juntas de seguimiento con los directivos de la empresa y no les agradó, ellos preferían un control de tipo árbol pues les parecía más intuitivo y comprensible. Lamentablemente la versión especificada del .NET Framework no contenían el control, el cual se incluiría en versiones posteriores, por lo que se buscó primero encontrar algún control provisto por algún otro desarrollo o empresa para integrarlo al proyecto, sin embargo los costos no fueron aceptables para el presupuesto del proyecto. Entonces se optó por construir nuestro propio control árbol utilizando únicamente HTML y JavaScript. Al elaborar el código para la funcionalidad del mismo, me encontré con que se requerían muchos recursos de memoria pues los controles se elaboraban en tiempo de ejecución y cada combinación se presentaba al usuario de manera individual, esto quiere decir que el usuario solamente podía ver desglosado un nodo y sus nodos hijos a la vez.

Los elementos que conformaban los puntos de venta fueron los que tomaron más tiempo y más pruebas, se elaboró un carrito de compras para la taquilla, se integró un monitor donde el cliente del usuario final podía escoger sus asientos tocando la pantalla, se integró el cobro mediante tarjetas de crédito, se elaboró una interfaz para la creación de plantillas de boletos y se utilizaban impresoras térmicas al momento de la venta.

El punto de venta fue un desarrollo web donde los usuarios registrados en el BackOffice accedían, el sistema verificaba la autenticación relacionando al usuario con la dirección IP del punto de venta, una vez iniciada la sesión el sistema mostraba una pantalla para configurar tipos de cambio para monedas extranjeras (en el caso de aplicar), posteriormente el sistema verificaba la disponibilidad de boletos, físicamente, es decir el usuario capturaba el folio inicial y final de los lotes de boletos en blanco con los que contaba la taquilla para llevar un control de las ventas registradas en el sistema contra la emisión de boletaje. Finalmente estaba el carrito de compras donde el usuario a través de "touchscreen" o atajos en el teclado seleccionaba la sección, en seguida seleccionaba el tipo de boleto y la cantidad de boletos a comprar, una vez que se aceptaba el sistema pasaba a una pantalla de "checkout" donde se especificaba el tipo de pago y el cambio a devolver si era el caso, una vez procesados los boletos se mandaban a impresión y se retornaba a la pantalla de carrito de compras. Al finalizar la sesión, el sistema pedía al usuario capturar el balance de su caja y lo comparaba con el balance de las ventas de sí mismo y emitía en un boleto un pequeño reporte de las ventas de esa sesión.

El proyecto se encontraba en un promedio del 90% de desarrollo cuando se puso a prueba en un ambiente productivo. El sistema se habilitaría de forma preliminar en tres taquillas de los puntos de venta del club Jaguares de Chiapas, franquicia que en ese momento era un cliente de la empresa. Tres miembros del equipo viajamos a Tuxtla Gutiérrez para preparar lo necesario. Tuvimos mucho trabajo dado que tuvimos que dar de alta en el sistema prácticamente todo, usuarios, secciones del estadio, secciones a la venta, tipos de boletos, configuración de venta, etcétera. El personal del club nos proporcionó los lugares que se reservarían y los boletos que se emitirían como cortesías. El primer día de venta tuvimos infinidad de problemas empezando con la conectividad, después el rendimiento de la aplicación no era el deseado, además de que a los operadores no se les había dado la suficiente capacitación y había que asistirlos constantemente. Durante la noche de ese día hicimos modificaciones al sistema de venta para hacerlo más ligero y sencillo, el resultado fue el deseado.

El sistema se extendió a más puntos de venta hasta finalizar la participación del club en el torneo de futbol soccer. A pesar del éxito, el proyecto no se dio por terminado y se incluyeron nuevos requerimientos. Gracias a esto, mi jefe directo fue ascendido a director de área y yo a líder de proyecto, ya estaba familiarizado con el proyecto y el manejo del mismo, esto me dio una gran experiencia en la administración de recursos, el manejo de tiempos, que era algo que ya hacía pero no lo hacía al nivel del proyecto en general. A partir de este momento los cambios dejaron de implicar

largas desveladas y nos enfocamos en cambiar las hojas de estilos y en preparar presentaciones para potenciales compradores del servicio. El producto cambió de nombre a “Nexon” y también se implementó en los puntos de venta de las taquillas del nuevo estadio “Omnilife” del club Guadalajara.

Este proyecto fue la piedra angular de mi experiencia en el área de desarrollo de sistemas, cuando me integré al proyecto era prácticamente una hoja en blanco y el hecho de que el proyecto estuviera tan bien pensado en cuanto a metodologías, buenas prácticas, documentación y calidad hizo que cambiara mi visión particular de hacer las cosas. Aun cuando un proyecto sea pequeño, hay que pensarlo siempre como un componente y pensar en su posible expansión, siempre hacer algo de la manera más genérica y parametrizable posible.

3) FEDERACIÓN MEXICANA DE FUTBOL

A partir del 3 de noviembre de 2009 comencé a trabajar en la Federación Mexicana de Futbol, una empresa responsable de manejar y dirigir todo lo relativo a las ligas de futbol soccer del país. Los procesos de desarrollo de sus sistemas no estaban en un nivel alto de desarrollo, a pesar de eso, los sistemas con los cuales gestionaban al futbol mexicano tenían un nivel alto de madurez y estaban en constante cambio, habían iniciado como sistemas sencillos y habían sido actualizados sin llevar un control de cambios ni manejo de requerimientos para los mismos. La mayor parte del tiempo el trabajo consistía en dar soporte a una aplicación llamada Sistema Integral de Información Deportiva (SIID), éste se basaba en una serie de módulos web que podían ser consultados a través de una intranet, el propósito del sistema era manejar la información relativa a los jugadores de todas las ligas del futbol mexicano. El sistema había sido pensado para manejar la mayor cantidad de información del lado del cliente, evitando lo más posible los “postbacks”, además se pensó en hacer componente individuales, cada uno con su código HTML y JavaScript para llamarlos a través de inicializadores, sin embargo, algunos de los desarrolladores no comprendían el concepto de objeto, haciendo que sus componentes mantuvieran sus funciones de manera global y esto era la causa de la mayoría de los problemas. Si un desarrollador hacia un componente con una función de cierre “cierra_ventana()” y en su código mandaba llamar a otro componente con una función con el mismo nombre, lo que causaba era la sobreescritura de la primera, causando así un funcionamiento anormal.

Realicé un sistema para el manejo de controversias dentro de las áreas jurídicas del organismo. Una controversia se presenta cuando algún o algunos jugadores tienen problemas de índole legal con su equipo o un equipo con otro equipo por el contrato de un jugador, etcétera. A pesar de no manejar procedimientos específicos para el levantamiento de requerimientos, se especificó que el sistema debía tener como objetivo mantener un archivo digital de los documentos legales escaneados y asignarlos a un registro para darles seguimiento, así como incorporar un sistema de alertas y notificaciones para las partes involucradas y para la propia Federación con la finalidad de agilizar trámites legales. El sistema debía dividir las controversias en aquellas cuyos involucrados eran mexicanos y las que participaba al menos un participante extranjero ya fuera uno o varios jugadores o un club extranjero.

Debido a que la institución no contaba con los procesos para desarrollar software de calidad para tener un buen manejo de los requerimientos, no se aseguraba que el producto final fuera lo esperado por el cliente, entonces compensé esta situación acercándome al cliente y a los usuarios finales y

acorde con ellos implementar un desarrollo basado en el modelo en espiral de tal manera que a partir de la segunda iteración los cambios y requerimientos se solicitarían a través de un correo electrónico para de esta manera se generara una evidencia de las solicitudes, mientras que yo manejaría el desarrollo, versionamiento y liberación del sistema.
El desarrollo inició desde cero, pero debía ajustarse al modelo de sistemas que se manejaba en ese lugar, es decir utilizar componentes JavaScript para manejar toda la información del lado del cliente y evitar lo más posible los “postbacks” y no utilizar “code behind”.

