UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INFORME DE TRABAJO PROFESIONAL

AUTOMATIZACIÓN DE ACTIVIDADES Y PROCESOS ADMINISTRATIVOS DE UN ÁREA DE UNA INSTITUCIÓN FINANCIERA

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:

ESCÁRCEGA GÓMEZ EDUARDO JESÚS

DIRECTOR:

ING. CARLOS ALBERTO ROMÁN ZAMITIZ

CIUDAD UNIVERSITARIA MAYO 2015

Agradecimientos

A mi madre,

Por su apoyo incondicional, su ejemplo, su dedicación, sus enseñanzas, su esfuerzo que me ha brindado en toda mi vida.

A la Universidad Nacional Autónoma de México, Facultad de Ingeniería y Profesores

Por haberme abierto sus puertas desde el momento en que ingresé a la carrera y por todo el conocimiento que me brindaron.

Espero algún día retribuírselos.

A mis compañeros y amigos de la Facultad

Por los momentos de apoyo y momentos agradables.

Al Ing. Carlos Alberto Román Zamitiz

Por su apoyo en el desarrollo del presente informe

A German Óscar Caballero Rodríguez

Por sus enseñanzas y apoyo laboral.

ÍNDICE

Agradecimientos ii
ÍNDICE iii
Índice de imágenes vi
Índice de tablas vii
INTRODUCCIÓN 1
CAPÍTULO 1: 2
ORGANIZACIÓN Y ACTIVIDAD PROFESIONAL 2
1.1 Planeación estratégica 3
1.1.1 Misión 3
1.1.2 Estructura 3
1.2 Subdirección de inteligencia estratégica 4
1.2.1 Participación profesional y actividades asignadas 5
CAPÍTULO 2: 7
MARCO TEÓRICO 7
2.1 Software, sistemas, ingeniería en sistemas. 8
2.1.1 Software 8
2.1.2 Sistemas 8
2.1.3 Ingeniería en sistemas 8
2.2 Ingeniería de software 9
2.2.1 Antecedentes 9
2.2.2 Definición 10
2.2.3 Importancia de la ingeniería de software 10
2.2.4 Objetivos de la ingeniería de software 11
2.2.5 Proyectos de software 11
2.3 Ciclo de vida del software 11
2.3.1 Modelos de ciclo de vida del software 12
2.3.1.1 Modelo en cascada o línea secuencial 12
2.3.1.2 Modelo en V 13
2.3.1.3 Modelo iterativo 15

2.3.1.4 Modelo de desarrollo incremental 16
2.3.1.5 Modelo en espiral 16
2.3.1.6 Modelo de prototipos 18
2.4 Metodologías de desarrollo de software 19
2.4.1 Definición y objetivo de metodología 19
2.4.2 Definición de proceso 20
2.4.3 Metodologías tradicionales 20
2.4.4 Metodologías ágiles 20
2.4.4.1 Surgimiento 20
2.4.4.2 Definición y objetivo 21
2.4.4.3 Manifiesto ágil 21
2.4.4.4 SCRUM 22
2.4.4.5 ¿Usar metodologías tradicionales o ágiles? 26
2.4.4.6 Características determinantes a considerar antes de elegir una metodología de desarrollo de software. 27
2.5 Validación y verificación de software 28
2.5.1 Conceptos 28
Objetivo de verificación & validación 28
2.6 Tecnología 30
2.6.1 Bases de datos 30
2.6.2 SQL Server Standard Edition 2012 31
2.6.3 Lenguaje de desarrollo C# 32
2.6.4 Framework .NET 34
2.6.5 Visual Studio 35
2.6.6 Desarrollo para Office mediante Visual Studio Tools for Office (VSTO) 37
CAPÍTULO 3: 40
SISTEMA CONTROL VACACIONAL MEDIANTE LA METODOLOGÍA DE DESARROLLO ÁGIL SCRUM 40
3.1 Problemática 41
3.2 Objetivo del proyecto 41
3.3 Alcance del proyecto 42
3.4 Metodología Scrum 43

3.4.1 Pila del producto 43
3.4.2 Planificación del Sprint 49
3.4.3 Pila del Sprint 49
3.4.4 Desarrollo 50
CAPÍTULO 4: 65
PRUEBAS, IMPLEMENTACIÓN Y RESULTADOS 65
4.1 Pruebas 66
4.2 Implementación 66
4.3 Resultados 67
CAPÍTULO 5: 68
CONCLUSIONES 68
Referencias 70

Índice de imágenes

Figura 1.1 Estructura organizacional 3
Figura 2.1 Modelo en cascada o lineal secuencial 12
Figura 2.2 Modelo en V 14
Figura 2.3 Modelo iterativo 15
Figura 2.4 Modelo desarrollo incremental....16
Figura 2.5 Modelo en espiral 18
Figura 2.6 Modelo de prototipos 19
Figura 2.7 Componentes de Scrum 23
Figura 2.8 Roles en Scrum 24
Figura 2.9 Proceso Scrum 25
Figura 2.10 Logotipo de SQL Server 2012 31
Figura 2.11 Ejemplo del entorno de SQL Server Management 32
Figura 2.12 Ejemplo del diseñador de formularios en SharpDevelop 33
Figura 2.13 Ejemplo de código en C# con MonoDevelop....33
Figura 2.14 Logotipo de .NET 4.5....35
Figura 2.15 Ejemplo de desarrollo con Visual Studio 2012....36
Figura 2.16 Ejemplo de Add-in para Microsoft Power Point 2010....38
Figura 2.17 Ejemplo de Apps for Office 2013 39
Figura 3.1 Diagrama UML, casos de uso para un empleado....46
Figura 3.2 Diagrama UML casos de uso para administrador....47
Figura 3.3 Diagrama UML de actividades....48
Figura 3.4 Diagrama de la base de datos....51
Figura 3.5 Diagrama de flujo de login de acceso....52
Figura 3.6 Ventana de acceso 52
Figura 3.7 Diagrama de flujo de la solicitud de vacaciones 53
Figura 3.9 Diagrama de flujo de consulta de historial 54
Figura 3.10 Ejemplo de reporte de historial 55
Figura 3.11 Diagrama de flujo de programación de empleados 55
Figura 3.12 Formulario de consulta de la programación de fechas de empleados....56
Figura 3.13 Diagrama de flujo de cancelación de solicitud....56
Figura 3.14 Formulario de cancelación de solicitud vacacional....57
Figura 3.15 Formulario de cancelación de solicitud vacacional....57
Figura 3.16 Diagrama de flujo de consulta de días y periodos del área 58
Figura 3.17 Reporte....58
Figura 3.18 Diagrama de flujo alta de empleados 59
Figura 3.19 Diagrama de flujo baja de empleado y su historial 60
Figura 3.20 Carga de días 60
Figura 3.21 Diagrama de flujo de carga historial vacacional para empleados que se cambian de área....61

Figura 3.22 Diagrama de flujo de la configuración para omitir días feriados en el cálculo de días a elegir por solicitud vacacional 62
Figura 3.23 Formulario de altas, bajas, cambios, captura de días, carga historial y configuración de días feriados. 62
Figura 3.24 Formato de solicitud de vacaciones.....63
Figura 3.25 Add-in sistema control vacacional 63
Figura 3.26 Seguimiento del Sprint.....634
Figura 3.27 Recta Burndown 634

Índice de tablas

Tabla 1.1 Priorización de actividades 1 49

INTRODUCCIÓN

Una institución financiera es aquella que oferta a los clientes la prestación de servicios financieros, dentro de los productos que pueden ofrecer está implícita la ingeniería y tecnología que trabajan de manera conjunta para satisfacer al máximo al cliente y atraer a más clientes.

Muchas de las actividades internas de una institución financiera se realizan de manera manual, el recibir, entregar, tramitar, analizar y difundir información para su adecuado funcionamiento, esto requiere una alta cantidad de tiempo y que puede ser sujeto a errores humanos.

De la misma manera existen empresas que se dedican a ofrecer al cliente soluciones financieras y de crédito, empresas que requieren principalmente realizar reportes y resúmenes de estados mensuales, trimestrales, anuales, etc., los cuales se realizan principalmente en la suite ofimática de Microsoft Office, además dichas empresas administran sus recursos internos a través de la misma suite ofimática.

Sin embargo con el paso del tiempo, la tecnología evoluciona de tal forma que las empresas también deben modernizar el flujo de trabajo y automatizar actividades que beneficien a la empresa y sus empleados y las aplicaciones a diseñar, desarrollar e implementar esté de acuerdo al plan de la empresa.

En el presente informe demuestro aplicar los conocimientos adquiridos en la carrera de Ingeniería en Computación de la Facultad de Ingeniería, en un área de Planeación Estratégica.

En el capítulo 1, muestro brevemente la estructura del área, así su visión, misión, y las actividades cotidianas.

En el capítulo 2, muestro el marco teórico, conocimientos que adquirí en mi etapa como estudiante de la carrera de Ingeniería en Computación, mismos que apliqué en el capítulo 3.

En el capítulo 4, muestro la etapa de pruebas, implementación y resultados del sistema.

Finalmente en el capítulo 5, las conclusiones del proyecto creado.

El desarrollo del proyecto Sistema Control Vacacional incluye una personalización en Microsoft Office Excel realizada a través de Visual Studio Tools for Office, que le brinda al usuario la potencia de .NET y una serie de formularios que le ayudan a visualizar mejor la información de captura, resumen de datos, etc.

CAPÍTULO 1:

ORGANIZACIÓN Y ACTIVIDAD PROFESIONAL

1.1 Planeación estratégica

Planeación estratégica, es una de las áreas que integra la institución financiera a la cual voy a referirme, dicha área se dedica a proveer información financiera cualitativa y cuantitativa de manera oportuna a diversas líneas de negocio y dirección general, afín de que tengan elementos suficientes de juicio que les permita dar seguimiento puntual al desempeño del banco frente a la competencia.

1.1.1 Misión

Proveer a los clientes internos de las diferentes líneas de negocio un servicio puntual y de alta calidad, con soluciones integrales, para aumentar el número de clientes y la rentabilidad de la institución financiera.

1.1.2 Estructura

El área está estructurada en subdirecciones adjuntas (divisiones) que atienden a los diferentes tipos de clientes internos (véase Figura 1.1).

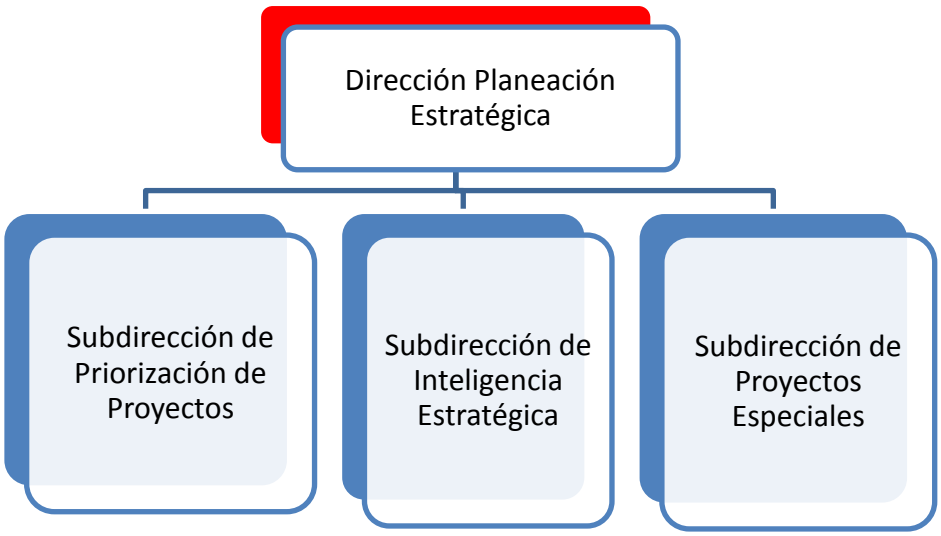

Figura 1.1 Estructura organizacional

1.2 Subdirección de inteligencia estratégica

La Subdirección de Inteligencia Estratégica, desempeña un papel fundamental en todo el Grupo Financiero, ya que se encarga de suministrar servicios e información a las demás divisiones de ventas y departamentos a efectos de su buen funcionamiento.

Destacan actividades como:

- Elaboración de análisis de mercado (Grupo Financiero vs Competencia) para la toma de decisiones.
- Coordinación con otras áreas para la preparación de información financiera del Grupo Financiero
- Entrega y recepción de información de la competencia por parte de las fuentes regulatorias.
- Elaboración de reportes, bases y análisis (semanales, mensuales, trimestrales).
- Análisis e investigaciones especiales (por pedido).

Además de administrar los recursos internos, humanos, financieros presupuestales y materiales del área.

Dentro de las actividades estratégicas que se realizan en dicha subdirección son:

- La auditoría de ventas, como la entrega de estados de resultados de ventas de manera mensual, trimestral, a efecto de medir el crecimiento de las diversas líneas de negocio.
- Asignación de metas a cumplir en un año fiscal para ejecutivos, y su respectivo seguimiento.
- Resúmenes e histórico de ventas por sector y segmentos.
- La planeación, asignación y control de presupuesto para la creación de nuevos productos o la mejora de los existentes y para las demás direcciones.
- Estimaciones de cierres de año y proyecciones de plan de los siguientes años fiscales.
- Elaboración de ingresos y egresos de las líneas de negocio.

Dentro de las actividades administrativas la subdirección realiza:

- Gestiona los recursos humanos de toda el área.
- Custodias de expedientes del personal (curricula, descripciones de puesto, evaluaciones, incapacidades, formatos vacacionales, cambios de área, actas de equipo, entre otros)
- Logística e inmobiliaria

- Solicitud, compra, revisión, renovación, cambio de equipos de cómputo, telefonía fija CISCO, móvil, plan de datos, software, creación de cuentas de correo electrónico, correo genérico.

En el área se realizan nuevos sistemas en sustitución de macros mismos que facilitan los procesos y actividades administrativas, estratégicas, reduciendo tiempo en realizar las actividades de manera manual, destacando el uso de tecnología de Microsoft, como Visual Studio Tools for Office, con lo cual se realizan aplicaciones para la suite ofimática de Office, escritas en el lenguaje de programación C# 5.0 y aprovechando .NET 4.0, entregando a los usuarios “add-ins”, dentro de un software que ya conocen, reduciendo al mínimo la curva de aprendizaje del sistema en cuestión.

También existen planes e iniciativas para crear desarrollos web para automatizar actividades que por ahora son manuales y la información se almacena y procesa principalmente con la herramienta de Microsoft Office Excel.

1.2.1 Participación profesional y actividades asignadas

El Grupo Financiero, cuenta con su respectiva área de sistemas, sin embargo dicha área se dedica a la construcción de productos para clientes de las diversas líneas de negocio, productos con fines de regulación por entidades y organismos como Banco de México (Banxico) y la Comisión Nacional Bancaria y de Valores (CNBV), y proyectos especiales.

De esta manera priorizan esos proyectos y aquellos proyectos que son de uso interno de las áreas de Staff, para automatizar actividades manuales y actividades cotidianas, son puestos en un segundo plano.

Por lo que fui contratado dentro de Planeación Estratégica con fines de aprender gradualmente las actividades administrativas y estratégicas a efecto de automatizar la mayor carga de actividades posibles, utilizando tecnología que le sea útil al área.

Ingresé a la dicha área principalmente para el desarrollo de aplicaciones para Office, utilizando Visual Studio Tools for Office, desarrolladas utilizando el lenguaje C# y el Framework .NET, así como desarrollos web y creación de bases de datos, comencé a aprender del sitio de Microsoft Virtual Academy del curso de .NET en donde existe el módulo Visual Studio Tools for Office, de la misma manera tomando material existente en la documentación oficial de Microsoft Developer Network (MSDN), que es una comunidad de programadores a nivel mundial de las diversas tecnologías creadas por Microsoft, donde los desarrolladores comparten código y suministran apoyo en preguntas y el blog Channel 9 de Microsoft, donde principalmente existen video tutoriales.

Para familiarizarme con las actividades financieras y administrativas del área, para su posterior automatización, mis compañeros del área, me explicaron gradualmente el proceso de cada una de las tareas administrativas/ estratégicas que me fueron asignadas desde un inicio, como:

- Administración y control vacacional.
- Elaboración de indicadores de rendimiento y comparación de estados financieros contra otros años fiscales.
- Preparación de información financiera y contable para analizar la captación y colocación.
- Elaboración de reporte semanal de seguimiento del Grupo Financiero vs la Competencia principal, a través del intercambio de información bancaria.
- Elaboración del reporte de captación y colocación de sucursales por territorio, zona, municipio, sucursal, del Grupo frente a la competencia principal.
- Entregar información a clientes internos.
- Capacitación del personal de los sistemas elaborados.
- Realizar múltiples reportes procesando información financiera, a través de diversas fuentes para su entrega oportuna y posterior análisis.
- Generar información confiable, oportuna, a efecto de que Planeación Estratégica analice y determine crecimientos, y participación de mercado en cada una de las áreas del Grupo Financiero.