Un postback es un HTTP POST, es decir, el contenido de la etiqueta HTML <form> es enviado de vuelta al servidor. POST es uno de los muchos métodos de petición, conocidos como “Request” soportados por el protocolo HTTP. El modelo de petición POST está diseñado para que un servidor web acepte datos encapsulados en el contenido de la misma petición.

Code-behind es un modelo de programación en el cual el diseño y el código ejecutable se encuentran en archivos diferentes, quedando el código en un archivo del mismo nombre que el del diseño pero con la extensión del lenguaje en que está escrito, por ejemplo utilizando el lenguaje c# los archivos serían nombrados como page1.aspx y page1.aspx.cs, esto permite que se puedan realizar modificaciones al diseño o a la programación de manera separada, haciendo menos probable una alteración no deseada en el diseño o la programación.

La elaboración del código HTML de la interfaz gráfica, el código de funcionalidad en JavaScript y las consultas SQL fueron completamente realizados en Notepad++, que ofrece herramientas para la codificación, el desarrollo tomó aproximadamente un mes entre codificación y pruebas. La parte de la base de datos estaba en manos de otra persona a quien se le debían pasar las especificaciones de tablas, campos y demás objetos que debían crearse o alterarse. Esta consistía en dos tablas, una con los campos necesarios para el registro y otra para el catálogo de tipos de controversias y éstas se unían al resto de tablas de la base de datos de la Federación que contenía información sobre los empleados, los jugadores y clubes para tener llaves foráneas de los registros de los involucrados. El resultado fue un sistema que cumplía con lo requerido. Había tres perfiles contemplados para el usuario final: administrador, gestor, jugador o representante, cada uno hacía funciones diferentes del sistema. El sistema fungía básicamente como un canal de información entre las partes y la FMF sobre el estado de sus controversias. El gestor registraba las controversias y solicitaba información y documentación a las partes a través del sistema, posteriormente las partes consultaban el estado de

la controversia y capturaba o cargaban la información que se les había requisitado, finalmente la Federación resolvía la controversia y notificaba a los involucrados.
A pesar de que se intentaron implementar buenas prácticas y procesos al proyecto, ante la constante petición de modificaciones y el tiempo invertido en su desarrollo, no fue posible elaborar una documentación detallada del mismo.

4) TELNORM

Abandoné la FMF el 12 de febrero de 2010 y el 15 de ese mismo mes ingresé formalmente a Telnorm, una empresa dedicada a la implementación de equipos IVR (Interactive Voice Response) y desarrollo de soluciones GPS (Geo Posicionamiento Satelital). Lo cierto es que esta empresa tampoco tenía procesos ni metodologías para el desarrollo de sus sistemas, sin embargo, no tenían constantemente "bomberazos", básicamente porque tenían pocos sistemas, la mayoría de sus proyectos consistían en integrar hardware o software y no tanto en desarrollar productos desde cero. Estaban por crear nuevos proyectos y se me pidió iniciar un proyecto para la administración y monitoreo de un Call Center, el cual se integraría con otro proceso ya en desarrollo para la comunicación VoIP (Voz sobre IP), la idea consistía en tener una base de datos de alta demanda donde se estuviera actualizando en tiempo real la información de los recursos del call center.
Al no haber procesos que aseguraran la calidad del producto a desarrollar, decidí implementar lo que había hecho en la Federación, sin embargo, aquí no existía un cliente como tal, era un proyecto que requería mucha más investigación.
Decidí usar como base un componente que realicé en mi tiempo libre utilizando el lenguaje C# para interactuar con bases de datos SQL y MySQL bajo el concepto de realizar un manejo estándar y automático de las consultas y sentencias que se pueden enviar a una base de datos desde una capa superior, la finalidad es reducir la codificación. Se encuentra publicado en el sitio web de sourceforge.net y que puede ser descargado de la dirección http://sourceforge.net/projects/dbframework.

El proyecto avanzó libremente con base en la información que tenía de otros compañeros que tenían a su cargo otra parte de la misma solución consistente en un software de telefonía IP. El proyecto se presentó a un posible cliente, el cual era el banco Compartamos pero ellos necesitaban una solución de escritorio, así que el proyecto cambió a ser una solución de escritorio, básicamente sin cambios y utilizando los mismos componentes, todo esto se realizó sin requerimientos formales ni metodología alguna, bajo el argumento de que sólo era una demostración. Cuando lo presentamos en una

demostración, el cliente no estuvo satisfecho pues el software no cubría los requerimientos que él había especificado al área de ventas y a la dirección del área de desarrollo, esto causó mi descontento pues a mí como líder de proyecto nunca me fueron entregados tales requerimientos, obviamente se pretendía desarrollar una solución genérica, pero es importante obtener y comprender las necesidades de los clientes para poder crear una solución, hacer de esta genérica depende de las buenas prácticas que se incorporen a lo largo del desarrollo del proyecto. A pesar del fracaso en la negociación el proyecto continuó, esta vez se incorporaron las características de los requerimientos de Compartamos Banco sobre todo en la parte de monitorización de los agentes. Esta vez redefinimos alcances con el área de ventas, para saber qué era lo que se pretendía vender, rediseñé la base de datos, se redefinieron las interfaces ahora la empresa podía ofrecer al mercado una solución de escritorio y una solución web. Aunque propiamente no definimos una metodología de trabajo decidí continuar implementando el modelo iterativo.

El producto final ofrecía las siguientes características:

- Administración de agentes. Este módulo permitía agregar, editar y eliminar la información de los agentes, suspenderlos, ver reportes de su desempeño, registros históricos de sus sesiones, llamadas atendidas y los resultados de cada una, así como estadísticas y posicionamiento.

- Administración de grupos. Los agentes debían por definición pertenecer al menos a un grupo, el concepto de grupo estaba diseño para una fácil administración de los agentes, así como para facilitar la tarea de asignarlos a campañas y horarios ya definidos.

- Administración de campañas. Una campaña se define como el periodo de tiempo en el que se realizan llamadas con un fin específico, por ejemplo cobranza o promociones. Se permitía establecer las fechas de inicio y fin de la campaña, así como los días y horarios en los que estaría operando dentro del periodo. Se permitía asignar agentes individualmente pero lo más recomendable era asignar grupos.

- Reportes. El registro de una llamada estaba ligado a una campaña y a un agente y/o un grupo, guardaba un estado de finalización, el tiempo en llamada y el tiempo que hubo en cada estado

del ciclo de la misma. Con toda esta información podían generarse reportes tanto específicos como generales yendo de lo particular de una llamada o un agente al comportamiento de la campaña a lo largo del tiempo.

- Monitoreo en tiempo real. De manera gráfica se observaba a los agentes atender a las llamadas, esto permitía al administrador supervisar el comportamiento de los agentes, grupos y campañas para hacer cambios para compensar o balancear el tráfico de las llamadas en caso de ser necesario.

- Parametrización. Las opciones de parametrización permitían asignar las cadenas de conexión necesarias para cada usuario final, así como las principales características de la presentación.

- Orígenes de datos. El sistema soportaba únicamente bases de datos SQL, MySQL y archivos xls y xlsx.

Posteriormente lideré un proyecto consistente en un portal web que organizaba conferencias telefónicas y permitía reproducir el audio de las mismas a los usuarios participantes, con ayuda de una API (Interfaz de Programación de Aplicaciones) de un producto llamado Cybertech que provee las funcionalidades para administrar los audios de sistemas IVR. El desarrollo era considerablemente sencillo así que traté de aprovechar el proyecto para concientizar acerca de las ventajas de utilizar buenas prácticas de desarrollo, comenzando por el uso de nomenclaturas y documentación, sin embargo no fue sencillo, al no haber un compromiso o una parte que auditara la calidad. Después de ver que los otros desarrolladores y líderes de proyecto no estaban motivados a aplicar metodologías para mejorar el desarrollo y que la dirección del área estaba conforme con el funcionamiento del área a pesar de que posteriormente tuvieran que resolver problemas por falta de planeación decidí buscar otra manera de trabajar.

Durante el tiempo que laboré en esta empresa, me mantuve difundiendo las ventajas de trabajar de manera organizada, utilizando una metodología, así como buenas prácticas en cada proceso. Logré que la empresa centralizará sus proyectos en un repositorio y comenzara a utilizar un versionador, el cual fue Tortoise, una herramienta de software libre. Anteriormente cada desarrollador tenía una copia de cada proyecto en su equipo de cómputo, sin llevar un control de versiones entre cada copia.

Establecí con el área las nomenclaturas que debían utilizarse durante la codificación y un estándar para el desarrollo de componentes y aplicaciones modulares.

5) FACILEASING

Busqué empleo dentro de una consultoría, el trabajar para una consultoría tiene ventajas y desventajas respecto a ser empleado de una empresa, entre sus principales desventajas podemos encontrar que usualmente pagan por honorarios, lo cual para algunos no es desventaja, y el trabajo puede no ser constante. Sin embargo las ventajas incluyen además de mejores sueldos, una mejor administración de proyectos puesto que usualmente las consultorías tienen certificaciones de calidad aunque depende de la misma consultoría. Me contraté en Sinersys Technologies, una consultoría de TI con certificación de modelo de procesos CMMI nivel 3, donde después de evaluar mis conocimientos y mis prácticas me enviaron a trabajar a una empresa que se llama Facileasing, la cual se dedica al arrendamiento de flotas vehiculares, trabajar aquí fue una muy buena experiencia, todo estaba muy bien documentado y cada requerimiento se especificaba muy bien. Estuve a cargo de la modificación de dos sistemas que ya existían y la creación de un tercero. El primero era la modificación de un sistema llamado ASV (Administración de Servicios Vehiculares) para incorporar las recomendaciones de proveedores. Cuando un vehículo presentaba algún desperfecto o por programación requería servicio, el cliente llamaba al call center de Facileasing donde solicitaba una “Recomendación de Proveedor”. En el sistema ASV el agente del call center buscaba un proveedor cercano que se ajustara a las necesidades del cliente, en el sistema este movimiento se guardaba como una “Recomendación de Proveedor”. Si el proveedor llevaba la unidad vehicular con el proveedor de servicios que le había sido recomendado por Facileasing, el proveedor se comunicaba ahora al call center para notificar de una Orden de Servicio.

Los requerimientos especificaban muy bien todos los campos necesarios para capturar la información, así como la funcionalidad. Elaboré los diagramas de flujo y de secuencia para la documentación del proceso y la modificación y continué con la planeación y el desarrollo.

Se organizó la información de las asesorías en la base de datos, se cambió el almacenamiento de las asesorías a una sola tabla donde se organizó toda la información de las mismas (valet, recomendar proveedor, asistencia vial, mecánica, aseguradora y de procedimientos) se unificó en una sola tabla distinguiendo por el tipo de asesoría. Se migraron los datos de las asesorías almacenadas hasta el momento utilizando Transact-SQL para adecuarlas al formato de los nuevos registros. Las asesorías se vincularon con los servicios de mantenimiento a través de llaves foráneas a un catálogo de tipo de asesorías previamente creado.

La pantalla de captura de asesorías se modificó para permitir la selección del tipo de asesoría, con esto se harían visibles dinámicamente los campos relacionados con la asesoría seleccionada. Se modificaron los reportes, páginas, componentes web y de base de datos relacionados para cumplir con la nueva estructura.

Posteriormente se me asignó un proyecto nuevo para integrarlo a un sistema ya existente denominado PPF (Portal de Proveedores Facileasing), el cual consistía en desarrollar un componente que permitiera a los proveedores de Facileasing cargar sus facturas electrónicas a través del portal, este componente unificaría de manera lógica al sistema ASV con el PPF, haciendo de esta manera que todo el proceso quedara centralizado, documentado y evidenciado de manera digital, ya que previamente las facturas se recibían vía mensajería o si eran documentos digitales mediante correos electrónicos, sin tener un control preciso de correspondencia con respecto a los registros guardados por el sistema ASV.

Durante la fase de planeación del proyecto, se contempló utilizar un módulo provisto por una empresa externa y que además ya se utilizaba para el resguardo de archivos, este producto era eFactura y brindaba la opción de validar a través de un web service si la factura digital era correcta para propósitos fiscales, es decir, cumplir con la estructura de CFD o CDFI emitidas por el SAT. Sin embargo, la planeación del proyecto debió extenderse debido a que la empresa fue absorbida por el grupo financiero Bancomer, quienes impusieron sus procesos para la validación de CFD's y CFDI's, el cual consistía en recibir los comprobantes fiscales digitales en un archivo comprimido formato Zip a través de un correo electrónico, pasadas hasta 48 horas el sistema regresaba a vuelta de correo un archivo txt con las cadenas de los Comprobantes Fiscales Digitales válidos y otro con los inválidos. Definitivamente este cambio a la planeación original reducía de manera dramática la usabilidad y la eficiencia del proceso. Entonces se planeó crear dos procedimientos dentro de un mismo sistema. El primero recopilaría la información a determinada hora, obtendría todos los documentos y los comprimiría en un archivo formato Zip y finalmente enviaría el correo al destinatario de Bancomer. El segundo proceso se mantendría monitoreando la llegada del correo respuesta de Bancomer y extraería los archivos, los cuales serían distinguidos por la nomenclatura de sus nombres, se leería el contenido de los archivos y se ejecutaría un proceso para marcar como válidas los comprobantes fiscales de tal manera que se pudiera continuar con el proceso. Posteriormente se modificó el alcance del proyecto especificando que el sistema ejecutaría el proceso utilizando las herramientas provistas

por Bancomer para todas las áreas de la empresa. Con un alcance más definido y una vez ya establecidos los protocolos de Bancomer para la interacción con sus herramientas, la fase de análisis continuó con la elaboración de los requerimientos funcionales y no funcionales, diagramas UML de secuencia y de clases, dando origen a un proyecto nuevo denominado GAF (Gestor de Archivos Fiscales). El objetivo era integrar a todas las áreas de Facileasing al proceso de validación de facturas digitales establecido por Bancomer y al mismo tiempo crear un componente que permitiera unificar los sistemas ASV y PPF para dar consistencia a la información manejada por ambos sistemas. Bancomer tenía como propósito hacer eficiente y redituable a su reciente adquisición, en lo personal considero que sus sistemas de validación resultaban ineficientes en comparación con lo que se había estimado originalmente, además el tiempo que se empleó en volver a planificar el proyecto hizo que el desarrollo del proyecto como su implementación no resultarán tan benéficos en relación al costo de desarrollo y lo que sería después el mantenimiento.