CAPÍTULO 2:

MARCO TEÓRICO

2.1 Software, sistemas, ingeniería en sistemas.

2.1.1 Software

Software es el conjunto lógico de la computadora que cumple específicas funciones y que gracias a ellas es posible la realización de actividades, estas aplicaciones generan una interfaz o sistema de comunicación entre el usuario y la computadora.

2.1.2 Sistemas

Un sistema es un conjunto de elementos de hardware, software, personas, procedimientos, herramientas, organizados de tal manera que lleven a cabo un objetivo en común.

2.1.3 Ingeniería en sistemas

La ingeniería en sistemas, de acuerdo con la definición de Defense Systems Management College 1986, define el plan para gestionar las actividades técnicas del proyecto, identifica el ciclo de desarrollo y los procesos que serán necesarios aplicar. Desde la Ingeniería de sistemas se desarrolla la línea base técnica para todo el desarrollo, tanto de hardware como de software.

Las funciones que corresponden a la ingeniería en sistemas son:

- ✓ **Definición del problema:** Determina las expectativas hacia el producto, necesidades y restricciones obtenidas y analizadas en los requisitos del sistema. Trabaja cerca y constantemente con el cliente para establecer las necesidades operacionales del requerimiento.
- ✓ **Análisis de la solución:** Determina las opciones posibles para satisfacer los requisitos y necesidades del cliente así como sus restricciones. Estudian y analizan las soluciones encontradas y eligen la mejor opción.
- ✓ **Planificación de procesos:** Determina grupos de tareas técnicas que se deben realizar, el esfuerzo requerido para cada una, su prioridad y los riesgos que implican para el proyecto.
- ✓ **Control de procesos:** Determina los métodos para controlar las actividades técnicas del proyecto y los procesos; la medición del progreso, revisión de los productos intermedios y ejecución de las acciones correctivas, cuando corresponda.
- ✓ **Evaluación del producto:** Determina la calidad y cantidad de los productos elaborados, a través de evaluaciones, inspecciones, pruebas, etc.

2.2 Ingeniería de software

2.2.1 Antecedentes

El software rápidamente ha evolucionado en los últimos años, a comparación de otras actividades profesionales que han tomado mayor tiempo para evolucionar y generar conocimiento.

La industria del hardware también se ha desarrollado de manera prominente e importantemente acelerada, como una consecuencia de la ley de Moore, el incremento constante de capacidad de operación y almacenamiento de elementos, su miniaturización y la reducción de costos de fabricación, ha dado como resultado computadoras y dispositivos móviles de alto rendimiento y capacidad que permiten ejecutar cualquier actividad y tarea. En la actualidad ya no están reservadas las computadoras como lo era antes para el ejército, institutos de investigación, la banca, o empresas de un tamaño colosal, ahora están presentes en todos lados, ofreciendo soluciones complejas en pequeños periodos de tiempo a múltiples enfoques, áreas y demás.

En un momento determinado se hizo un mayor hincapié en el desarrollo de hardware mientras que el software creció de tal manera que se hizo desorganizada su producción, sin el uso de estándares, lo que provocó la llamada “Crisis del Software”, que fue un término acuñado en 1968 en la primera conferencia sobre desarrollo de software de la organización NATO, el problema consistía en diversos factores, los lenguajes de programación eran de bajo nivel, los sistemas no estaban documentados, o la documentación existente era pobre, escasa, simple, el código de los programas era ofuscado, solo los desarrolladores iniciales sabían cómo realizar un mantenimiento y actualizaciones al producto y cuando nuevos desarrolladores eran integrados al proyecto, difícilmente entendían el código fuente, los proyectos desbordaban los presupuestos planteados al inicio, lo que llevaba a las compañías a gastar excesivamente en la fabricación y producción de sistemas debido al retraso en la entrega, la mala administración, la inexistente manera de llevar el cálculo del costo en las actividades y procesos, todos estos factores perjudicaban en la entrega final del producto al cliente, este estaba insatisfecho y/o molesto porque el producto desarrollado no cumplía con las características y requerimientos que se habían solicitado a los desarrolladores, la calidad era cuestionable, y muchas veces no se entregaba a tiempo el producto o el precio final era muy alto.

Desde 1968 hasta la fecha, poco a poco las empresas, institutos de investigación de diversas universidades, organismos de estandarización y los mismos desarrolladores empezaron a buscar mayor productividad, eficiencia operativa, organización y nuevas maneras de trabajar haciendo énfasis en asegurar la calidad del software para el cliente y de esta manera definir pautas y estándares universales para la producción y mantenimiento del software.

Los esfuerzos finalmente se han encaminado en tres principales direcciones:

1. Identificación de factores clave que determinan la calidad del software
2. Identificación de los procesos necesarios para producir software y mantenerlo.
3. Estructuración y desarrollo de la base de conocimiento necesaria para la producción y el posterior mantenimiento del software.

Surge así, la ingeniería de software, nacen estándares, conceptos de ciclo de vida de software y metodologías para administrar de principio a fin un proyecto de software.

Los estándares y metodologías son útiles porque:

- ✓ Agrupan lo mejor y más apropiado de las buenas prácticas y usos del desarrollo de software.
- ✓ Proporcionan un marco para implementar procedimientos de aseguramiento de la calidad del software.
- ✓ Proporcionan continuidad y entendimiento entre el trabajo de personas y organizaciones distintas.

2.2.2 Definición

La ingeniería de software es definida originalmente en 1968 en la conferencia de NATO, Fritz Baver la define como “El establecimiento y uso de principios de ingeniería para obtener software económico que trabaje de forma eficiente en máquinas reales”.

La ingeniería de software es entonces la rama de la ingeniería que se encarga de diseñar, desarrollar, operar, controlar, mantener productos de software de manera cuantificable, y sistemática.

2.2.3 Importancia de la ingeniería de software

Actualmente organismos gubernamentales, instituciones y cualquier empresa hace uso de sistemas para su operación diaria interna, para automatizar actividades manuales, reducir procesos y flujos de operación en diversas áreas como la educación, medicina, finanzas, automotriz, aeroespacial, telefonía, civil, energética y muchas otras más.

Todas las empresas que desarrollan software se rigen mediante metodologías tradicionales o agiles y adaptaciones que cada empresa realiza y genera su propio marco de trabajo.

La ingeniería de software desempeña un papel fundamental en el desarrollo de sistemas, no solo se debe desarrollar un aplicativo para resolver un problema determinado, también es necesario una buena administración a lo largo de todo el ciclo de desarrollo de un producto.

Mediante la ingeniería de software es posible analizar, diseñar, desarrollar, validar programas de manera adecuada, oportuna, correcta, organizada. Esto es desde la concepción de la idea inicial por parte del cliente, el análisis y diseño del producto a raíz de la toma de

requerimientos, desarrollo del programa, validación, entrega e implementación del producto y su posterior mantenimiento.

2.2.4 Objetivos de la ingeniería de software

La ingeniería del software tiene como objetivo la producción de sistemas de software asegurando la calidad cumpliendo con los requisitos y necesidades del cliente, el bajo costo y entregando el producto a tiempo.

2.2.5 Proyectos de software

Un proyecto es una secuencia de actividades, las cuales tienen que desarrollarse en un determinado orden, tienen un objetivo que es entregar al cliente un producto funcional que automatice alguna tarea, estas actividades deben ser concluidas en un tiempo determinado dentro de un presupuesto planificado, siguiendo una metodología que puede ser ágil o tradicional, de acuerdo a un conjunto de especificaciones y como resultado tener un producto funcional entregable al cliente.

2.3 Ciclo de vida del software

Es el conjunto de fases por las que pasa un sistema desde su concepción inicial, pasando por el desarrollo de la aplicación, lanzamiento o implementación, su posterior mantenimiento y actualización, hasta que el software es retirado o reemplazado por uno nuevo.

Este describe el desarrollo de software desde la fase inicial hasta la final.

Los procedimientos que un ciclo de vida debe tener son:

- ✓ **Definición de los objetivos**: Para que se va a hacer, definir el resultado del proyecto, a donde se pretender llegar.
- ✓ **Análisis de requerimientos:** Realizar la toma de requerimientos del cliente, y estudiarlos a detalle.
- ✓ **Diseño general:** Requisitos globales de la aplicación que se pretende desarrollar como breve bosquejo.
- ✓ **Diseño a detalle:** Esto es la definición precisa de cada actividad a realizar en el desarrollo del sistema.
- ✓ **Desarrollo:** Es aquí donde a través de herramientas de programación y lenguajes se crearan las funciones definidas durante la etapa de diseño de la solución.
- ✓ **Pruebas unitarias:** Es el procedimiento donde cada actividad o parte del sistema o aplicación es sometida a un estudio a efecto de garantizar que se realizó en relación a las especificaciones y requerimientos del cliente.

- ✓ **Integración:** Garantizar que todos los módulos se comuniquen y se armen en la aplicación.
- ✓ **Mantenimiento:** Posterior a la entrega suministrar de actualizaciones futuras a la aplicación.

2.3.1 Modelos de ciclo de vida del software

2.3.1.1 Modelo en cascada o línea secuencial

Es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediata anterior.

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida (véase Figura 2.1).

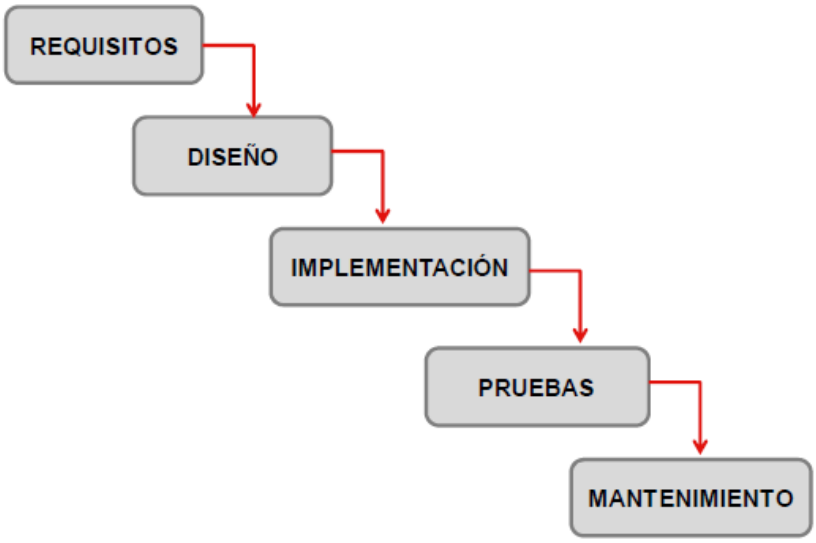

Figura 2.1 Modelo en cascada o lineal secuencial

Ventajas

Apropiado en general, para proyectos estables (especialmente los proyectos con requisitos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación. Funciona bien para proyectos pequeños donde los requisitos están bien entendidos y se tenga la certeza de que no van a cambiar.

Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.

Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una a la vez.

Desventajas

Es muy difícil que las características del proyecto sean tales para llevar este modelo a la práctica, puesto que los requerimientos cambian constantemente. Difícilmente un cliente va a establecer al principio todos los requisitos necesarios, generalmente el cliente introduce o elimina requerimientos a lo largo del desarrollo de una aplicación por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.

El cliente al estar cambiando, introduciendo o eliminando requisitos, es difícil que para los desarrolladores tengan un control de las actividades, lo que lleva al caos al adoptar ese modelo.

2.3.1.2 Modelo en V

Este modelo tiene como características esencial que las pruebas del software deben iniciarse tan pronto como sea posible en el ciclo de vida. También muestra que las pruebas no son sólo una actividad basada en la ejecución.

El modelo en V es un modelo que ilustra cómo las actividades de prueba (verificación y validación) se pueden integrar en cada fase del ciclo de vida. Dentro del modelo en V, las pruebas de validación tienen lugar especialmente durante las etapas tempranas, por ejemplo, revisando los requisitos de usuario y después durante las pruebas de aceptación de usuario (véase Figura 2.2).

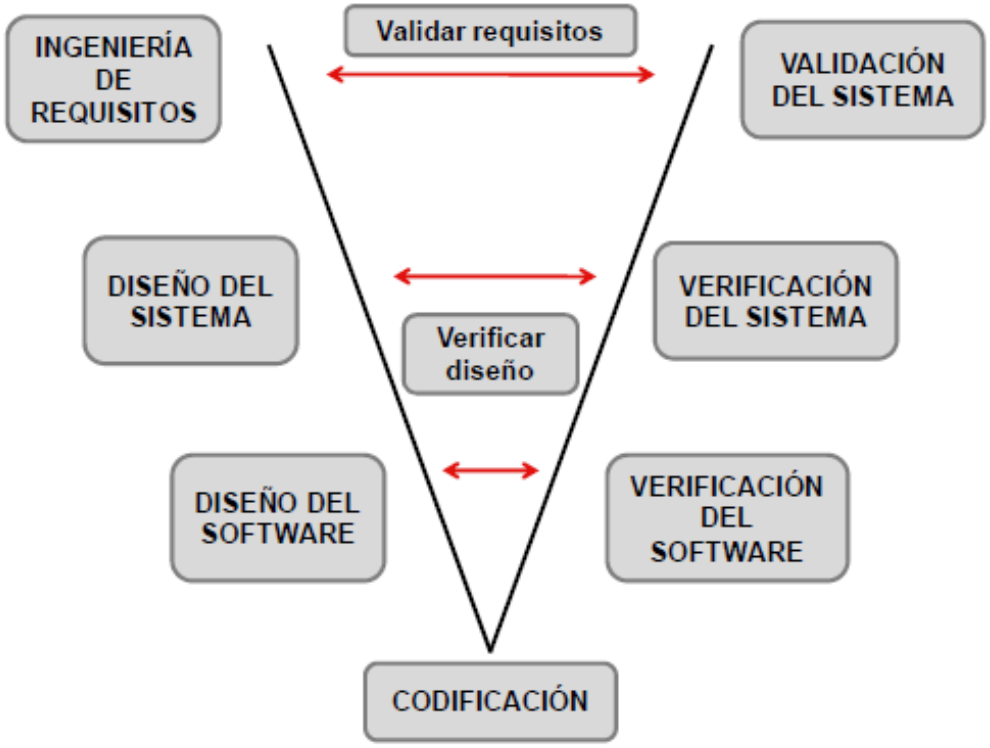

Figura 2.2 Modelo en V

Ventajas

- ✓ Es un modelo fácil de usar, en cada fase hay entregables específicos
- ✓ Tiene alta oportunidad al éxito debido a la realización de pruebas en tempranas etapas del ciclo de vida.
- ✓ Es un modelo que funciona en proyectos pequeños donde los requisitos son entendidos fácilmente.

Desventajas

- ✓ Es un modelo rígido como el modelo en cascada
- ✓ Al ser rígido tiene poca flexibilidad, por lo cual una adaptación o ajuste al alcance es cara en tiempo y dinero.

2.3.1.3 Modelo iterativo

Este modelo tiene como objetivo reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de recogida de requisitos.

Consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto. El cliente es quien después de cada iteración evalúa el producto y lo corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente (véase Figura 2.3).

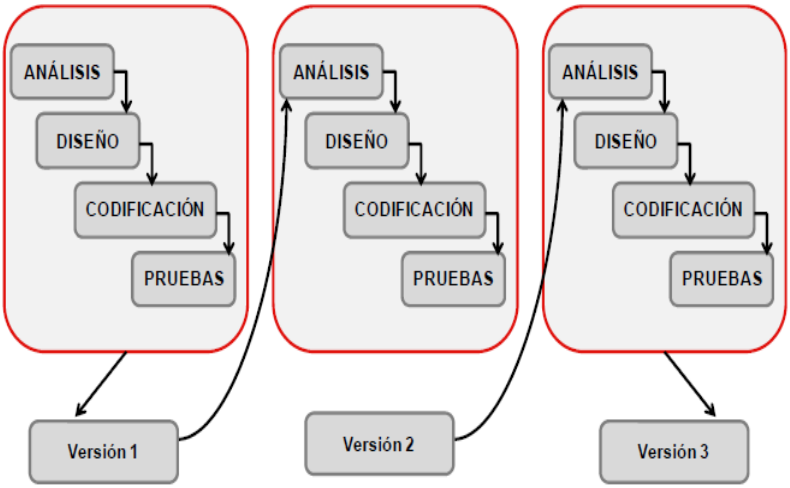

Figura 2.3 Modelo iterativo

Este modelo se utiliza principalmente en aquellos proyectos en los que los requisitos no están claros por parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y de esta manera conseguir la conformidad del cliente.