Se elaboró el plan de trabajo y se comenzó el desarrollo, en el núcleo de la aplicación se programó en el archivo global.asax, encargado de ejecutar métodos a nivel aplicación, la rutina para que a determinada hora, establecida desde el archivo web.config, se obtuvieran todos los archivos de una determinada carpeta, que sería el repositorio de los CFD y CFDI's, los comprimiera en un archivo extensión Zip y enviara un correo. Esta rutina se ejecutaría diariamente, excepto sábados y domingos y cuando no se encontrase ningún archivo a validar. En el siguiente paso del desarrollo se creó una interfaz web que seguía el siguiente proceso:

- El empleado se autentifica en el sistema
- El sistema identifica a qué área de Facileasing pertenece el empleado
- El empleado selecciona los archivos CFD y CFDI's para su validación
- El sistema extrae de los archivos el monto total y el nombre de quien expide, para crear un registro en base de datos y asociarlo al empleado y a la fecha para su posterior identificación
- El sistema carga los archivos a una carpeta en el servidor y muestra un mensaje para confirmar la operación

Mientras tanto, de manera paralela, se modificó el PPF para que el proveedor capturara sus facturas digitales, con las siguientes características:

- El proveedor se autentifica en el sistema
- El proveedor captura la información de montos de sus facturas
- El sistema valida el RFC y el monto de los archivos cargados por el proveedor
- El sistema ubica los archivos a una carpeta determinada en el sistema para permitir que el sistema GAF los procese para su validación
- El proveedor puede dar seguimiento a la validación de sus facturas mediante una pantalla, donde los registros de las facturas muestran sus estados.
    - Listo para facturar
    - En proceso de validación
    - Válido
    - No Válido, refacturar.
- El sistema analiza de manera independiente la respuesta de los servicios de Bancomer y actualiza la base de datos y los estados de los registros relativos a documentos CFD y CFDI's.
- Finalmente el sistema PPF reubica los archivos para su resguardo y posterior consulta.

Ambos sistemas se implementaron de manera preliminar con vistas a una posible actualización por parte de los servicios provistos por Bancomer. En lo particular me parece poco práctico que un sistema tarde hasta dos días en devolver una respuesta a un proceso que podría ejecutarse en tiempo real, pero en ocasiones hay que apegarse a lo que el cliente solicita, que es algo con lo que en esta área laboral se suele encontrar, muchas veces el cliente no sabe bien lo que quiere o lo quiere de cierta manera que no es eficiente. En este sentido no importa qué tan bien desarrollemos una metodología de desarrollo, de calidad y apliquemos buenas prácticas porque al final tendremos un producto muy bien hecho pero que no cumple con las expectativas del cliente. Es importante que en la fase de planeación se cuente con la presencia del encargado del proceso que se va a automatizar y de un desarrollador que tenga la experiencia para determinar si se pueden o no se pueden hacer las cosas y cómo hacerlas, es común escuchar comentarios con tono "humorístico" en el ambiente de las tecnologías de la información donde se hace alusión a que con la finalidad de vender una solución o cerrar un contrato se dice que sí a todo sin antes validar tiempos, riesgos y viabilidad, lamentablemente tiene su parte cierta, y se aprecia cuando al elaborar un plan de trabajo y contemplar el tiempo de los recursos, estos empiezan a tener sobrecargas de trabajo.

6) INSTITUTO MEXICANO DEL SEGURO SOCIAL

Posteriormente me integré al INAP (Instituto Nacional de Administración Pública) en su modalidad de consultoría para trabajar en el proyecto SIPSI (Sistema Integral de Prestaciones Sociales). Este proyecto lleva el control de todos los cursos que imparte el IMSS a toda la población del país.

Previamente la administración y el control de los servicios que proporciona Prestaciones Sociales del Instituto se hacía mediante la operación, funcionamiento y trabajo coordinado de dos sistemas computacionales, el Sistema de Información de Prestaciones Sociales (SIPS) y el Sistema de Información de Prestaciones Sociales Institucionales (SIPSI), mediante los cuales se realizaba la programación de actividades, el proceso de inscripción, la generación de credenciales, el cobro y control de cuotas, el seguimiento mensual y la entrega de información a otras entidades financieras y de logística.

Anteriormente el proceso de los servicios proporcionados por Prestaciones Sociales a través del sistema SIPS era un sistema desarrollado en el lenguaje Cobol, tecnología con más de veinte años en el mercado. Por otro lado el SIPSI era un sistema que utilizaba tecnologías más actuales, pero que combinaba más de una para diferentes procesos, para la actualización de versiones nuevas y/o cualquier cambio por mínimo que fuera de la aplicación, también implicaba el uso de la fuerza laboral a nivel nacional de las Coordinaciones Delegacionales de Informática (CDI's) para llevarlo a cabo en todos y cada uno de los equipos de las Delegaciones y Unidades Operativas de Prestaciones Sociales. Con la tecnología web, por otro lado, ha sido posible reducir en mucho el esfuerzo necesario para realizar cualquier cambio en el sistema de manera centralizada, permitiendo mantener la operación en tiempo y forma del sistema. Además, se contaban con bases de datos locales en las Delegaciones y en las Unidades Operativas, las cuales debían sincronizarse para replicar su información y actualizar una base de datos central.

El objetivo principal del proyecto de Rediseño del Sistema de Información de Prestaciones Sociales Institucionales era contar con un sistema capaz de cubrir en su totalidad el proceso general de los servicios que brinda el Instituto respecto a las Prestaciones Sociales. En cuyo esquema se contempló el uso de la plataforma web y una Base de Datos Centralizada, buscando agilizar y optimizar dicho proceso.

El Instituto Mexicano del Seguro Social a través de la Coordinación de Prestaciones Sociales brinda diferentes servicios como son: cursos, actividades complementarias, tales como campañas, talleres, pláticas, ligas deportivas, eventos, servicios y actividades teatrales que benefician a toda la población. El objetivo de estos servicios es promover el estilo de vida saludable, prevención y detección de enfermedades, accidentes y adicciones, educación para una vida mejor, fortaleciendo los valores y comportamientos que propicien una socialización positiva de los individuos, impulsando las potencialidades individuales y el desarrollo de habilidades productivas y recreativas.
Se desarrolló un nuevo sistema web que abarcaban los procesos de:

- Planeación y programación de servicios.
- Inscripción y cobro de cuotas de recuperación por servicios.
- Registro de asistencias a cursos regulares, actividades complementarias y aforo en uso de instalaciones.
- Promoción y difusión de los servicios.
- Informes de resultados.

Planeación y programación de servicios:
Este proceso es el encargado de la elaboración y autorización de los programas anuales de actividades. En nivel central, basados en el análisis de resultados, se elaboran y autorizan los Criterios Técnicos para la Programación Anual de Actividades, los cuales son enviados a las Delegaciones para su análisis, captura, integración y aprobación del Programa Anual Delegacional, y finalmente enviado a normativo central para su revisión y autorización del Programa Anual por Delegación. Considerando que dentro de los Criterios de Programación se encuentra la posibilidad de que las UOPSI's programen cursos locales (Opción Local) es necesario incorporar el alta de éstos cursos durante el proceso y de acuerdo a la demanda que exista en la Unidad.

Inscripción y cobro de servicios:
Este proceso se lleva a cabo sólo a nivel de las Unidad Operativas, y es el proceso por el cual se recibe y atiende los requisitos de inscripción y cobro, tanto para cursos, como ligas deportivas en el caso de actividades complementarias y uso de instalaciones. La inscripción y cobro de servicios, genera una ficha de pago con la cual el socio-alumno podrá cubrir la cuota de recuperación abonando en el banco el monto detallado en dicha ficha. En el caso de cursos, después de aceptar la solicitud se emite una credencial que identifique al socio-alumno.

Para la inscripción a las actividades complementarias el sistema debía tener la opción de registrar nominalmente a los participantes por medio del llenado de un formato en línea, específico para cada servicio, o bien, directamente en el lugar del evento.

Registro de asistencias y aforos:
Por medio de este proceso se recaba la información necesaria para el control de asistencias a cursos, actividades complementarias así como el registro de aforos en teatros.
Como requerimiento de este proceso se solicitó que la captura en la Delegación de las listas de asistencia se automatice a través de medios digitales, optimizando el tiempo de análisis y disminución de inconsistencias en los listados.

Promoción y difusión de los servicios:
Esta es una nueva funcionalidad que se presenta en el sistema, con la cual se pretende promocionar las actividades y eventos organizados por Prestaciones Sociales, así como también, se pretende que el sistema detecte las preferencias de los alumnos inscritos en base cursos a los que se han tomado en Prestaciones Sociales del Instituto.