Ventajas

- ✓ No hace falta que los requisitos estén totalmente definidos al inicio del desarrollo, sino que se pueden ir refinando en cada una de las iteraciones a través de la presentación de prototipos al cliente, esto hace que se reduzca la incertidumbre al desarrollar.
- ✓ Es posible gestionar las entregas de manera más fácil.
- ✓ Permite gestionar mejor los riesgos.

Desventajas

Por parte del cliente, al no tener claramente los requisitos definidos desde un principio, puede verse también como un inconveniente ya que pueden surgir problemas relacionados con la arquitectura.

2.3.1.4 Modelo de desarrollo incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa poco a poco. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software (véase Figura 2.4).

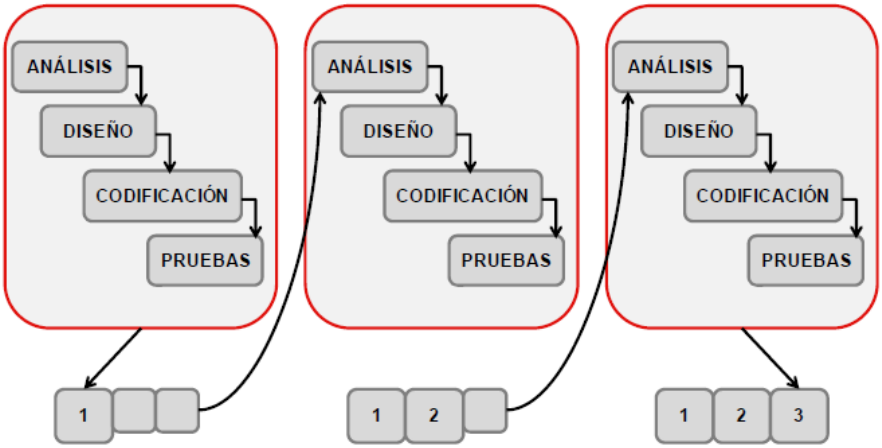

Figura 2.4 Modelo desarrollo incremental

Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

Ventajas

- ✓ Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.
- ✓ Es un modelo más flexible, por lo que se reduce el coste en dinero y tiempo en el cambio de alcance y requisitos.
- ✓ Es fácil probar y depurar en una iteración más pequeña
- ✓ Es más fácil gestionar riesgos.

Desventajas

- ✓ Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.

2.3.1.5 Modelo en espiral

Fue desarrollado por Barry Boehm en 1985, Las actividades de este modelo se conforman en una espiral, cada bucle representa un conjunto de actividades. Las actividades no están

fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgos, comenzando por el bucle anterior

Anteriormente al riesgo no se le prestaba mucha atención, por lo cual muchos proyectos de software fallaban, a raíz de esto surge el modelo Espiral, el modelo tiene el objetivo principal de generar evaluación y resolver riesgo.

De esta manera se permite tanto a desarrolladores como a clientes detener el proceso cuando se considere conveniente.

Este modelo de desarrollo combina las características del modelo de prototipos y el modelo en cascada. El modelo en espiral está pensado para proyectos largos, caros y complicados, no está hecho para proyectos pequeños.

El modelo se compone de:

1. Determinar o fijar objetivos:
   a. Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
   b. Fijar las restricciones
   c. Identificación de riesgos del proyecto y estrategias alternativas para evitarlos
   d. Hay una cosa que solo se hace una vez: planificación inicial o previa
2. Análisis del riesgo
   a. Se estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.
3. Desarrollar, verificar y validar (probar)
   a. Tareas de la actividad propia y de prueba
   b. Análisis de alternativas e identificación de resolución de riesgos
   c. Dependiendo del resultado de la evaluación de riesgos, se elige un modelo para el desarrollo, que puede ser cualquiera de los otros existentes, como formal, evolutivo, cascada, etc. Así, por ejemplo, si los riesgos de la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos.
4. Planificar
   a. Revisamos todo lo que hemos hecho, evaluándolo y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

Todo el proceso comienza en la posición central, desde allí se mueve en el sentido de las agujas del reloj (véase Figura 2.5).

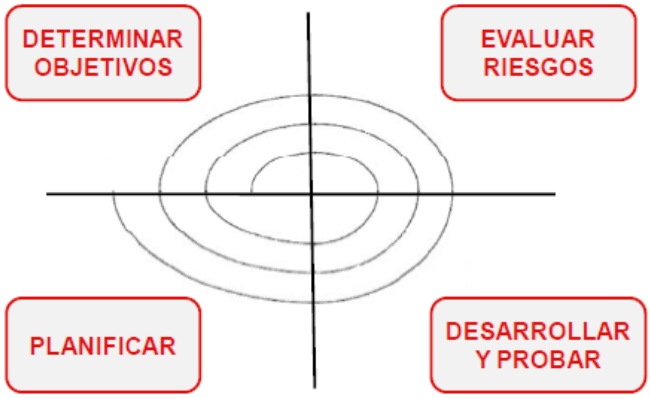

Figura 2.5 Modelo en espiral

Ventajas

- ✓ Reduce riesgos del proyecto
- ✓ Incorpora objetivos de calidad
- ✓ Integra el desarrollo con el mantenimiento

Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.

Desventajas

Es un modelo que genera mucho trabajo adicional. Al ser el análisis de riesgos una de las tareas principales exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos (lo cual es bastante difícil), además de ser costoso monetariamente, no es un modelo que funcione para proyectos pequeños.

2.3.1.6 Modelo de prototipos

El modelo comienza en la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas donde es obligatorio conocer más detalle. Al término de esto, aparece un diseño rápido, que se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo, como una maqueta, lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer (véase Figura 2.6).

Figura 2.6 Modelo de prototipos

Ventajas

- ✓ Ofrece visibilidad del producto desde el inicio del ciclo de vida con el primer prototipo. Esto puede ayudar al cliente a definir mejor los requisitos y a ver las necesidades reales del producto.
- ✓ Permite introducir cambios en las iteraciones siguientes del ciclo. Permite la realimentación continua del cliente.
- ✓ El prototipo es un documento vivo de buen funcionamiento del producto final. El cliente reacciona mucho mejor ante el prototipo al estar en contacto con esa maqueta puede ver de manera tangible lo realmente necesario y lo que no es útil, sobre el que puede experimentar, que no puede hacer en un documento escrito.
- ✓ Este modelo reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios

Desventajas

Es un tipo de modelo que es lento, al estar desarrollando n prototipos para ver realmente qué es lo que necesita el cliente, haciendo que el equipo de desarrollo invierta más tiempo en cada prototipo, y aumenta el costo final del producto.

2.4 Metodologías de desarrollo de software

2.4.1 Definición y objetivo de metodología

Una metodología es la colección de métodos aplicados a lo largo del ciclo de vida de software y su objetivo es abordar, gestionar y administrar un proyecto para llevarlo al éxito.
La metodología comprende los pasos a seguir para poder idear, implementar y mantener el software desde que surge la necesidad de crearlo hasta que es liberado.

Define qué se va a hacer, cómo se hará, y en qué tiempo, durante el desarrollo y posterior mantenimiento del producto.

2.4.2 Definición de proceso

De acuerdo a la definición de ISO 12207/ UNE 77104, un proceso es un conjunto repetitivo de actividades que se relacionan entre sí, que se realizan de manera sistemática mediante las cuales obtenemos un resultado, transformando entradas en salidas.

2.4.3 Metodologías tradicionales

Las metodologías tradicionales tienen la característica de ser estrictas en la especificación precisa de todos los requerimientos de software, modelado, etc., de ahí que reciban el nombre de metodologías “pesadas”, puesto que hacen un mayor énfasis en la planificación total de todo el trabajo a realizar y una vez que todo fue detallado se comienza el ciclo de desarrollo del software, a efecto de obtener un software eficiente y de calidad.

Las metodologías tradicionales se enfocan en el control de los procesos, a través de una exhaustiva definición de roles, actividades, herramientas, entre otros factores, mismos que desembocan en una documentación extensa a lo largo de todo el proyecto.

Estas metodologías tienen como característica principal la difícil adaptabilidad de cambios en los requerimientos del software, lo que eleva el presupuesto inicial del producto.

Las metodologías tradicionales buscan a medida de lo posible la ejecución de un proyecto en el tiempo planificado sin desbordar los costos y recursos estimados al inicio. Durante el desarrollo el líder realiza actividades de seguimiento y vigilancia continua a efecto de evitar desviaciones acerca de lo planificado. En la parte de gestión de riesgo al momento de existir requisitos crecientes, se multiplican las horas de desarrollo, presionando al equipo lo que va afectando su rendimiento, generando desbordamiento de costos y un incumplimiento de fechas.

2.4.4 Metodologías ágiles

2.4.4.1 Surgimiento

Surgen a raíz de la reunión en Salt Lake City en donde 17 críticos convocados por Kent Beck, creador de Extreme Programming, discuten sobre modelos de desarrollo de software. En la reunión se buscaba una alternativa a las metodologías formales, que se consideraban pesadas, caracterizadas por una rigidez en su normatividad, y una fuerte dependencia de planificaciones detalladas previas al desarrollo de los productos.

En la reunión, los integrantes llegan a 4 postulados a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.

- El software que funciona, por encima de la documentación exhaustiva.
- La respuesta al cambio, por encima del seguimiento de un plan.
- La colaboración con el cliente de manera directa, por encima de negociación contractual.

2.4.4.2 Definición y objetivo

Las metodologías ágiles, son un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo, a lo largo de todo el ciclo del proyecto.

El objetivo es esbozar los principios que deben permitir a los equipos desarrollar software de manera rápida y respondiendo a los cambios que puedan surgir a lo largo de un proyecto.

Estas metodologías hacen que el desarrollo sea incremental, cooperativo, sencillo y adaptado a las necesidades que se tengan. Las metodologías ágiles se basan en los 12 principios del manifiesto ágil, el cual surge en la reunión de Salt Lake City.

2.4.4.3 Manifiesto ágil

Los 12 principios del manifiesto ágil son:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación directa con la persona, de manera presencial.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención constante a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

2.4.4.4 SCRUM

Es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software haciendo uso de prácticas iterativas e incrementales, en la que los retos y desafíos no se pueden predecir, tan solo se puede centrar y enfocar en responder rápidamente al cambio producido por requisitos emergentes, maximizando las habilidades de todo el equipo para entregar pronto el software.

Esta metodología de desarrollo ágil nace en 1993, sus creadores fueron Jeff Sutherland y Ken Schwaber, quienes adaptaron los estudios y las prácticas de producción de Hirotaka Takeuchi e Ikujijo Nonaka, al campo del desarrollo de software. A su vez Hirotaka y Nonaka crearon esta metodología alrededor de los 80´s, para proyectos de tecnología.

En 1996 Jeff Sutherland y Ken Schwaber lo presentan como un proceso formal, también para la gestión del desarrollo de software en OOPSLA 96, en el campo del desarrollo de software Scrum está considerado como un modelo ágil por la Agile Alliance.

Es una metodología de carácter adaptable, orientado a personas antes que a los procesos, el desarrollo es iterativo e incremental.

Esta metodología se caracteriza principalmente por:

- Colaboración estrecha y directa con el cliente.
- Predisposición y respuesta al cambio.
- Preferencia por el conocimiento tácito de las personas al explícito de procesos.
- Desarrollo incremental con entregas funcionales frecuentes.
- Comunicación verbal directa entre los implicados en el proyecto.
- Motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso.
- Simplicidad, eliminación de artefactos innecesarios o irrelevantes en la gestión del proyecto.

Estructura

Sprint

Es el periodo de tiempo durante el cual se desarrolla un incremento de funcionalidad, constituye el núcleo de Scrum que divide de esta forma el desarrollo de un proyecto en un conjunto de pequeñas etapas.

Esta iteración es la base del desarrollo ágil, cada ciclo produce un incremento terminado y operativo del producto, mismo que es posible revisar y evaluar.

Su duración es entre 15 a 60 días, aunque es recomendable una duración de 30 días.

Componentes de Scrum

- *Pila de producto o Product Backlog:* Es la relación de requisitos del producto, no es necesario excesivo detalle, el propietario del producto es su responsable y es quien la define. El documento está en constante cambio, es accesible a todas las personas que intervienen en el desarrollo, de esta manera todos pueden contribuir y realizar sugerencias.
- *Pila del Sprint o Sprint Backlog:* Es una lista de actividades comprometidas por el equipo para su realización en el sprint, con nivel de detalle suficiente para su ejecución. Las actividades están asignadas a personas, tienen tiempo estimado y los recursos necesarios.
- *Incremento:* Es una parte del producto desarrollada en un sprint en condiciones de ser usada (véase Figura 2.7).

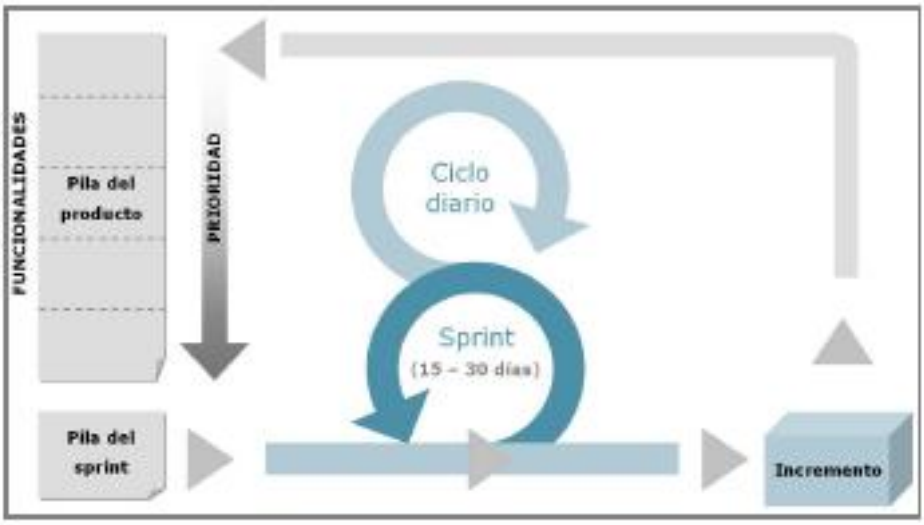

Figura 2.7 Componentes de Scrum

Roles

Existen 4 roles diferentes:

- *Propietario del producto*: Persona que va a determinar las prioridades y define los requisitos del producto, será quien otorgará la financiación del proyecto.
- *Scrum Manager:* Es la persona que gestiona y facilita la ejecución del proceso, a través de coaching de las personas para trabajar de manera colaborativa, es quien modera reuniones, realiza estimaciones, resuelve impedimentos, también es quien garantiza el funcionamiento del producto ante el propietario del producto.
- *Equipo:* Es un conjunto de personas que construyen el producto del cliente en una iteración, es auto-gestionado, auto-organizado, multifuncional.
- *Interesados:* Personas que asesoran y valoran, también aportan sugerencias (véase Figura 2.8).