Informes de resultados:
Con dicho proceso se da soporte a las necesidades de información como son los resultados de la operación en comparación con la estructura programática e información presupuestal, los reportes de metas que solicita PREI de forma mensual y en general de aquella información que da cuenta de la gestión de los procesos de Prestaciones Sociales, entre ellos se encuentran el Informe Anual a la Asamblea General, Cuenta de la Hacienda Pública Federal, Informe de Gobierno, Reporte de Gestión, y otros.

El rediseño del Sistema de Prestaciones Sociales está basado en una arquitectura web, cuyos datos se encuentren centralizados y sean actualizados en línea en tiempo real, cubriendo como mínimo las necesidades actuales de los procesos de negocio que realiza Prestaciones Sociales.

Características del Sistema
Programa Anual
Criterios para la Programación Anual de Actividades:

Este módulo permite dar de alta, baja y modificar los criterios bases para la generación y validación de los programas que capturarán las UOPSI´s. Estos criterios son integrados a nivel central por la División Apoyo Técnico y Mejora de Procesos. La estructura programática podrá ser cambiada para cada ejercicio fiscal.

Inscripción

Registro de usuarios:

Esta funcionalidad está disponible solamente para los perfiles de Unidad Operativa.

Se registran los datos para la preinscripción según un formato establecido tanto para derechohabientes, no derechohabientes y personas derivadas del Área Médica.

Para Actividades Complementarias, se registran los datos para las siguientes actividades:

- Pláticas
- Campañas
- Talleres
- Ligas deportivas
- Eventos
- Servicios

Para el caso de cursos: Se emite la credencial que identifique al socio-alumno y su reposición en caso de extravío, así como un reporte de participantes (Lista de asistencias).

Para uso de instalaciones: Se confirma la reservación al nombre correspondiente.

Para ligas deportivas: Se confirma la inscripción del equipo y de sus integrantes.

Validar vigencia:

Valida la certificación de vigencia de derechos, de acuerdo a la calidad jurídica del solicitante para la inscripción a cursos a fin de hacer válidos los descuentos y beneficios que apliquen.

Enlace a banco:

Se recibe la información del banco de manera diaria, de los pagos que se realicen los socio-alumnos por cualquiera de los servicios que a continuación se enlistan:

- Inscripción y credencial.
- Costo del curso (parcial o total).
- Reposición de credencial.
- Uso de instalaciones.
- Inscripción a ligas deportivas.

Seguimiento mensual

Registro de asistencia a cursos:

El instructor registra la asistencia de los alumnos a los cursos (por grupo) desde el inicio del curso hasta el fin. Se emite un reporte mensual de asistencias a cursos.

La información que genere este proceso es la fuente que alimenta el Reporte de Seguimiento Mensual de Metas PREI.

Registro de asistencia a actividades complementarias:

Se realiza el alta de asistencia de los socios-alumnos a las actividades complementarias, de igual forma se registra la información por uso de teatros. El registro se realiza de forma nominal a través de un formulario en línea y/o en el lugar del evento. Se emite un reporte mensual de asistencias.

La información que genera este proceso es la fuente que alimenta el Reporte de Seguimiento Mensual de Metas PREI.

Registro de reservaciones para uso de Instalaciones:

Se registran los datos para la solicitud de uso de instalaciones, la cual es la reservación de dicha instalación, que será usada en la fecha, hora y para la actividad señalada en la solicitud.

Se emite un reporte mensual de ocupación de instalaciones.

Módulo de reservaciones de instalaciones:

Se incluye un módulo de reservaciones para el seguimiento y control sobre el uso de instalaciones. Este módulo cubre el proceso desde la reservación inicial, pago y utilización del servicio.

Las reservaciones son mantenidas activas sin pago, hasta 72 horas desde su ingreso, luego son canceladas, teniendo la opción de modificar desde un perfil administrativo este período de acuerdo con las necesidades de prestaciones sociales.

El pago se realiza a través de la emisión de una ficha de depósito y por medio de la interfaz dentro del sistema donde a través de la referencia numérica generada por la institución bancaria se coteja el pago de la reservación.

Cobro de cuotas:

Se realizan las cuentas que están previamente definidas y relacionadas a un curso o convenio. El usuario al momento de preinscribirse recibe una referencia generada a través de algoritmos proporcionados por la institución bancaria y monto a depositar para quedar inscrito.

Se registra el cobro por los siguientes conceptos:

- Inscripción y credencial
- Reposición de credencial
- Cursos
- Ligas deportivas
- Uso de instalaciones
- Teatros

Las cuotas para los conceptos mencionados pueden ser:

- Cuota única
- Mensuales
- Periodos variables

Conciliaciones Bancarias:

Recepción de Archivo:

Se recibe información de la institución bancaria diariamente, de los pagos que realicen los socios-alumnos por cualquiera de los servicios para cada convenio establecido (bienestar social, FIDEIMSS y FIDTEATROS). El detalle de información del depósito a cada convenio, está compuesto por:

Tipo de transacción (inscripción, mensualidad, reposición de credencial, etcétera.)

- Guía CIE
- Referencia (línea de captura)
- Datos de depósito

- Número de convenio
- Nombre del depositante
- Fecha de depósito
- Monto

Se emite el reporte diario de inconsistencias en depósitos bancarios por convenio, por delegación y por unidad operativa.

Información presupuestal:
Avance Financiero de Presupuesto de Ingresos
Reporte mensual de ingresos

- por convenio
- por delegación
- por unidad operativa
- por servicio (cursos, ligas o uso de instalaciones).

Dicho reporte se genera con la información proporcionada por el banco.

Enlace con sistemas internos
El sistema establece un enlace con los diferentes sistemas internos con los que cuenta el Instituto:

- SINDO
- PREI
- ECE

Para esto se tendrá una interfaz con cada sistema:
Con SINDO para la validación de vigencia de derechos de acuerdo a la calidad jurídica del solicitante para la inscripción a cursos, actividades complementarias y uso de instalaciones, a fin de hacer válidos los descuentos y beneficios que apliquen.
Con PREI para la integración del Programa Anual de Metas y del Informe Mensual de Avance de Metas.
Por último con ECE para certificar usuarios referidos del área médica que implica la realización del registro y seguimiento del usuario a una actividad en alguna UOPSI.

Promoción de actividades:

- Administración de correo electrónico, (creación, eliminación, actualización, consulta, reporte).

- Envió automatizado de correo (promoción de actividades).
- Administración de mensaje de texto, (creación, eliminación, actualización, consulta, reporte).
- Envío automatizado de mensaje de texto.
- Administración de preferencias de usuario, (creación, eliminación, actualización, consulta, reporte).
- Búsqueda de usuario por preferencia.
- Identificación automática de preferencias del usuario.
- Envío automático de actividades a realizar según los resultados obtenidos en la identificación automática de preferencias de usuario.
- Búsqueda de preferencia por usuarios.
- Administración de información de actividades (creación, eliminación, actualización, consulta, reporte).
- Administración de usuario a foro (creación, eliminación, actualización, consulta, reporte).
- Administración de estadísticas (consulta, reporte, creación, eliminación, actualización, consulta, reporte).

Requerimientos de documentación

Se entregaron los siguientes entregables por fase (fases del proceso de desarrollo - RUP):

- Fase de Inicio
    - Plan de proyecto
    - Lista de casos de uso
    - Plan de gestión de riesgos
- Fase de elaboración
    - Plan de gestión de riesgos actualizado.
    - Plan de proyecto actualizado.
    - Documento de especificación de requerimientos
    - Arquitectura de la solución (documento de arquitectura).
    - Documentación de análisis (modelo de análisis)
    - Documentación de diseño (modelo de diseño).
    - Plan de pruebas.
    - Matriz de trazabilidad

- Fase de construcción
    - Plan de proyecto actualizado.
    - Plan de gestión de riesgos actualizado.
    - Plan de pruebas de aceptación.
    - Código fuente probado (se ha completado el plan de pruebas de sistema e integración).
    - Manual de instalación de la aplicación.
    - Matriz de Trazabilidad actualizada.