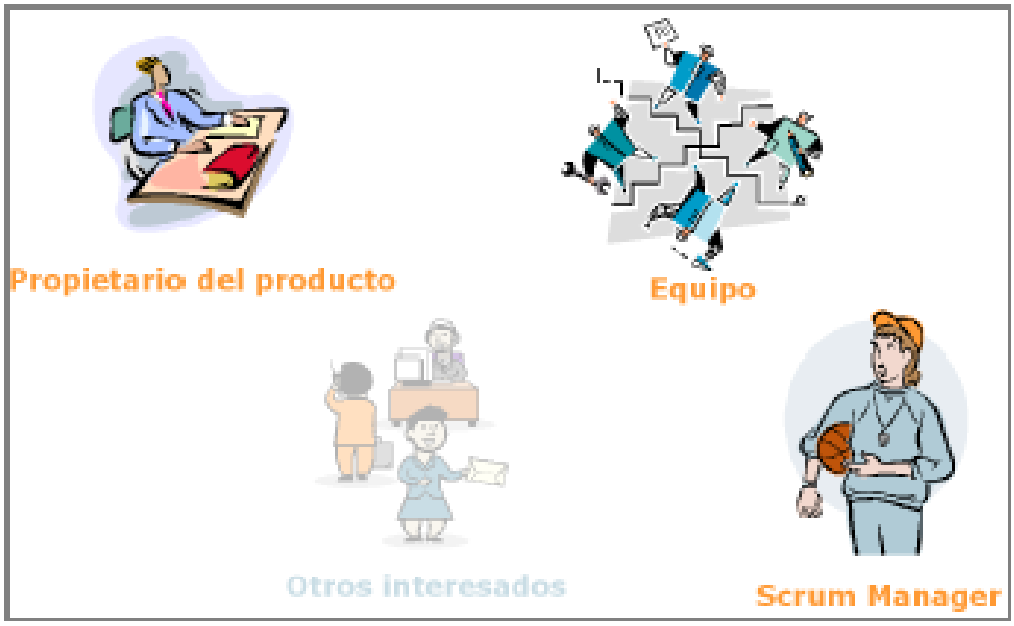

Figura 2.8 Roles en Scrum

Reuniones

- *Planificación del Sprint:* Es una reunión previa al inicio de cada Sprint, que tiene como duración una jornada de trabajo, en la cual, el propietario del producto explica comenzando con la visión general del proyecto, detallando las necesidades y requerimientos. El equipo estima el esfuerzo de los requisitos de mayor importancia y prioriza, al término de esto, se elabora la pila del sprint.
  El Scrum Manager define en una frase el objetivo del Sprint.
- *Reunión diaria:* Es una breve revisión del trabajo realizado el día anterior y el previsto para el día siguiente, tiene una duración de 15 minutos y es moderada por el Scrum Manager, en esta reunión se realizan 3 preguntas:
  - ¿Qué hiciste ayer?
  - ¿Cuál es el trabajo para hoy?
  - ¿Qué necesitas para continuar?

  Se actualiza la pila del sprint.
- *Revisión del Sprint:* Al finalizar cada iteración o ciclo se lleva a cabo una revisión con todas las personas comprometidas e implicadas en el proyecto. Su objetivo es analizar el incremento generado, es una reunión con una duración máxima de 4 horas moderada por el Scrum Manager donde se presenta el incremento, se hace un planteamiento de sugerencias y anuncio del próximo Sprint, si lo hubiera (véase Figura 2.9).

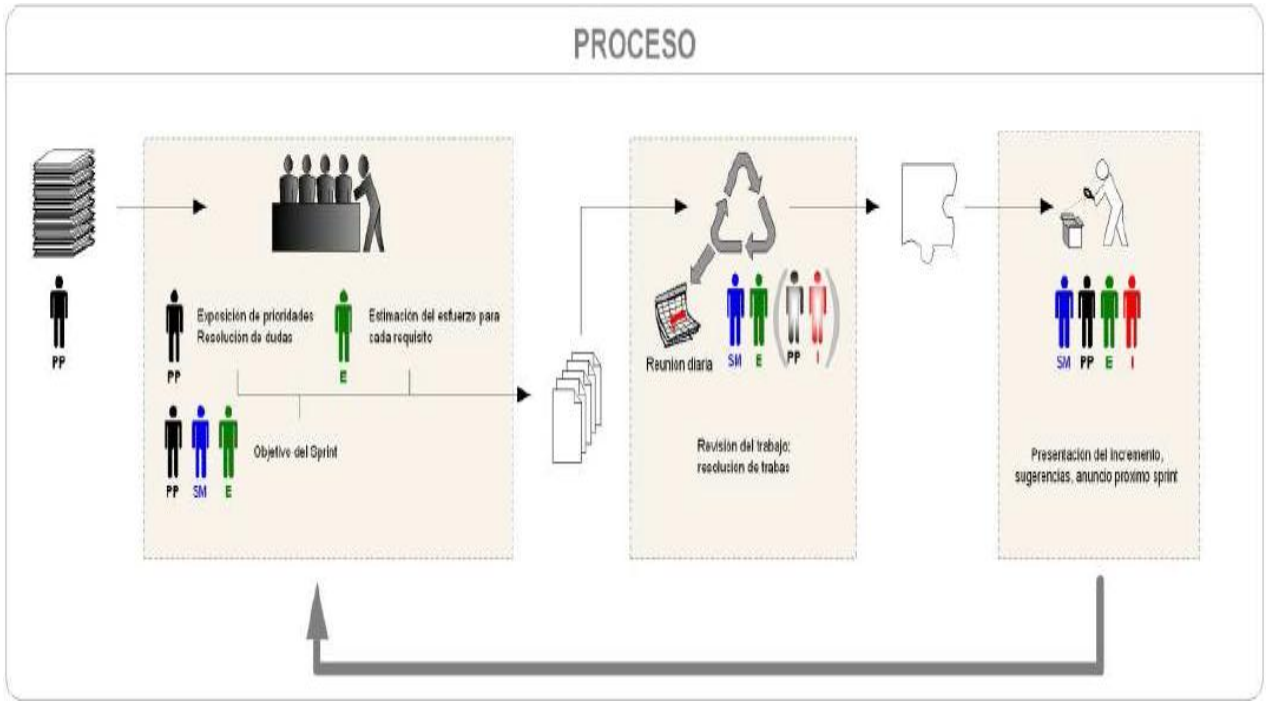

Figura 2.9 Proceso Scrum

Valores

Existen valores que permiten llevar al éxito un producto a través de Scrum.

- ✓ La delegación de atribuciones,
- ✓ Equipos auto-organizados, que permitan tomar decisiones que se consideren pertinentes y oportunas.
- ✓ Estrecha colaboración, la cual es necesaria para que funcione la auto-organización, como un control eficaz y verdadero, cada integrante del equipo debe de colaborar de manera abierta, según su conocimiento y capacidad y no por el puesto que desempeña en una empresa o su rol.

Ventajas de Scrum

- ✓ Fácil de entender
- ✓ Requiere poco esfuerzo para su implementación
- ✓ Transparencia en la planificación y desarrollo de los módulos, al tener una comunicación directa y fluida entre el cliente y el equipo.
- ✓ Monitorización constante del progreso del desarrollo.

Desventajas de Scrum

- ✓ No apto para proyectos de gran escala
- ✓ No es recomendable con un equipo de más de 20 personas
- ✓ No es recomendable para proyectos de una duración mayor a 6 meses.

2.4.4.5 ¿Usar metodologías tradicionales o ágiles?

Las metodologías tradicionales son más efectivas y necesarias en proyectos de software de gran tamaño, proyectos que demandan recursos y tiempo elevados, donde se deben estipular a detalle los procesos involucrados, los roles, actividades, etc.
Si se implementa una metodología tradicional es porque se conoce y sabe exactamente qué es lo que se va a hacer, los requerimientos, las necesidades del cliente, las fechas, los costos y presupuestos de manera definida. En el desarrollo se gestionan riesgos que pueden impactar en el producto, y de ser así se evalúan los impactos que cada modificación supone puedan afectar, se cuenta con una cantidad numerosa de desarrolladores, la cultura de la empresa está basada en disciplina, orden, análisis del riesgo.
Sin embargo en ambientes inestables donde hay constante cambio, donde el tiempo es menor para la entrega de un producto, implementar una metodología tradicional no es recomendable. Puesto que al tener una visión general del objetivo del proyecto, conocer de manera escasa los requerimientos del software a desarrollar y tener a un cliente con dudas

sobre lo que realmente quiere, se vuelve un caos utilizar alguna metodología de gestión tradicional.
En cambio, una metodología ágil permite hacer pequeñas iteraciones o pasos a través de un ciclo de construcción incremental, y conforme existen avances por cada iteración, los clientes pueden ver y tocar los módulos creados ya terminados, así el cliente podrá determinar si es lo que realmente quiere, o de lo contrario cambiar y/o proponer nuevos requisitos y hacer modificaciones en el ciclo de construcción. La documentación del proyecto se va generando conforme es requerida (análisis de requisitos, planificación, diseño, codificación, pruebas), de esta manera se minimiza el riesgo y permite tanto al proyecto como al equipo adaptarse a los cambios rápidamente. De aquí que las metodologías agiles sean de carácter adaptativo.

De lo anterior, no existe una metodología universal para todos los proyectos de tecnologías de la información, sin embargo, se podrá elegir entre una metodología tradicional o ágil a través de 3 puntos determinantes, como son: el cliente, la organización y el proyecto, se debe hacer el diagnostico, por parte del equipo y el líder del proyecto acerca de qué tipo de metodología es pertinente usar en cada desarrollo tomando en cuenta estos 3 puntos determinantes.

2.4.4.6 Características determinantes a considerar antes de elegir una metodología de desarrollo de software.

- Cliente
  - *Prioridad de negocio*: ¿Cuál es el factor más importante para los intereses del cliente?

- Proyecto
  - *Estabilidad de los requisitos*: ¿El cliente definió todos los requisitos?, ¿es posible realizar una lista detallada de los requisitos del cliente? ¿ésta se mantendrá estable durante todo el desarrollo?
  - *Maleabilidad y costo de materia prima:* ¿Qué tan fácil es modificar el producto a raíz de cambio/adición de requisitos?
  - *Criticidad del sistema que se va a construir:* Si el sistema llega a presentar alguna falla ¿qué impacto tiene económicamente para la organización o el cliente?
  - *Costo de prototipo:* Si se elabora una maqueta, dibujos, demos, entre otros, hace ver mejor los requisitos y de esta manera el cliente logra ver la idea ya aterrizada, y de esta manera surgen funcionalidades y posibilidades nuevas que aportan mayor valor al concepto que se tenía inicialmente.
  - *Tamaño del equipo:* Los proyectos grandes requieren equipos de muchos integrantes y en ocasiones físicamente distantes, los proyectos pequeños no necesitan una gran cantidad de personal con 10 personas máximo.

- Organización
    - *Cultura de la organización:* Tomar en cuenta la forma de trabajo de la organización.
    - *Nivel técnico del equipo:* El equipo deberá tener a personas con el mayor conocimiento y experiencia posible.
    - *Estrategia de desarrollo:* Entornos de desarrollo basados en procesos son adecuados para usar metodologías tradicionales y en cambio entornos de desarrollo basados en personas es preferible utilizar modelos agiles.

2.5 Validación y verificación de software

2.5.1 Conceptos

Anomalía: Es un comportamiento en la operación del software que se desvía parcial o totalmente de las expectativas base del producto.

Caso de prueba: Conjunto de entradas de prueba, condiciones de ejecución, y resultados previstos para un objetivo determinado, también es conocido como un escenario de prueba.

Diseño de la prueba: Documentación que especifica

Verificación de software: Busca comprobar que el sistema cumple con los requerimientos especificados tanto en requisitos funcionales como no funcionales, respondiendo a la premisa ¿Estamos construyendo el producto correctamente?

Validación de software: Busca comprobar que el software realice lo que el usuario necesita ajustándose a los requisitos del cliente, respondiendo a la premisa ¿Estamos construyendo el producto correcto?

La verificación de software está enfocada a un aspecto de desarrollo, realizando actividades de casos de pruebas, y revisando directamente el código fuente de la aplicación, mientras que la validación del software se centra en interpretación semántica. El desarrollador al mal interpretar los requisitos del cliente, puede desarrollar otra aplicación que no le sea útil al cliente/usuario, lo cual es lo más grave puesto que todo el trabajo hecho es inútil y se tendría que re hacer, de la manera adecuada.

Objetivo de verificación & validación

Provocar fallas, a raíz de casos de prueba para detectar anomalías, y defectos y corregirlos antes de que el producto sea liberado y entregado al cliente.

Los casos de prueba y pruebas son la mejor manera de detectar errores.

Existen diversas causas que pueden propiciar a que el software tenga errores, tales como:

- Sintaxis: el desarrollador al escribir el código fuente, cambia las letras, le hacen falta llaves, comas, puntos. Etc.
- En la implementación de algoritmos: No inicializar adecuadamente variables, condiciones particulares, comparar tipos de datos no adecuados o no convertirlos adecuadamente(cast)
- Precisión de cálculos: Formulas incorrectas, mal orden en la jerarquía de operaciones a realizar, faltas por no establecer adecuadamente los truncamientos o redondeos de resultados.
- Por documentación: Una mala documentación propicia que el equipo desarrolle módulos inadecuados.

Se debe tomar en cuenta la evaluación de la calidad de los productos, la calidad se debe aplicar de manera adecuada con métodos y herramientas, las revisiones técnicas formales efectivas y una sólida gestión y medida, la cual se confirma durante la etapa de pruebas.

La verificación y validación de software abarcan una amplia variedad de actividades, como revisiones técnicas formales, auditorias de configuración, supervisión y pruebas de rendimiento, simulación, revisión de la documentación, revisión de la base de datos, análisis de algoritmos, pruebas de instalación, pruebas de usabilidad, etc.

La calidad del software es la medida en que el producto cumple con sus especificaciones. El software debe funcionar correctamente bien, y que cumpla con todos los requisitos del cliente.

2.6 Tecnología

2.6.1 Bases de datos

Para tratar datos de manera mecánica y que estos tengan una alta disponibilidad, integridad, confiabilidad, es necesario que estén estructurados de tal forma que pueda ser posible acceder a los datos de manera automática e independiente de los programas que gestionan los datos en cuestión.

Las bases de datos son un conjunto de datos relacionados entre sí, que están estructurados, de forma que puede accederse a estos, independientemente de los sistemas que gestionen esos datos, la independencia permite la posibilidad de modificar la estructura de los datos sin necesidad de modificar el código fuente de los programas que los manipulan.

Sistema de gestión de bases de datos (SGBD)

Organizan y estructuran los datos de tal manera que puedan ser recuperados y manipulados por los usuarios y los programas.

Las estructuras de los datos y las técnicas de acceso proporcionados por un SGBD se denomina modelo de datos. El modelo de datos se subdivide en 2 tipos:

- Lenguaje de Definición de Datos o DDL (Data Definition Language), orientado a describir de una forma abstracta las estructuras de datos y las restricciones de integridad.
- Lenguaje de Manipulación de Datos o DML (Data Manipulation Language), es el orientado a describir las operaciones de manipulación de los datos.

Bases de datos relacionales

Una base de datos relacional es una base que permite establecer conexiones o relaciones entre los datos y mediante las conexiones relacionar datos de varias tablas.

Principales ventajas del uso de bases de datos relacionales son:

- Actúan sobre las tablas en su conjunto, en vez de hacerlo sobre los registros.
- Se pueden realizar consultas complejas que utilizan varias tablas, de una manera simple y fácil de entender.
- Provee herramientas que garantizan evitar la duplicidad de datos.

La organización relacional se caracteriza porque las tablas de la base de datos tienen una estructura de matriz, donde las filas son los registros y las columnas son campos.

Características principales:

- Todos los registros de la tabla deben tener el mismo número de campos, aunque alguno de ellos este vacío, deben ser registros de longitud fija.
- Cada campo debe tener un nombre o etiqueta que hay que definir previa a la utilización.
- La base se forma por varias tablas, una por cada tipo de campo.
- Los registros de una tabla pueden estar en cualquier orden.
- El contenido de un campo está definido por un rango de valores posibles
- Permite la creación de nuevas tablas a partir de las que ya existen relacionando campos de distintas tablas.

2.6.2 SQL Server Standard Edition 2012

Es un sistema de gestión de bases de datos producido por Microsoft, y el cual está basado en el modelo relacional, los lenguajes para las consultas T-SQL, y ANSI SQL, es la competencia directa de otros gestores de bases de datos como los privados como Oracle, DB2, y algunos libres como MySQL, PostgreSQL.

Dentro de las características más importantes destacan:

- Soporta transacciones y procedimientos almacenados.
- Incluye un entorno grafico de control y administración, lo que facilita en gran medida la implementación de DDL y DML.
- Permite trabajar en arquitectura cliente/servidor, de esta manera la información y datos se alojan en el servidor y los clientes (computadoras) acceden solo a la información a través de sitios web o programas.

SQL Server 2012 Standard Edition proporciona la administración básica de bases de datos y base de datos de Business Intelligence para que los departamentos y pequeñas organizaciones ejecuten sus aplicaciones y admite también herramientas de desarrollo comunes, tanto locales como en la nube, que habilitan la administración eficaz de bases de datos con recursos de TI mínimos (véase Figura 2.10 y 2.11).

Figura 2.10 Logotipo de SQL Server 2012

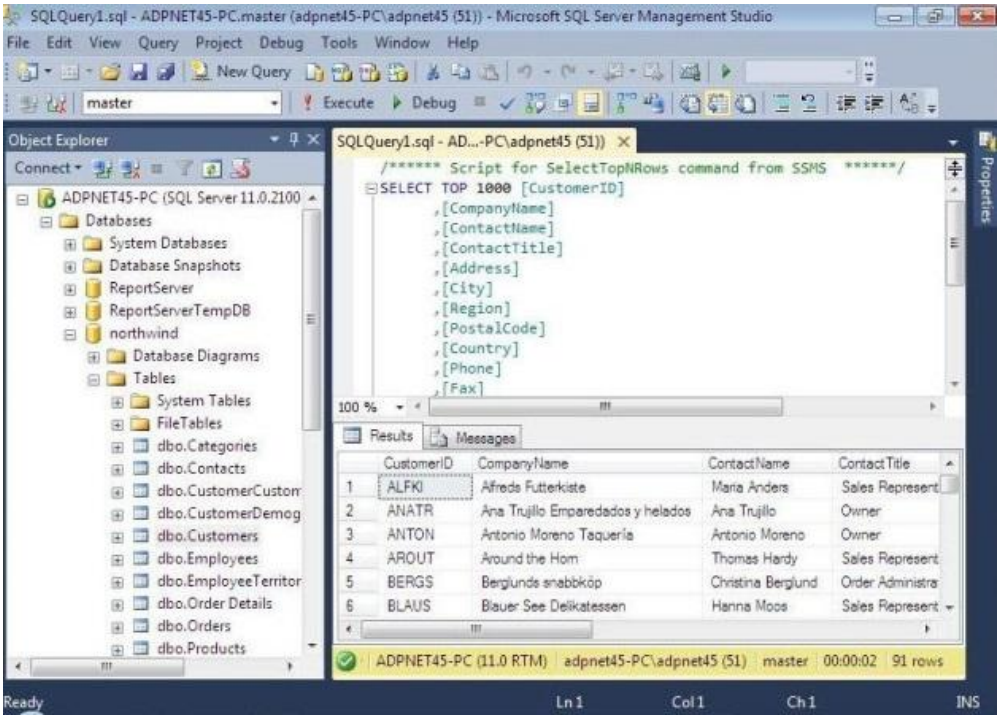

Figura 2.11 Ejemplo del entorno de SQL Server Management

2.6.3 Lenguaje de desarrollo C#

Es un lenguaje de programación que fue diseñado por Microsoft Corporation, en el año 2000, el proyecto fue liderado por Anders Hejlsberg, para compilar diversas aplicaciones que se ejecutan en .NET Framework, se caracteriza por ser simple, eficaz, con soporte al paradigma orientado a objetos.

Es una evolución de los lenguajes C y C++. Utiliza muchas de las características de C++ en las áreas de instrucciones, expresiones y operadores.

C# presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados como lo es la liberación de memoria de forma automática, sin necesidad de realizar punteros como C/C++, presenta un sistema de tipos unificado, esto es que todos los datos derivan de la clase object, por lo que dispondrán de todos los miembros que están definidos en dicha clase.

La versión más actual de C# es la 5.0

Es posible programar con este lenguaje desde diversos Entornos de Desarrollo Integrado, como lo son Visual Studio (todas sus versiones), SharpDevelop, que es una IDE, de software libre con una interfaz gráfica de usuario muy parecida a Visual Studio 2010 dicho IDE está disponible para plataformas Windows, Linux, Mac. Incorpora casi todas las funcionalidades de Visual Studio, como la posibilidad de crear proyectos de Windows Forms, Intellisense, el

completado de código fuente automático, depurador, coloreado de sintaxis para los lenguajes que soporta, etc (véase Figura 2.12).

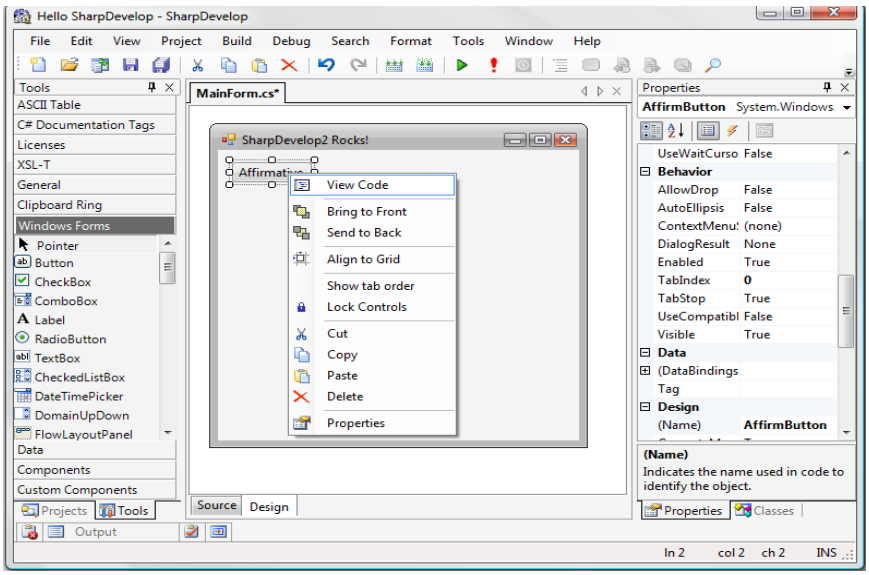

Figura 2.12 Ejemplo del diseñador de formularios en SharpDevelop

Existe el Entorno de Desarrollo Integrado MonoDevelop, del Proyecto Mono, en el cual también se puede programar con el lenguaje C#, en ambientes Windows, Linux, Mac, y es al igual que SharpDevelop, un proyecto de software libre, incluye manejo de clases, ayuda incorporada, completamiento de código, Stetic que es el diseñador de interfaces gráficas de usuario, soporte para proyectos, y un depurador integrado desde la versión 2.2 (véase figura 2.13).

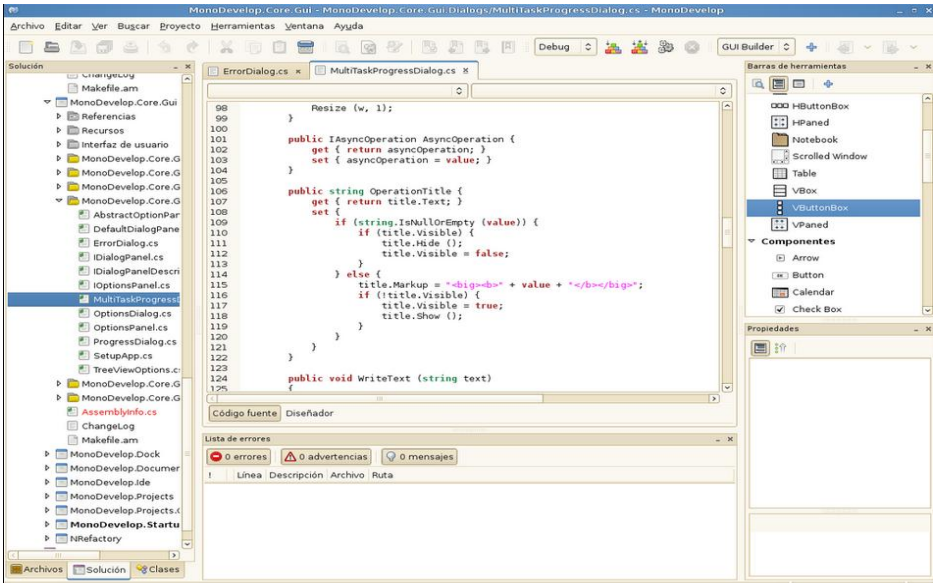

Figura 2.13 Ejemplo de código en C# con MonoDevelop

2.6.4 Framework .NET

Es una plataforma de desarrollo de Microsoft para compilar aplicaciones de Windows, Windows Phone, Windows Server y Microsoft Azure. Está formado por Common Language Runtime (CLR) y la biblioteca de clases de .NET Framework, que incluye clases, interfaces y tipos de valor que son compatibles con una amplia gama de tecnologías.

.NET Framework proporciona un entorno de ejecución administrado, un desarrollo e implementación simplificados, e integración con una gran variedad de lenguajes de programación, incluidos Visual Basic y Visual C#.

La versión más actual es .NET 4.5.2, y permite:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que minimiza los conflictos en el despliegue y versionado de software.
- Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceros desconocidos o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET tiene 2 componentes principales que son:

- Common Languaje Runtime: El cual es un motor en tiempo de ejecución, se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la comunicación remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que promueven su seguridad y solidez.

- Biblioteca de clases: Proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones.

  A través de .NET la administración de la memoria es automática, CLR proporciona estos servicios en nombre de la aplicación, la biblioteca de clases es extensa, lo que permite a los desarrolladores mejor disponibilidad de elementos en vez de programarlos, solo usarlos, incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones como

ASP.NET, para la elaboración de sitios web, ADO.NET, para la comunicación con datos, Windows Communication Foundation para realizar aplicaciones orientadas a servicios, existe interoperabilidad entre lenguajes, con esta característica, las rutinas escritas en un lenguaje están accesibles a otros lenguajes, y los programadores pueden centrarse en crear aplicaciones en su lenguaje o lenguajes preferidos (véase Figura 2.14).

Figura 2.14 Logotipo de .NET 4.5

Microsoft ha anunciado que la plataforma .NET 2015 es ahora código abierto, incluyendo ASP.NET, el compilador para .NET, y librerías, de esta los desarrolladores pueden trabajar con esta plataforma en diversos sistemas operativos, Windows, Mac, Linux/Unix.

2.6.5 Visual Studio

Es un Entorno de Desarrollo Integrado para sistemas operativos Windows, tiene la ventaja de soportar múltiples lenguajes tales como C/C++, C#.NET, Visual Basic.NET, F#, Python, Ruby, ASP.NET, entre otros.

A través de Visual Studio es posible crear aplicaciones locales, sitios web, aplicaciones móviles, videojuegos para plataforma Xbox o conjuntarla con otros productos como Unity, y aplicaciones para Office.

La versión más actual de Visual Studio es del año 2013, y existe en desarrollo la versión 2015 que incluye entre otras cosas un emulador para desarrollar aplicaciones en Android y también para iOS, esto permite a los desarrolladores a construir aplicaciones para una gran diversidad de plataformas, y sistema operativo.

Sin embargo el IDE que utilicé fue Visual Studio 2012, entre las características que destacan:

- ✓ Crear, depurar, optimizar, publicar aplicaciones para la tienda de Windows.
- ✓ Administrar el ciclo de vida de cualquier aplicación a través de Team Foundation Server, sin embargo se requiere una cuenta de Microsoft.
- ✓ Mejores herramientas para la creación de interfaces gráficas de usuario como la incorporación de Blend, para Windows Presentation Foundation.
- ✓ Desarrollo de aplicaciones y soluciones de negocio para Office 2010, 2013.

- ✓ Interfaz gráfica del Entorno de Desarrollo Integrado más amigable.
- ✓ Actualización en el desarrollo para Sharepoint.
- ✓ Administrador de pruebas.
- ✓ Buscador general para el IDE, permite buscar cualquier cosa en toda la solución o proyecto de desarrollo.
- ✓ Edición de código fuente para HTML 5, JavaScript, proporciona IntelliSense (Auto-completado de código y descripción breve de funciones, etc.)
- ✓ Mejoras en la programación asíncrona para Visual Basic y C#.
- ✓ Mayor facilidad de generar código C# a raíz de diagramas de clases UML.
- ✓ ASP.NET con la inclusión de impedir ataques scripting entre sitios.
- ✓ El desarrollo web es más fácil, por la inclusión de nuevos estándares para desarrollar con HTML, CSS.
- ✓ LightSwitch permite conectarse y gestionar orígenes de datos, así como asignar roles a grupos enteros, desde Windows Azure.

Existen diversas versiones de Visual Studio: Professional, Premium, Ultimate (véase Figura 2.15).

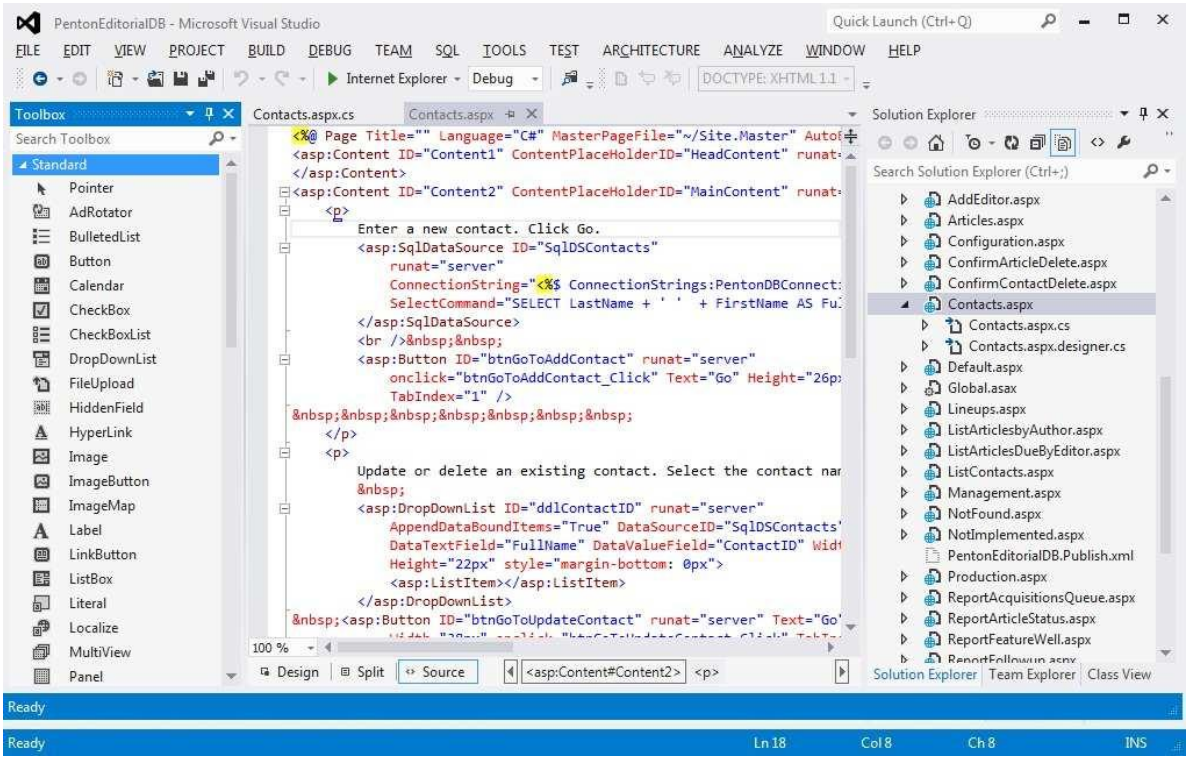

Figura 2.15 Ejemplo de desarrollo con Visual Studio 2012

2.6.6 Desarrollo para Office mediante Visual Studio Tools for Office (VSTO)

Son herramientas para desarrolladores que trabajan con Visual Studio en entornos de negocios, donde los usuarios finales o clientes principalmente pertenecen al sector financiero, ventas, economía, planeación, contabilidad etc. Y que dichos usuarios requieren un mayor alcance de la ofimática para sus actividades diarias, reemplazando las limitadas macros.

El objetivo es diseñar y desarrollar aplicaciones de alta calidad basadas en la interfaz gráfica de usuario que puede ofrecer C#.NET y/o VB.NET mediante Visual Studio, de manera conjunta con Office, para que personalicen y otorguen valor agregado a la experiencia del usuario y mejoren de esta manera su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project. Se agrega un contenido personalizado a un programa que ya conocen los usuarios y están familiarizados con él, por lo que la curva de aprendizaje del sistema creado se reduce y se tiene la manipulación directa de la personalización con Office.

Existen 2 tipos de herramientas, la que es para Microsoft Office 2010 (que también es compatible con Office 2007) y Microsoft Office 2013 también denominada App for Office.

Con VSTO 2010 se utiliza .NET y los lenguajes Visual Basic o C#, mientras que en VSTO 2013 es posible utilizar diversos lenguajes como JavaScript, HTML5, CSS, Python, entre otros, debido a que se basa en integrar la web y lo que se tiene es una página web incrustada dentro de una solución de Office.

Esta moderna plataforma de desarrollo integra en Office 2013 soluciones de la web para crear una experiencia tipo explorador totalmente interactiva. Además, le permite crear soluciones de manera mucho más rápida, ya que usa lenguajes web basados en estándares como JavaScript, HTML5 y JQuery. Por ejemplo, puede combinar datos de ventas de una base de datos de CRM con los mapas del servicio web Bing para crear un informe de ventas más eficaz.