- Fase de transición
    - Sistema testeado por el usuario (las pruebas de aceptación del usuario se completaron)
    - Aplicación entregada.
    - Manual de instalación.
    - Manual de usuario.

El Instituto establece como norma para desarrollo de sistemas un documento en la cual se detallan las etapas, disciplinas, actividades y roles a seguir basado en el Proceso Unificado de Desarrollo de Software (RUP).

La construcción, es la fase que sigue a la elaboración del sistema y tiene como objetivo la implantación de los requerimientos (casos de uso) y el diseño de alto nivel (arquitectura).
En esta fase participan desarrolladores de sistemas .NET, encargados de la codificación de los componentes que son parte del sistema.

El proceso de construcción es el seguimiento de las actividades que a continuación se detallan:

- Definición de casos de uso y requerimientos
    - Casos de uso
    - Especificaciones de requerimientos funcionales y no funcionales
- Diseño de alto nivel
    - Herramientas de proyecto
    - Diseño de base de datos

    - Políticas
    - Arquitectura .NET
    - Selección de componentes reutilizables
- Ambiente de desarrollo
    - Repositorio CVS
    - Infraestructura de servidor web
    - Infraestructura de base de datos
    - Instalación de Frameworks
    - Herramientas IDE
    - Ambiente de pruebas unitarias e integrales
- Diseño detallado
    - Diagramas UML
    - Modelado UML
    - Aplicar patrones de diseño
- Codificación
    - Interacción CVS
    - Uso de IDE para aterrizar diseño
    - Plasmar pruebas unitarias
- Pruebas integrales
    - Integrar pruebas unitarias
    - Realizar pruebas integrales y de regresión
- Control de versiones
    - Administrar versiones
    - Incorporar cambios
    - Mantener liberaciones
- Liberar versiones
    - Empaquetar
    - Scripts de instalación o actualización
    - Crear rama de versiones
    - Pruebas funcionales
- Mantenimiento y correcciones

- Incorporar nuevos requerimientos

Se elaboró un diseño detallado cuyo propósito es generar los espacios de nombres y clases .NET necesarios para cubrir cada requerimiento.

Para el diseño detallado, se necesita cumplir lo siguiente:

- Utilización del lenguaje unificado de modelado (UML). Se deben de realizar los siguientes diagramas UML:
  - Diagramas de paquetes. Contiene el diagrama de todos los espacios de nombres del sistema, indicando las dependencias entre los mismos y los espacios de nombres de terceros que se están reutilizando.
  - Diagramas de clases. Contiene por cada espacio de nombres, las clases derivadas del sistema y debe contener la siguiente información:
    - Símbolo de clases
    - Propiedades privadas, protegidas y públicas
    - Constantes
    - Métodos privados, protegidos y públicos
    - Relaciones de asociación, composición, herencia e implantación.
    - Si la clase implanta una interface, indicarlo con el símbolo UML correspondiente.
  - Diagramas de secuencia. Se deben utilizar para expresar el flujo de mensajes entre los objetos del sistema. Debe usarse como un mecanismo de comunicación para agilizar el entendimiento de los algoritmos implantados por los métodos de una clase. No se debe utilizar los diagramas para explicar a detalle los pasos del sistema; dado que complicaría la lectura del diagrama. Se debe favorecer la lectura de código fuente para el entendimiento del flujo y solo usar el diagrama como apoyo de la descripción de los pasos más significativos.
  - Diagramas de actividad. En el caso de que un flujo sea conveniente modelarlo como una serie de actividades (diagrama de flujo) se deberá usar a este diagrama.
  - Diagramas de estado. Cuando el sistema involucre métodos que por el flujo de mensajes cambian el estado de un objeto, utilizar a este diagrama para ilustrar los estados posibles y sus transiciones.

La herramienta estándar de modelado del IMSS para sistemas .NET es Racional Rose XDE. Todo sistema deberá ser diseñado por medio de esta herramienta.
Tomando como base los diagramas de clases y por medio de la herramienta de ingeniería directa se generará al esqueleto de código .NET; el cual sirve como entrada a la disciplina de codificación. Cuando en la codificación se requiera cambiar la estructura de las clases o sus relaciones; se debe aplicar técnicas de ingeniería inversa para asegurar la consistencia del modelo con el código.

Los patrones de diseño elaborados por Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides se tienen que tomar en cuenta al realizar el diseño detallado del sistema y así asegurar que cumpla con criterios de modularidad, flexibilidad y parametrización.
No es conveniente que se traten de aplicar todos los patrones de diseño para todo el sistema. La aplicación de los patrones de diseño son el resultado de que los arquitectos y diseñadores se enfrentan con problemas relativos a:

- Creación de objetos.
- Estructura de objetos.
- Comportamiento de objetos.

No existe una norma bien definida de cuántos, cuándo, dónde y cómo aplicar dichos patrones, para ello se debe basar en el criterio al diseñar el sistema. Cuando se detecta la necesidad de utilizar un patrón de diseño se debe utilizar la herramienta de modelado de Rational para que genere el esqueleto de clases y los diseñadores se encargarán de especificar las clases correspondientes al sistema.

Los patrones de diseño .NET elaborados por Microsoft y los referidos por la norma del Instituto, se tienen que tomar en cuenta al realizar el diseño detallado del sistema y así asegurar que cumpla con criterios de escalabilidad, desempeño, parametrización y modularidad.
No es conveniente aplicar todos los patrones de diseño para la totalidad del sistema. La aplicación de los patrones de diseño son el resultado de que los arquitectos y diseñadores se enfrentan con problemas relativos a:

- Presentación web.
- Servicios.
- Distribución.
- Diseño de sistemas distribuidos.

Como en el punto anterior, no hay normas que indiquen cuántos, cuándo, dónde y cómo aplicar estos patrones; el criterio al diseñar vuelve a presentarse como el indicador de lo necesario y la justificación de su uso debe ser documentada.

La codificación en los sistemas .NET se debe realizar con C# u otro lenguaje soportado por la plataforma .NET. Los estándares de codificación son detallados en la norma del Instituto.
La herramienta de codificación para los sistemas .NET con la que se desarrolla en el Instituto es Visual Studio .NET 2010; con ésta se centraliza la automatización las tareas de compilación, empaquetamiento y distribución de componentes. Dicha herramienta, que es un Ambiente Integrado de Desarrollo (IDE), asegura que todo código o herramienta de construcción utilizada sea independiente del IDE utilizado.

Las pruebas unitarias son el punto de control para validar que el código construido cumple con lo especificado en el diseño detallado del sistema. Además sirven como ejemplos de uso de las clases probadas y para validar que el código se puede integrar al repositorio de control de versiones.
Estas pruebas deben demostrar que el código se ejecuta correctamente y que devuelve los resultados correctos.
Al construir pruebas unitarias se deben considerar los siguientes lineamientos

- Construir pruebas para cualquier código que pueda generar error.
- Construir pruebas para cualquier código que genere error.
- Cualquier código nuevo es “erróneo” hasta que tenga pruebas que demuestren lo contrario.
- Ejecutar pruebas unitarias de manera local en cada compilación.
- Ejecutar todas las pruebas unitarias antes de depositar el código en el sistema de control de versiones.
- No desarrollar pruebas unitarias para los métodos que únicamente sirven como acceso de atributos de clases.
- No desarrollar pruebas unitarias para métodos heredados de clases padre y que no se redefinieron en la implementación que se está probando.
- Se deben utilizar objetos falsos (mock objects) para construir pruebas en las que participen clases que todavía no se construyen o elementos que no se encuentran bajo nuestro control (clases de bibliotecas estándar, conectividad, servicios)

- Construir pruebas que se encarguen de comprobar:

- Que los resultados son los correctos.
- Que los límites son correctos.
- Que las relaciones inversas devuelven resultados correctos.
- Que las referencias cruzadas devuelven resultados correctos.
- Que se manejan adecuadamente las condiciones de error.
- Que el desempeño cae dentro de los límites establecidos.