El desarrollo de Aplicaciones para Office 2013, es una nueva forma de que los usuarios interactúen con las personalizaciones de Office, sin tener que instalar o habilitar complementos, los usuarios entran a la Tienda de Apps for Office y descargan la que requieran o a través de un catálogo empresarial interno.

Desafortunadamente entre una versión y otra no existe compatibilidad, puesto que en VSTO 2010 la instalación es casi como la de cualquier programa, y el resultado del desarrollo es un add-in. Mientras que en el desarrollo de aplicaciones para Office 2013 la instalación se hace mediante la tienda de aplicaciones que reside en la nube de Office 2013 y es una solución web. Para poder utilizar el add-in se requiere tener Microsoft Office 2010, y de la misma manera para poder usar la solución web es forzoso contar con Microsoft 2013.

Ventajas de usar VSTO frente a las macros en lenguaje Visual Basic for Applications (VBA)

- ✓ Utiliza código que se almacena por separado con respecto al documento (para las personalizaciones de nivel de documento) o en un ensamblado que carga la aplicación (para los complementos de nivel de aplicación).
- ✓ Proporciona acceso a los modelos de objetos de Office y a las API de .NET Framework como la incorporación de todas las Windows Forms a la suite de Office, mismo que permite un mayor alcance y manipulación de datos tal como un sistema.
- ✓ Diseñado para proporcionar seguridad, facilitar el mantenimiento del código y hacer posible el uso del entorno de desarrollo integrado (IDE) de Visual Studio y todas sus características, funcionalidades y ventajas.
- ✓ Diseñado para el uso en el ámbito empresarial.

La imagen muestra un ejemplo de una personalización con VSTO para Office 2010, con una integración bastante estrecha entre los formularios que nos ofrecen Windows Forms o Windows Presentation Foundation en .NET con Microsoft Power Point. Una nueva pestaña en la cinta de opciones, y el menú en el panel del lado derecho, permiten al usuario tener un valor agregado de alta calidad con varias funcionalidades para automatizar en mejor medida sus actividades cotidianas de una manera más simple (véase Figura 2.16).

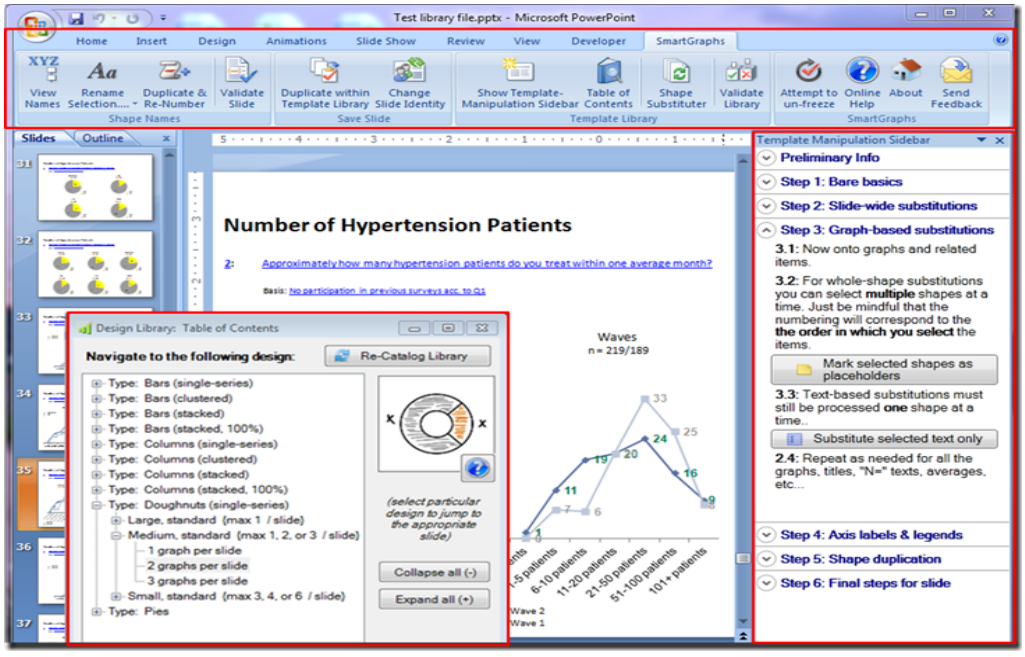

Figura 2.16 Ejemplo de Add-in para Microsoft Power Point 2010

Y la siguiente imagen muestra la funcionalidad web que tiene una App for Office 2013, un reporte con servicios web de mapeo que despliega datos de celdas en ubicaciones a Bing Maps, con un panel para realizar búsquedas (véase Figura 2.17).

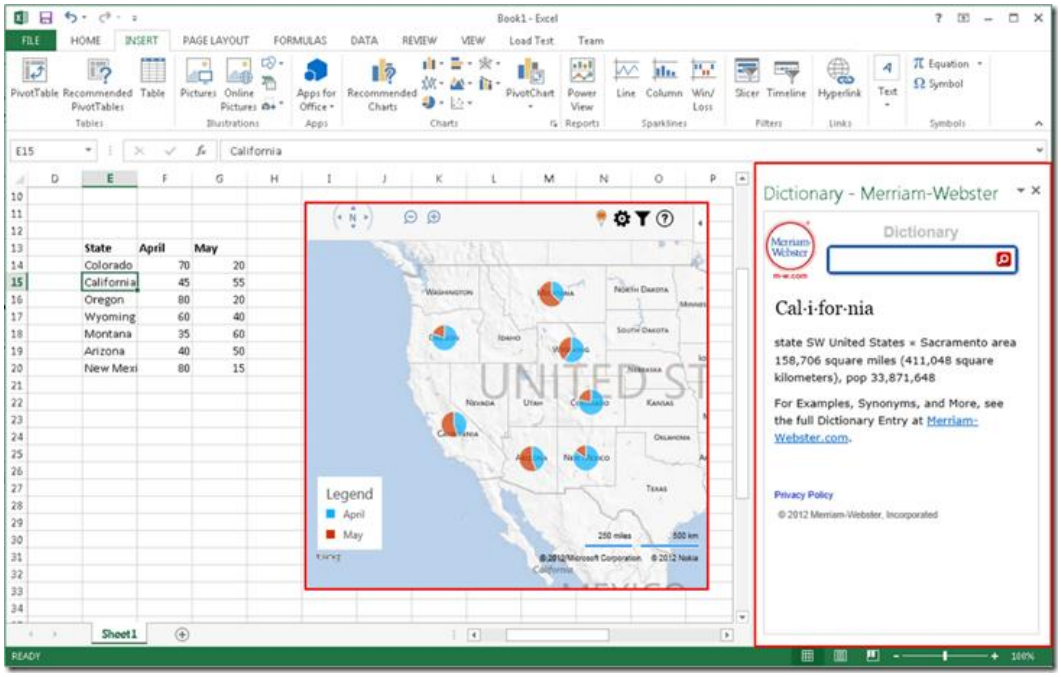

Figura 2.17 Ejemplo de Apps for Office 2013

CAPÍTULO 3:

SISTEMA CONTROL VACACIONAL MEDIANTE LA METODOLOGÍA DE DESARROLLO ÁGIL SCRUM

3.1 Problemática

El área de recursos humanos, no cuenta con un sistema global para administrar todas las vacaciones del personal de todo el Grupo Financiero, por lo que desde años atrás se estipulo como obligación que cada área debe llevar su propio control vacacional.

Anteriormente cuando un empleado solicitaba vacaciones a su supervisor inmediato, llenaba un formato con los días a tomar y lo entregaba en algunos casos solo al supervisor, sin pasar por la administración del área, lo que complicaba demasiado la administración de los formatos y expedientes, se preguntaba al supervisor de cada empleado por los formatos faltantes y al no tener un registro en una base de datos generaba confusión en relación a los días que le quedaban o que había tomado.

En ocasiones el empleado solicitante no avisaba los días que pretendía salir de vacaciones más que a su supervisor inmediato, el resto de compañeros del equipo se veía afectado por los pendientes que pudieran quedar sin una solución.

En ocasiones el empleado o su jefe perdían el formato de vacaciones. Los empleados de nuevo ingreso que no sabían cómo llenar el formato, solicitaban vacaciones aun cuando no cumplían el año de antigüedad, por desconocimiento de la política y reglamento y escribían mal las fechas.

Era difícil llevar un seguimiento de los días disfrutados por parte de empleados que están en el interior de la república, y no se sabía con exactitud si habían salido más o menos días.

Se presentaban problemas cuando 2 o más empleados de una misma línea de negocio querían salir de vacaciones, puesto que debe quedarse un respaldo en oficina por cualquier eventualidad que surja.

Los empleados acumulaban sus vacaciones de varios periodos, lo que hacía complicado saber cuántos días eran de un periodo y cuantos eran de otro.

Los empleados que no salían de vacaciones por exceso de trabajo, perdían el derecho a gozar sus vacaciones al olvidar la fecha de vencimiento de sus periodos.

Existía un descontento general por este tipo de problemáticas.

3.2 Objetivo del proyecto

Este proyecto surge de la necesidad de ayudar a la subdirección de Estrategia y Administración a controlar y gestionar las vacaciones de empleados de una manera más fácil, amigable y simple, y a los empleados mismos para que puedan realizar solicitudes, cancelaciones, etc., de una mejor manera.

3.3 Alcance del proyecto

En relación a los requerimientos del cliente, este proyecto contempló la realización de un add-in para la suite ofimática de Microsoft Excel y una base de datos relacional, que permiten solicitar, controlar, gestionar, administrar las vacaciones de empleados de un área, y mejorar la exactitud de la información, y cuyas características principales son las siguientes:

- Solicitud de vacaciones
  Los empleados a través de un formulario eligen el periodo disponible de vacaciones y las fechas de inicio y termino e imprimen su formato de solicitud vacacional.
- Consulta de vacaciones realizadas
  Los empleados a través de un reporte pueden ver el histórico de solicitudes con los días tomados por empleado.
- Programación de empleados
  Los empleados a través de un formulario pueden consultar las fechas en que salen sus compañeros a efecto de evitar traslapes en las salidas de vacaciones.
- Cancelación de solicitud de vacaciones
  Los empleados pueden solicitar a la asistente, la cancelación de sus días de vacaciones por cualquier situación. Y ellos mismos podrán programar en otras fechas sus días.
- Consulta de estado de días
  El administrador puede tener acceso a un reporte donde tienen los datos de periodos con fechas de vencimiento, para estar monitoreando si ya está próximo un periodo de un empleado a vencer.
- Altas, bajas de empleados.
  A través de un formulario el administrador podrá tener un control mayor de los datos de los empleados.

Está dirigido a:

Usuarios: Empleados del área

Administradores: Asistentes del área quienes llevan el control de expedientes físicos de los empleados del área.

Las tecnologías que serán empleadas son SQL Server 2012 para realizar la base de datos relacional y Visual Studio Ultimate 2012 con Visual Studio Tools for Office 2010.

El resultado entregable es un paquete ejecutable que contiene el add-in para cada usuario.

El proyecto fue hecho haciendo uso de la metodología ágil Scrum.

A través de este proyecto se pretende tener un mejor control, organización y administración de la información.

3.4 Metodología Scrum

Para el desarrollo de este proyecto incorporé el uso de Scrum para un desarrollo ágil, y la adapté en este caso debido a que el equipo de desarrollo lo haría una sola persona, además que tanto el Producto Owner y Scrum Manager serían la misma persona, mi supervisor inmediato.

Tomé la decisión de implementar esta metodología porque el implementar una metodología tradicional sería más complicado, tanto desarrollar la aplicación como elaborar la documentación necesaria siendo el equipo de desarrollo de una única persona, además el área se enfoca más en personas que en procesos operacionales, tal como lo explico en el capítulo 2.

Como ciclo de vida el proyecto toma el modelo en prototipos, puesto que antes de construir la aplicación realicé prototipos dibujados, para ver si las interfaces gráficas que diseñé satisfacían al usuario.

3.4.1 Pila del producto

Los requerimientos del proyecto, los definió mi supervisor inmediato, y son los siguientes:

- Requerimientos para armar la información:
  Buscar y ordenar todos los expedientes de empleados, así como las copias de formatos de vacaciones, posterior a ello validar que la información de los formatos que se tenían fuera correcta con los formatos que tenían los empleados y asistentes.
  Políticas a tomar en cuenta en relación a la solicitud de vacaciones:
  Los empleados en sus primeros 10 años de servicios pueden gozar de 20 días de vacaciones al año, los empleados de 11 a 15 años de servicios pueden gozar hasta 25 días al año de vacaciones, y los empleados a partir de los 16 años de servicio pueden gozar hasta 30 días de vacaciones por año.

  El empleado para poder disfrutar vacaciones debe tener al menos 1 año de antigüedad.

  Los empleados al cumplir su aniversario por su tiempo de servicio, automáticamente activan un nuevo periodo de vacaciones con la carga de días correspondiente al tiempo de servicio. El periodo se toma como el año civil en que se activan los días de vacaciones.

  Ejemplo: Si un empleado ingresa a la empresa el 3 de abril 2014, cumple su primer año el 3 de abril 2015 y el periodo activado es el 2015 con 20 días de vacaciones.

Desde que un empleado activa un nuevo periodo de vacaciones, tiene 18 meses para disfrutarlo, de lo contrario se vence el periodo y los días que tenga asignados, sin ser acumulable o recompensado monetariamente.

- Requerimientos del desarrollo:
  Que el software fuera muy sencillo de utilizar dándome la libertad de diseñar las interfaces graficas de usuario y que estuviera integrado dentro de la hoja de cálculo de Microsoft Office Excel 2010 (add-in) y la información en una base de datos dentro del gestor de bases SQL Server.
  Es fundamental para un área de enfoque financiero, trabajar a diario con la herramienta de la hoja de cálculo, por las diversas actividades de reportes y consolidados, por lo que me hicieron hincapié en que el software debía de ser un add-in.

- Requerimientos de los módulos de la aplicación:

  - **Inicio de sesión**
    Que el sistema tuviera un inicio de sesión con roles para empleados y administradores (asistentes).

  - **Solicitud de vacaciones**
    El usuario ingresa el número de empleado, y puede ver en una ventana sus datos principales como Nombre, puesto, área, ubicación, el territorio al que pertenece, población, la extensión del teléfono cisco. Y los datos de antigüedad y correspondencia de día de vacaciones como son, fecha de ingreso a la empresa, tiempo de servicio, los días correspondientes de vacaciones en relación a la antigüedad, el periodo disponible, la efectividad de los días, que es el tiempo que tiene para poder utilizarlos, el estado del periodo vacacional.
    En la misma ventana incorporar dos calendarios seleccionables para las fechas de inicio y término de las vacaciones de un empleado, los días hábiles a disfrutar, los días restantes del periodo al que tiene derecho, y la fecha en que debe regresar a labores.
    Un botón para realizar la solicitud, que imprima el formato, y envié la notificación por correo electrónico al supervisor inmediato del empleado y a la subdirección de estrategia y administración.
    El formato expedido debe contar con un folio para realizar seguimiento.

  - **Historial de vacaciones**
    Un reporte que incluya los datos principales del empleado como nombre, número de empleado, área, fecha de ingreso al banco y las fechas de vacaciones que ha tomado por año, así como el número de días hábiles

disfrutados, y que dicha información sea mostrada en la hoja de cálculo de Microsoft Office Excel.

- **Programación de empleados**
  Un módulo en una ventana que permita que cualquier empleado pueda ver un calendario con las fechas de vacaciones que otro empleado tomará.
  El modulo debe permitir hacer consultas de muchos empleados o por nombre, y por rangos de fechas, así como el folio. De esta manera se evitan traslapes en la salidas de personas de una misma línea de negocio y/o los empleados toman en cuenta qué compañeros salen de vacaciones para trabajo entre varios colaboradores.

- **Cancelación de solicitud de vacaciones**
  Un módulo que permita al personal con rol de administrador, cancelar las vacaciones de un empleado cuando este necesite hacerlo por cualquier causa, utilizando para ello el folio de vacaciones del formato expedido y el empleado pueda volver a solicitas sus días de vacaciones. De esta manera se evita que el empleado, sobre todo los del interior de la república, haga mal uso del programa, el proceso debe pasar forzosamente por las asistentes y debe estar enterado el supervisor del empleado.

- **Consulta de días y periodos rápida**
  Un módulo con un reporte en la misma hoja de cálculo para la dirección, que muestre los días disponibles de cada periodo un empleado, así como si son vigentes o si ya prescribieron.