Las pruebas unitarias que se construyen correctamente tienen las siguientes características

- Se pueden ejecutar sin intervención humana.
- Ejercitan al detalle los métodos de una clase.
- Se pueden repetir y devuelven el mismo resultado consistentemente.
- No tienen dependencias entre sí.
- Las pruebas unitarias se encargan de validar que los límites (boundaries) de las pruebas se encuentren:
- En conformidad a los formatos especificados para cada prueba.
- En el orden o secuencia especificados.
- Dentro de los valores mínimos y máximos razonables para la prueba.
- En la cantidad adecuada para la prueba.
- En la secuencia ordenada de tiempos.

Las pruebas unitarias se realizan con NUnit (http://www.nunit.org/), DotNetMockObjects (http://sourceforge.net/projects/dotnetmock) y se integran al ambiente de desarrollo con TestDriven.NET (http://www.testdriven.net/).
Adicionalmente, todos los ensamblados se deben verificar con la herramienta FxCop (http://www.gotdotnet.com/team/fxcop/) para comprobar el cumplimiento de los lineamientos de diseño, desempeño, seguridad, convención de nombres y localización.

Las pruebas integrales tienen como objetivo la verificación de que todos los componentes al ser ensamblados, preservan su funcionalidad.
Un conjunto de componente es codificado por un desarrollador. Dicho desarrollador debe entregar el código fuente acompañado de las pruebas unitarias.

Al construir el sistema, cada clase usa a otras clases y por cada clase, se genera una prueba unitaria, que a su vez debe invocar a las pruebas unitarias de las clases usadas.
La filosofía de este tipo de pruebas toma como base la estructura de dependencias existente entre las clases en todo el sistema; de manera que se cubre un barrido de integridad partiendo desde los componentes más bajos del sistema y se va escalando en la jerarquía de los componentes de manera que conforme avanza el proceso se va garantizando la integridad que guardan entre si los componentes y al finalizar el proceso se obtiene como resultado la certeza de la integridad del sistema.
Las pruebas de integración deben ser ejecutadas con regularidad y son el punto de control para verificar los avances de codificación del equipo de desarrollo. Debe usarse como métrica para identificar la tasa de defectos al momento de integración y así establecer los controles requeridos para mejorar la calidad en diseño y código.

Para automatizar las actividades de compilación, ejecución de pruebas unitarias, empaquetamiento, manejo de control de versiones o mantenimiento del ambiente de desarrollo; es posible usar la herramienta NAnt (disponible en http://nant.sourceforge.net)

El control de versiones del código fuente se debe hacer con la herramienta CVS (disponible en https://www.cvshome.org). Las operaciones a realizar con dicha herramienta son:

- Alta de nuevo código fuente
- Extracción de código fuente para mantenimiento del mismo
- Incorporación de cambios de código
- Extracción de código fuente para compilación y soporte a pruebas integrales
- Soporte a varias ramas de código fuente, para soportar versiones de producción, pruebas y desarrollo.

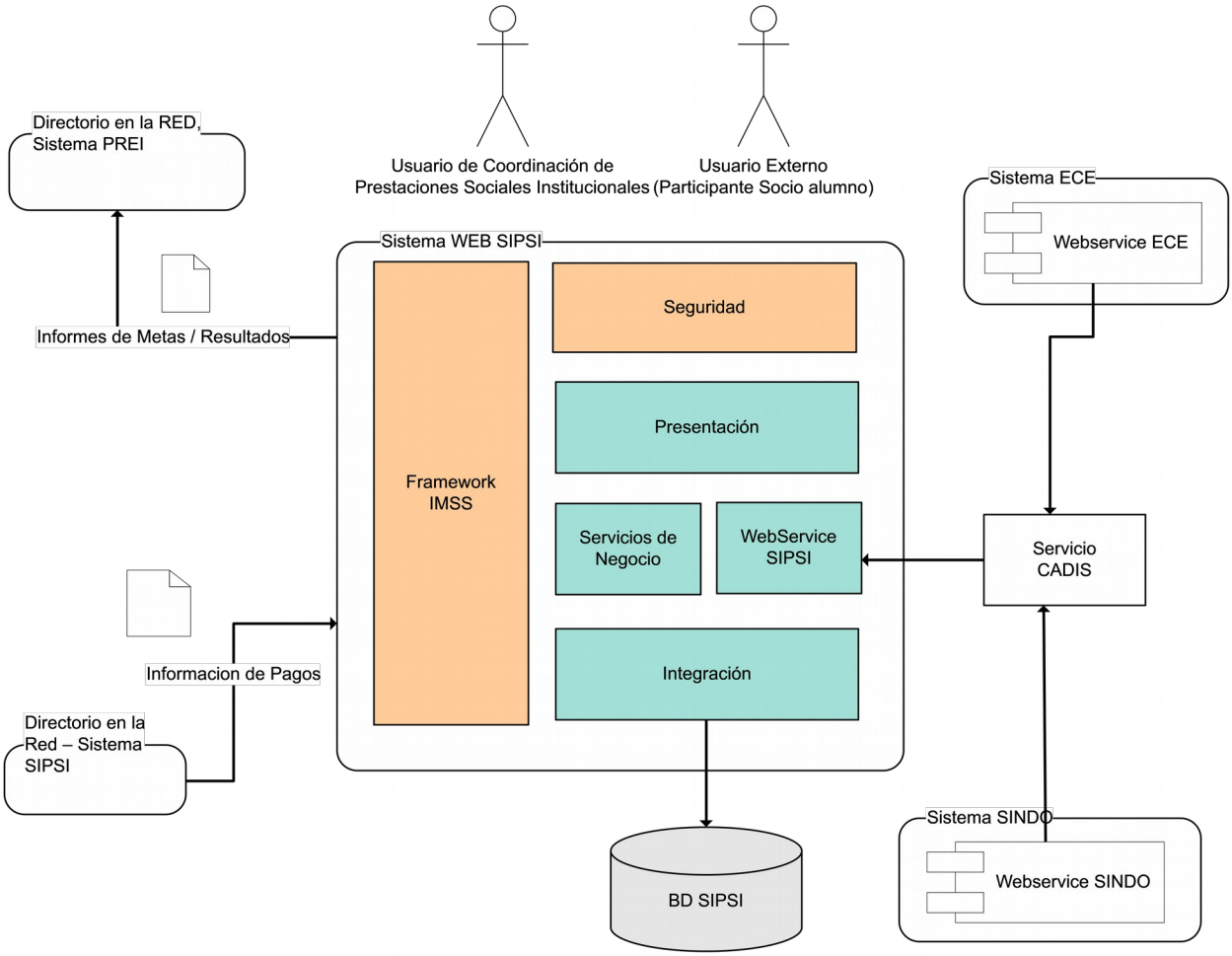
La solución se desarrolló utilizando cinco proyectos:

- mx.gob.imss.cia.sipsi.dto (*Data Transfer Object*). Este paquete contiene todas las clases que representan básicamente a todas las entidades de la Base de Datos y están organizadas en sub-paquetes (por cada módulo y/o funcionalidad). Estas clases son utilizadas para comunicar la información entre componentes de diferentes niveles de la arquitectura (DTO) Por ejemplo entre componentes de la capa de Datos (Componentes DAO) y componentes de la Capa de Negocios.