- **Altas, cambios, bajas**
  Un módulo donde se pueda hacer por parte del personal asistente, las altas o bajas de personal de toda el área, así como los cambios de personas que llegan de otras áreas de la empresa, y que ya cuentan con salidas de vacaciones, éstas se puedan integrar y considerar en la contabilización de días disponibles de periodos que tengan activos.
  Así como dar de alta aquellos días festivos a omitir en el calendario, año por año.

- **Menú de acceso a los módulos**
  La aplicación debe tener un menú en forma de panel incrustado desde el momento que se inicial el software de Microsoft Excel, para poder navegar por los diferentes módulos del sistema.

Para describir mejor los requerimientos elaboré 3 diagramas UML, el diagrama de caso de uso para un empleado y un administrador son los que se muestran en las siguientes imágenes (Figuras 3.1-3.3):

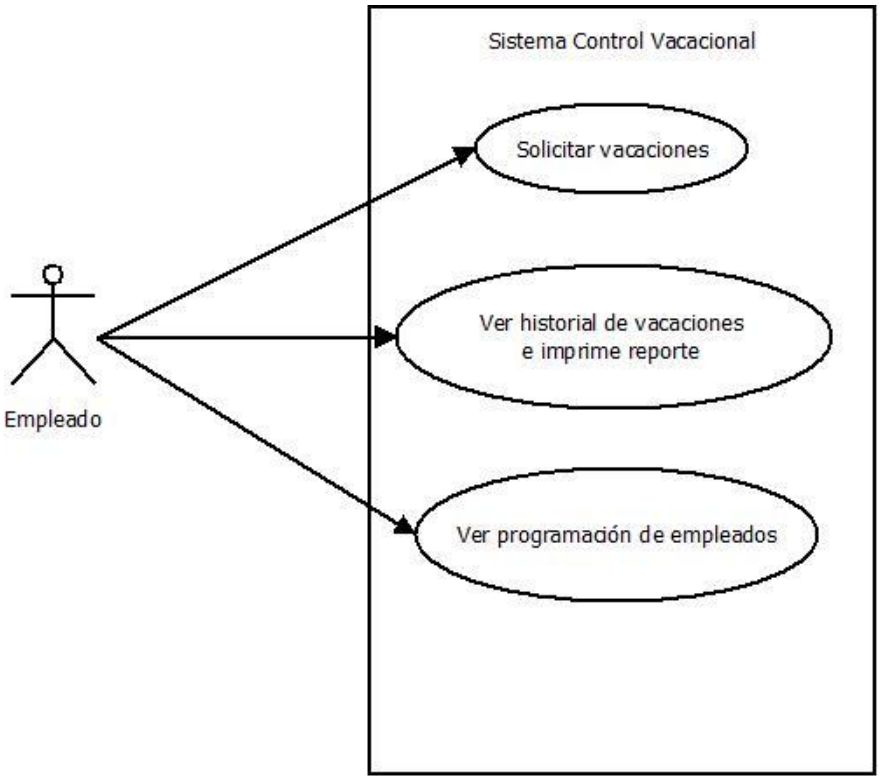

Figura 3.1 Diagrama UML, casos de uso para un empleado

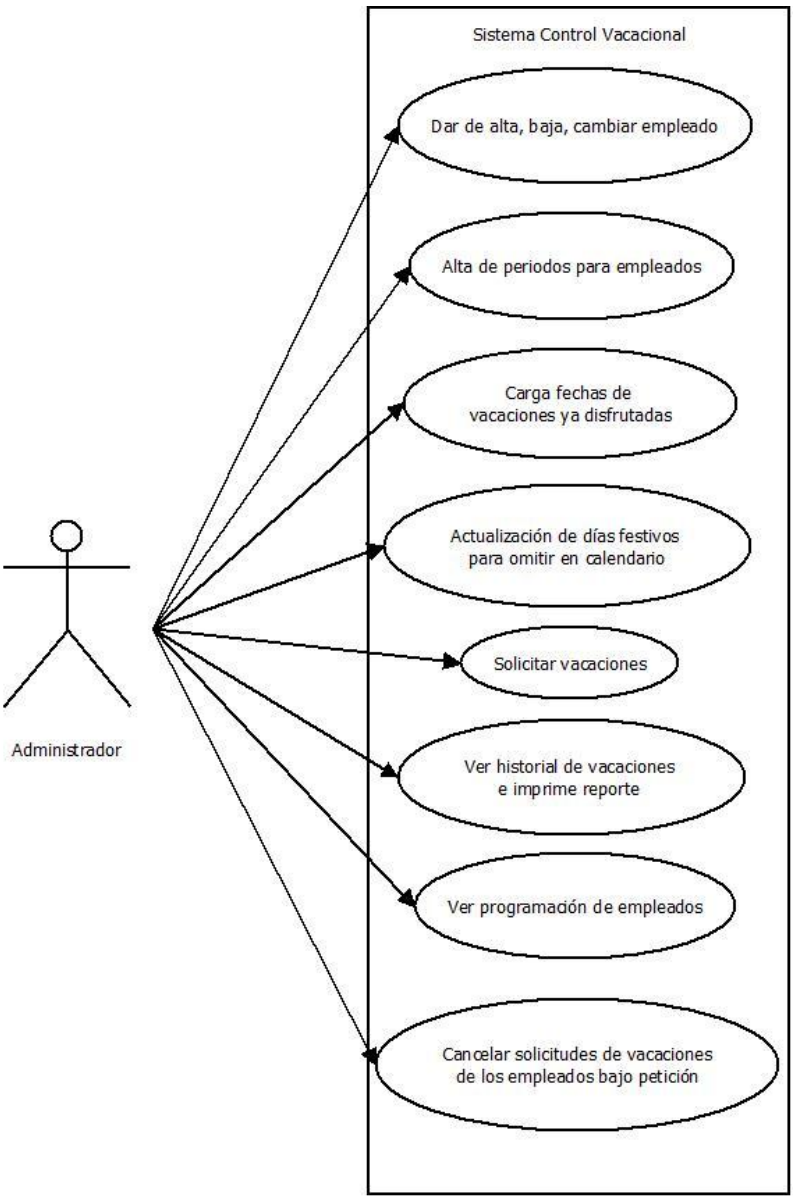

Figura 3.2 Diagrama UML casos de uso para administrador

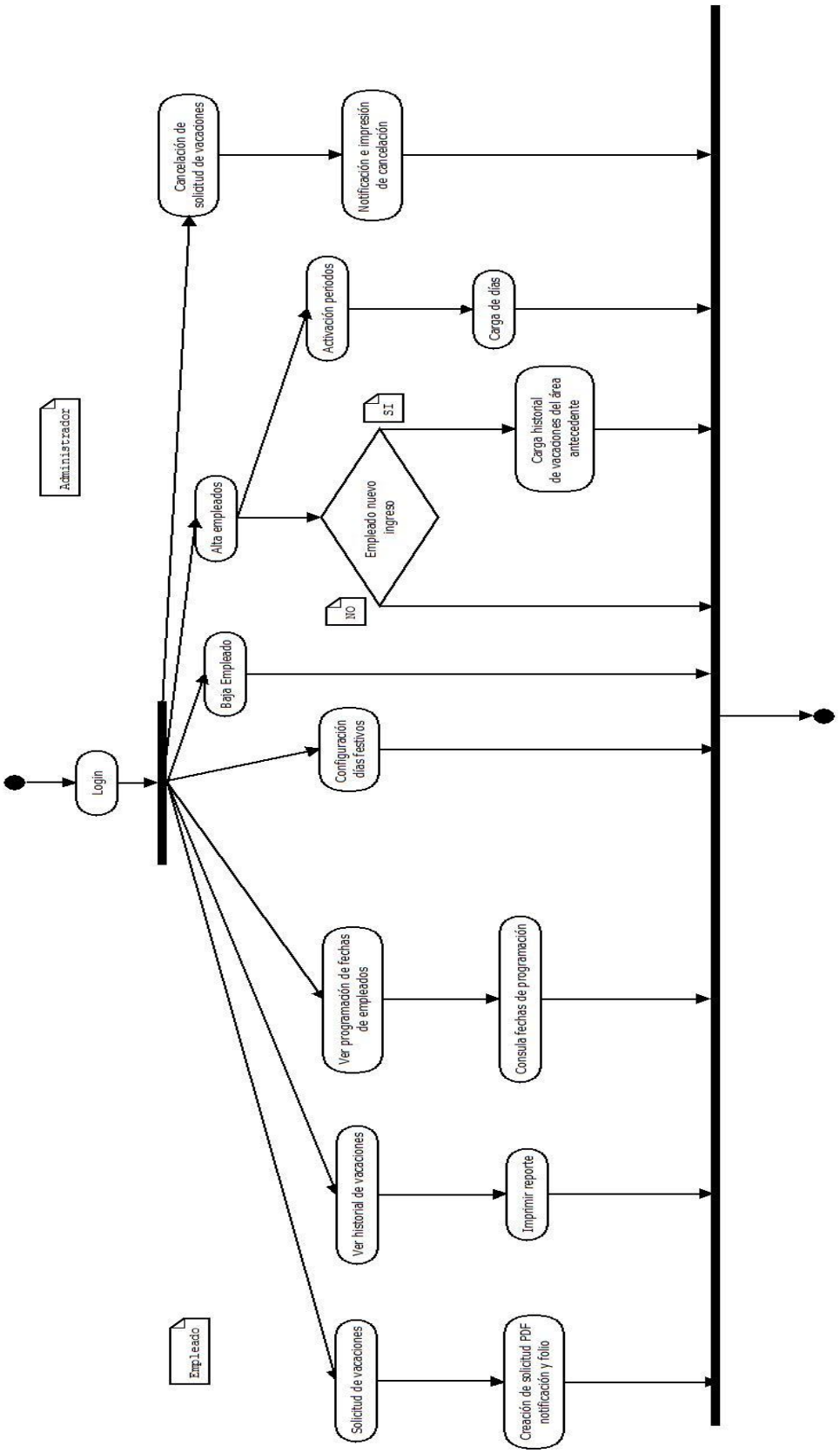

Figura 3.3 Diagrama UML de actividades

3.4.2 Planificación del Sprint

A efecto de realizar la pila del sprint, estudié los requerimientos para darle prioridad a cada uno de ellos y así tener la pila del Sprint.

Objetivo del Sprint

El objetivo que establecí fue entregar un sistema funcional, incorporando todos los requerimientos que el Product Owner definió, en un solo Sprint.

Estimación de esfuerzo para cada requerimiento y asignación de actividades por prioridad

En la siguiente tabla, muestro el estudio que realicé para establecer un orden por prioridad para cada actividad (véase Tabla 1.1).

# Req	Requerimiento	Estimación de esfuerzo (hrs)	Prioridad
1	Armar la información	20	1
2	Base de datos	5	2
3	Inicio de sesión y roles	10	4
4	Solicitud de vacaciones	15	6
5	Historial de vacaciones	6	7
6	Programación de empleados	8	8
7	Cancelación de solicitud de vacaciones	6	9
8	Consulta de días y periodos	4	10
9	Altas, cambios, bajas	15	5
10	Menú en panel de acceso	4	3
11	Documentación	6	11
12	Pruebas unitarias	8	12
13	Pruebas de integración	20	13

Tabla 1.1 Priorización de actividades

3.4.3 Pila del Sprint

Una vez que determiné el orden de prioridad y la estimación de esfuerzos por cada actividad la pila del sprint estaba conformada de la siguiente manera:

1. Armar información tomando en cuenta expedientes y formatos.
2. Realizar Base de datos
3. Menú de acceso y navegación
4. Inicio de Sesión y roles
5. Altas, cambios y administración de usuarios.
6. Solicitud de vacaciones
7. Historial de vacaciones
8. Programación de empleados
9. Cancelación de solicitud de vacaciones

10. Consulta de días y periodos
11. Documentación
12. Pruebas unitarias
13. Pruebas integración

3.4.4 Desarrollo

El principal problema que se me presentó fue que no existía información actualizada referente a los datos de los empleados, como fecha de ingreso o formatos de solicitud de vacaciones anteriores, así como formatos con datos inconsistentes y erróneos.

Antes de realizar la base de datos comencé a ordenar los expedientes de los empleados, los cuales estaban revueltos, con documentos traspapelados en otros expedientes de otros empleados, solicité a cada empleado me proporcionara sus formatos vacacionales para llevar un cotejo de los días de vacaciones y periodos solicitados.

Ya que terminé de elaborar el ordenamiento y clasificación de documentos y expedientes, validé la información con la asistente del área.

Base de datos

La base de datos la realicé en SQL Server 2012, (véase Figura 3.4), cuenta con 4 tablas que son las siguientes:

- Empleados: Es la tabla donde se almacena toda la información del personal.
  El registro del empleado, nombre, área, puesto, dirección, fecha de ingreso, tiempo de servicio, extensión CISCO, correo electrónico, territorio, población, contraseña.
- Vacaciones: Es la tabla donde se almacena toda la información correspondiente de las vacaciones del empleado solicitante, el folio de la solicitud de vacaciones, la fecha de solicitud de vacaciones, la fecha de inicio de vacaciones, la fecha de término de vacaciones, los días hábiles a disfrutar, el periodo al que corresponden los días, la fecha de regreso a labores.
- Periodos: Es la tabla donde se almacenan los periodos a los que tienen derecho los usuarios, los días que tiene derecho cada persona en relación a su antigüedad, la fecha en que cumple un periodo y la vigencia del mismo, y el estado de cada periodo.
- Roles: Es la tabla que contiene los roles de usuario.

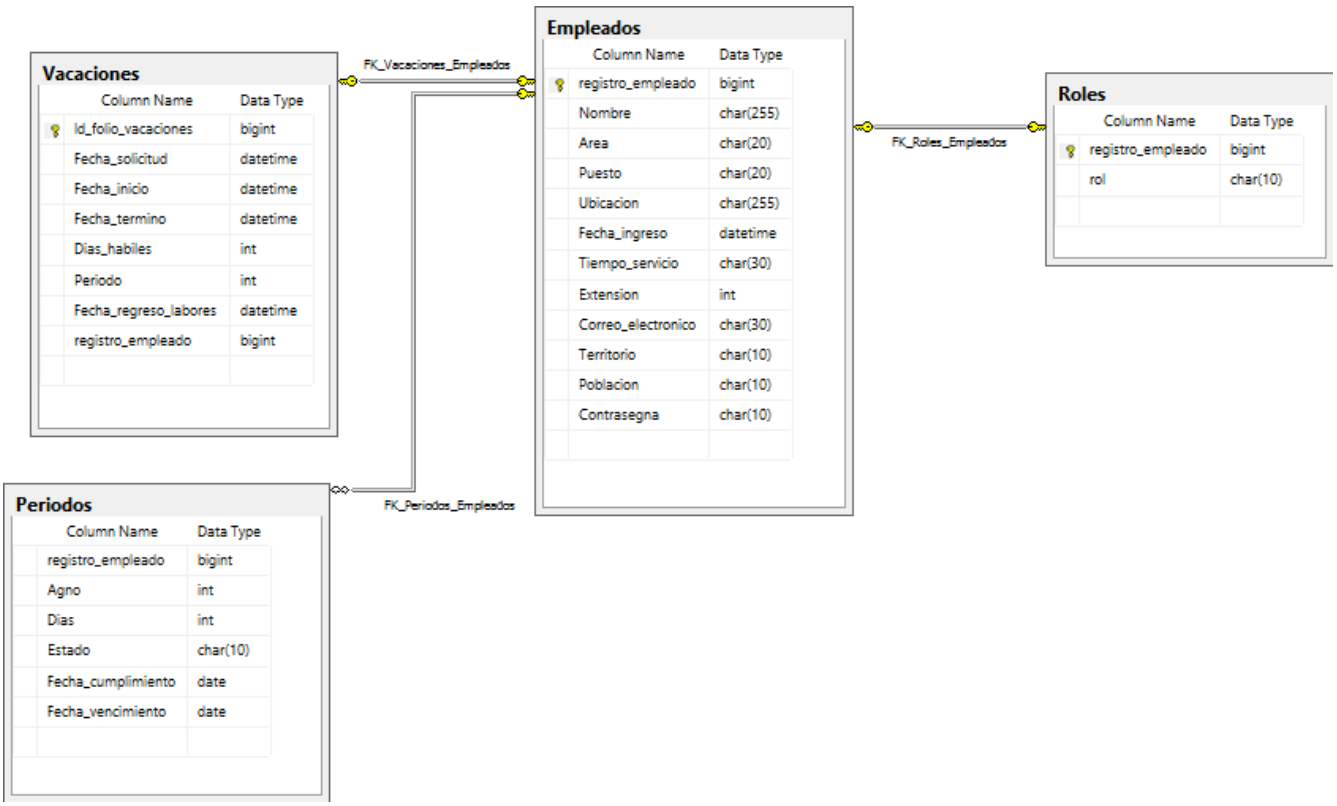

Figura 3.4 Diagrama de la base de datos

Una vez que terminé con la base de datos, comencé a programar cada uno de los módulos utilizando el Entorno de Desarrollo Integrado Visual Studio Ultimate 2012, el lenguaje de programación C#, el Framework .NET, y Visual Studio Tools for Office para Office 2010.