- mx.gob.imss.cia.sipsi.integration.dao (Data Access Object). Este paquete contiene sub-paquetes según los módulos funcionales existentes y contienen las clases que manejan el acceso a los datos. Estas clases implementan la manipulación de entidades de la base de datos utilizando el framework Hibernate para ser mapeados como objetos.
- mx.gob.imss.cia.sipsi.business. Este paquete contiene sub-paquetes por cada módulo funcional. Las clases contenidas en estos paquetes manejan la lógica de negocio del sistema y la interacción con los componentes DAO e Interfaces.
- mx.gob.imss.cia.sipsi.presentation . En este paquete se encuentran todas las clases de la capa de presentación clasificados en sub-paquetes de acuerdo a los módulos existentes en el sistema. Entre estos paquetes se encuentran aquellos que contienen los controles de usuario.
- mx.gob.imss.ctsm.sipsi.dto.seguridad.Usuario. Esta clase contiene la información del usuario interno de SIPSI que ingresó al sistema. Además tendrá datos propios de su conexión. Éste residirá en la sesión de la aplicación y éste será requerido por cada servicio (funcionalidad) al que el usuario acceda para validar sus credenciales, así como su nivel en la jerarquía de prestaciones sociales para acceder a la información jerárquica

A continuación se presenta cada una de las capas que conforman la arquitectura principal del sistema SIPSI. Para cada capa, se incluye una sección con su nombre y los componentes contenidos en ella.

El siguiente diagrama muestra las capas del sistema y cómo éstas se relacionan con el exterior:



SIPSI emplea el estilo arquitectónico de componentes distribuidos (capas), el cual permite organizar y administrar los componentes de acuerdo a la función que cumplen. Las capas permiten diferenciar y separar funciones únicas de las aplicaciones. La complejidad del almacenamiento de datos no tendrá influencia en la forma en que se ejecuta la lógica de negocios, la cual sólo dependerá y se originará en las necesidades del negocio.

Esto es, el acceso a los datos y la lógica de negocios se tratan en distintas capas. Esto permite un diseño modular de capas individuales, las cuales deben presentar interfaces claras, de forma tal que puedan ser mantenidas y desarrolladas de manera independiente. Las relaciones que mantienen los componentes de distintas capas deben tener una alta cohesión y un bajo acoplamiento. Un componente realiza un conjunto único de funciones encapsuladas, exponiendo únicamente sus interfaces, con el objetivo de favorecer el entendimiento, facilidad de mantenimiento y escalabilidad de la solución.

El sistema SIPSI está compuesto por varios módulos que son utilizados por todos los usuarios de la Coordinación de Prestaciones Sociales Institucionales. Las funcionalidades que conforman los subsistemas o módulos están detalladas a continuación:

Seguridad. Módulo que administra la seguridad del sistema SIPSI. Se combinan dos niveles de seguridad. La seguridad por perfiles y la seguridad de información por niveles jerárquicos.

Administración. Este módulo agrupa las funcionalidades que son tareas ejecutadas por el administrador del sistema SIPSI o por usuarios que cuenten con el perfil correspondiente para ejecutarlas. Entre estas funcionalidades tenemos:

- El proceso de calendario de labores
- Mantenimiento de parámetros del sistema y catálogos estáticos
-

Organización. Este módulo agrupa la funcionalidad que gestiona las entidades que representan la organización de la coordinación de prestaciones sociales institucionales así como sus recursos asociados

Planeación y Programación de Actividades. Este módulo contiene las funcionalidades asociadas al proceso de programación anual de actividades. En nivel central, basados en el análisis de resultados, se elaboran y autorizan los criterios técnicos para la programación anual de actividades, los cuales son enviados a las delegaciones y UOPSIs para su análisis, captura, integración y aprobación del Programa Anual Delegacional. Finalmente esta información registrada en delegaciones y UOPSI´s es enviada a normativo central para la revisión y autorización del programa anual.

Inscripción y cobro de cuotas de recuperación por servicios. Este módulo que es utilizado principalmente a nivel de las Unidades Operativas, permitirá atender los requisitos de inscripción y cobro, tanto para cursos, como ligas deportivas en el caso de actividades complementarias y uso de instalaciones. La inscripción y cobro de servicios, generará una ficha de pago con la cual el socio-alumno podrá efectuar el pago en el banco del monto detallado en la ficha de pago.

Seguimiento mensual. Este módulo permite recabar la información necesaria para el control de asistencias a cursos, actividades complementarias así como el registro de aforos en teatros.

Informes de resultados. Este módulo permitirá generar los reportes de los resultados de la operación en comparación con la estructura programática e información presupuestal, los reportes de metas que solicita PREI de forma mensual y en general de aquella información que da cuenta de la gestión de los procesos de prestaciones sociales como son: informe de metas mensual, anual e información directiva. Estos reportes (informes) son texto plano con formato de acuerdo a estructuras establecidas para PREI.

Conciliación bancaria: Este módulo permite realizar el proceso de conciliación bancaria, derivada del cobro de cuotas en forma automática para lo cual utilizará la información de pagos a las cuentas del IMSS, la cual será enviada por el banco diariamente.
Derivado de este proceso se generan otros servicios que permiten completar la conciliación, que se efectuará a partir del reporte de inconsistencias de las cuotas NO conciliadas.

Cédula de factores de riesgo Es un servicio de encuestas orientado a determinar los riesgos de salud de los usuarios socios-alumnos de prestaciones sociales (alumnos o participantes de actividades complementarias).

Promociones: Este módulo permite la administración de promociones a actividades, curso, eventos y actividades de teatros, la cual es enviada vía correo electrónico a los participantes (socios-alumnos) registrados en la base de datos del sistema de acuerdo a sus preferencias obtenidas mediante encuestas y el análisis de los datos de inscripciones anteriores.

Actualmente el sistema se encuentra en ambiente productivo brindando servicio a más de 500,000 usuarios entre socios-alumnos y usuarios de la Coordinación de Prestaciones Sociales Institucionales en todo el territorio nacional mientras se desarrolla y prepara una nueva liberación para satisfacer nuevos requerimientos y solicitudes de cambio.

CONCLUSIONES

Si pudiera resumir lo que la experiencia de ejercer mi carrera en la programación y la administración me ha enseñado podría resumirse a la frase “No trabajes duro, trabaja inteligente”. Sin duda alguna no importa el tiempo que tome el pensar un proyecto y diseñarlo, es natural que estas fases del proyecto tomen tiempo, en medida de lo bien que se haya pensado y diseñado el proyecto y el plan de trabajo, el tiempo de desarrollo será óptimo, los resultados serán satisfactorios, la cantidad de problemas después de la implementación será mínima y la extensibilidad del proyecto será mejor.

En lo particular dividiría las buenas prácticas en dos, la primera serían las buenas prácticas al programar, es decir adquirir una técnica que nos permita hacer un trabajo eficiente, extensible y confiable. La segunda, por otro lado, serían las buenas prácticas al administrar, es decir, tomar decisiones con información y afrontar los retos con mesura.

Desarrollar un sistema es abstraer la realidad a un ambiente virtual, es por esto que es muy importante entender las necesidades, variables, entradas, salidas y características generales de lo que se pretende abstraer. Es muy conveniente involucrar al cliente en las fases del desarrollo de manera que él sea parte de la construcción del producto

Es muy importante entender todo de manera genérica, ser muy consciente de que lo que funciona para un cliente puede funcionar para otros, considerar al cliente como un parámetro más del sistema. Desarrollar componentes bien documentados y que sean fácilmente integrables permitirá ahorrar tiempos y trabajo para futuros desarrollos, también permitirá la conectividad con otros sistemas, aunque estos estén desarrollados en otras tecnologías y permite la mejora y actualización de manera modular, dando lugar a sistemas que están divididos en partes a las que se les puede dar mantenimiento y actualizar de manera independiente.

BIBLIOGRAFÍA

Visual Studio :.NET Framework 3.5 para profesionales

Maximiliano Firtman, Leonardo Natale.

Alfaomega, 2010.

ASP.NET MVC framework preview

Steven Sanderson.

Apress, 2008.

Calidad en el Desarrollo de Software

Guillermo Pantaleo

Alfaomega Grupo Editor 2011

UML Applied, A .NET Perspective

Martin L. Shoemaker

Apress 2004