Desarrollo del Add – in

Login de acceso

Los usuarios pueden entrar al sistema desde la banda de opciones de Microsoft Office Excel en la opción de complementos y pulsar el botón que activa el sistema control vacacional.

Al inicio los usuarios deben escribir su número de empleado y su fecha de ingreso a la empresa, que es su contraseña.

El flujo que sigue el sistema para validar a un usuario es el siguiente (véase Figuras 3.5 y 3.6):

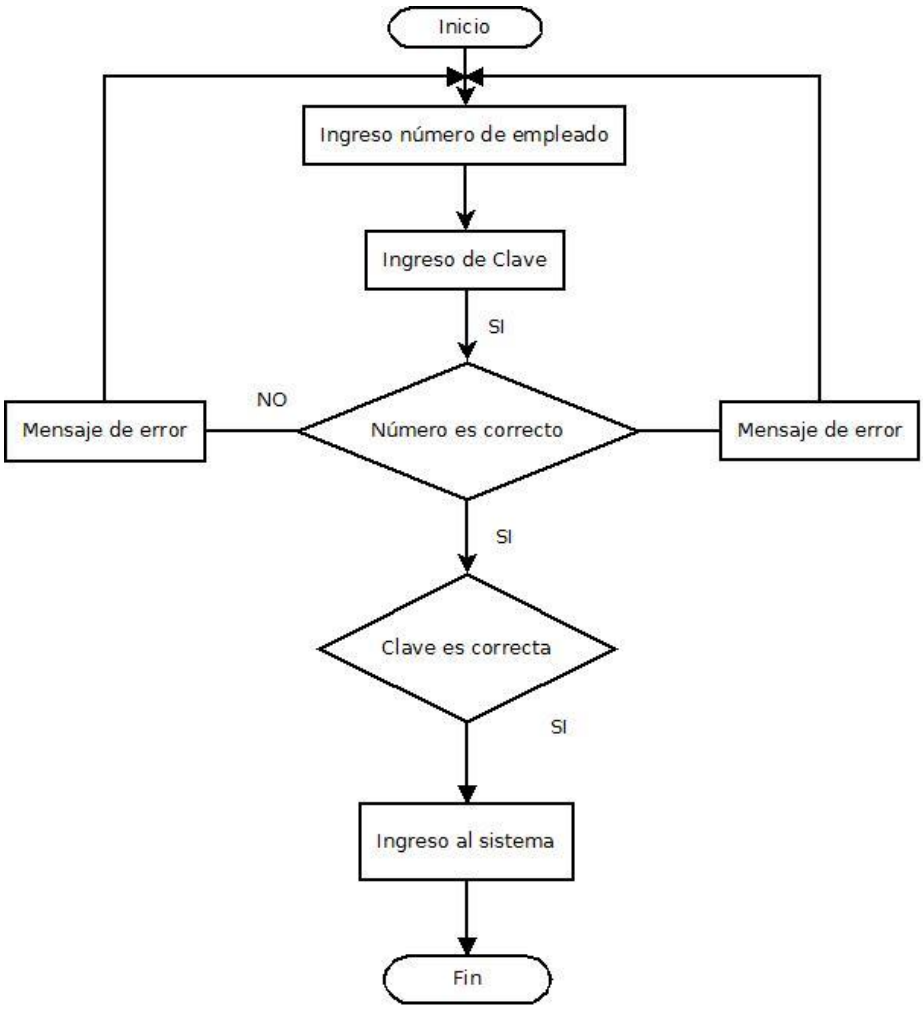

Figura 3.5 Diagrama de flujo de Login de acceso

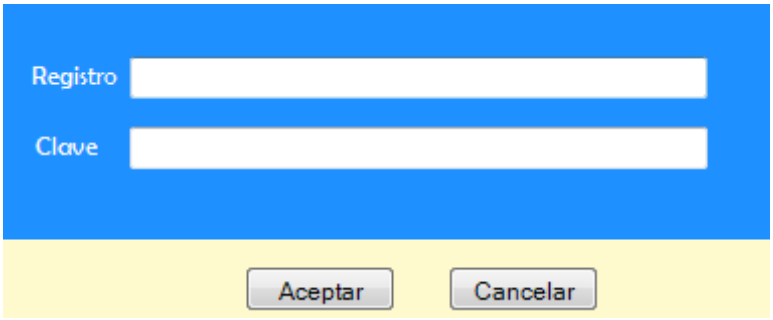

Figura 3.6 Ventana de acceso

Solicitud de vacaciones

El diagrama para la solicitud de vacaciones es el siguiente (vease Figuras 3.7 y 3.8):

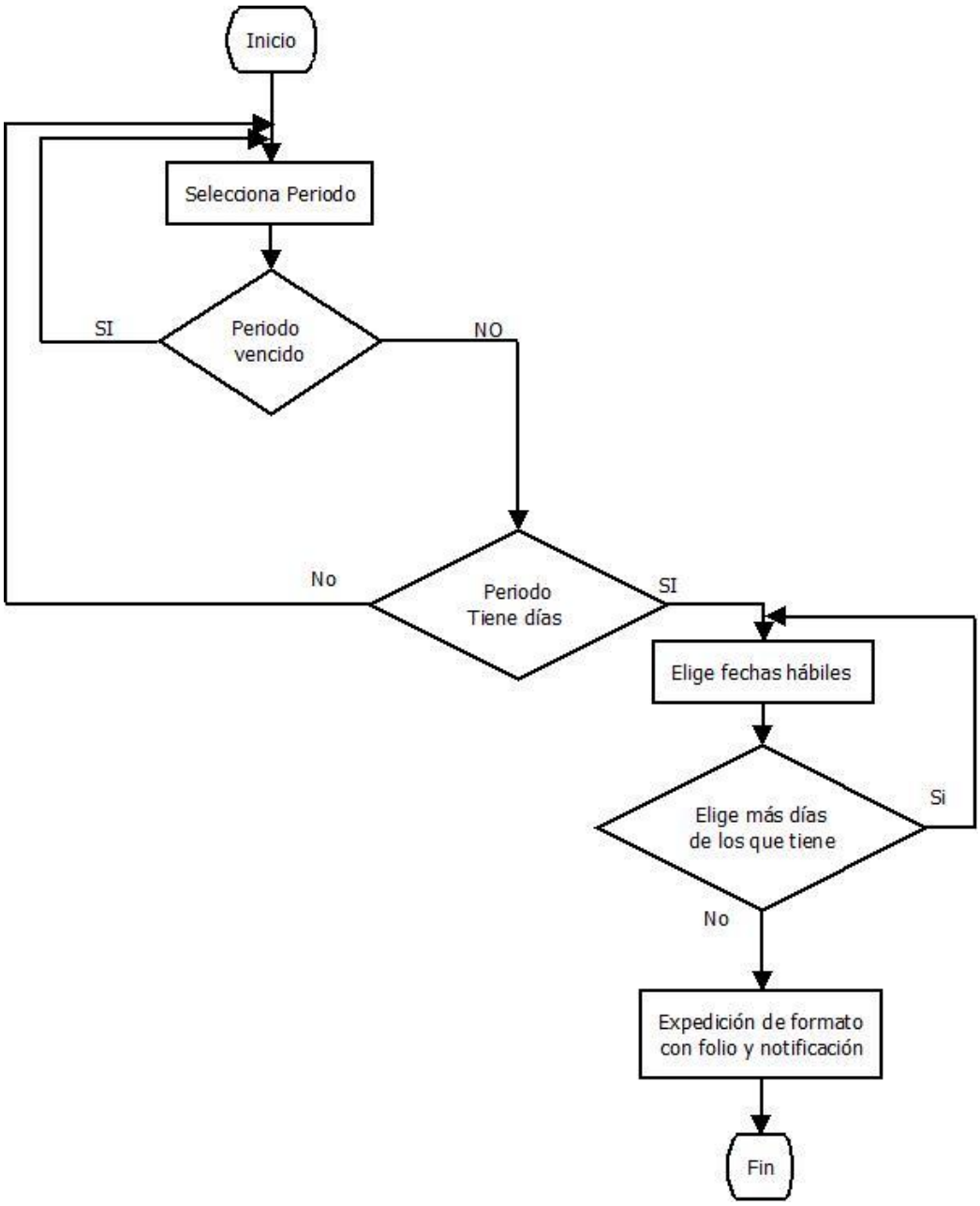

Figura 3.7 Diagrama de flujo de la solicitud de vacaciones

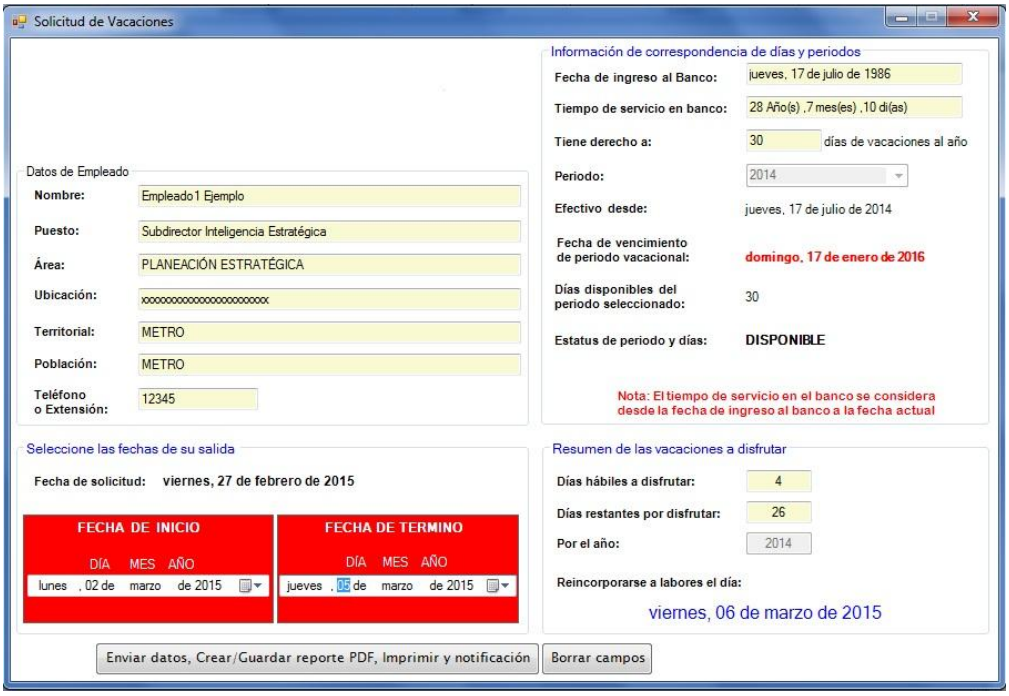

Figura 3.8 Formulario de Solicitud de vacaciones

Consulta de historial de vacaciones disfrutadas

(véase Figuras 3.9 y 3.10)

Figura 3.9 Diagrama de flujo de consulta de historial

Datos del Empleado

	1134000
Registro	
Nombre	Empleado1 Ejemplo
Fecha de ingreso	jueves, 17 de julio de 1986
Área	PLANEACIÓN ESTRATÉGICA

Resumen de estado de periodos

Dias_pendientes_2012	Status_periodo_2012	Fecha_cumplimiento_per_2012	Fecha_vencimiento_2012	Dias_pendientes_2013	Status_periodo_2013	Fecha_cumplimiento_per_2013
0	DISPONIBLE	17/07/2012	17/01/2014	0	Terminado	17/07/20

Detalle del historico de vacaciones disfrutadas

Numero_Registro	Periodo	Fecha_solicitud_vacaciones	Fecha_inicio_vacaciones	Fecha_termino_vacaciones	Dias_habiles_toma	Fecha_regreso_labores
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20
1134000	2012	03/12/2012	04/12/2012	06/12/2012	3	09/12/20

Figura 3.10 Ejemplo de reporte de historial

Ver programación de empleados

(Véase Figuras 3.11 y 3.12)

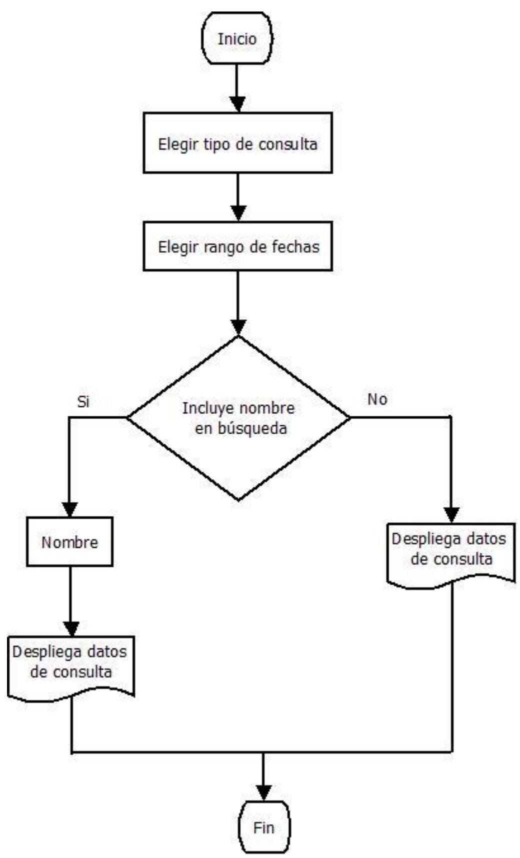

Figura 3.11 Diagrama de flujo de programación de empleados

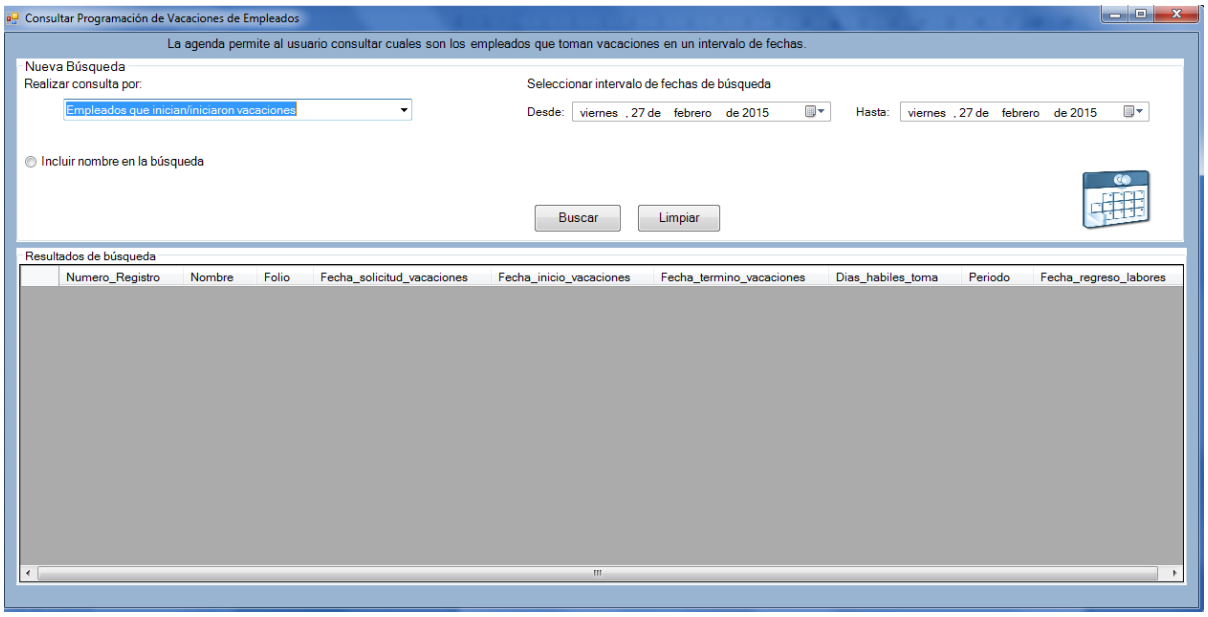

Figura 3.12 Formulario de consulta de la programación de fechas de empleados

Cancelación de solicitud de vacaciones

(Véase Figura 3.13)

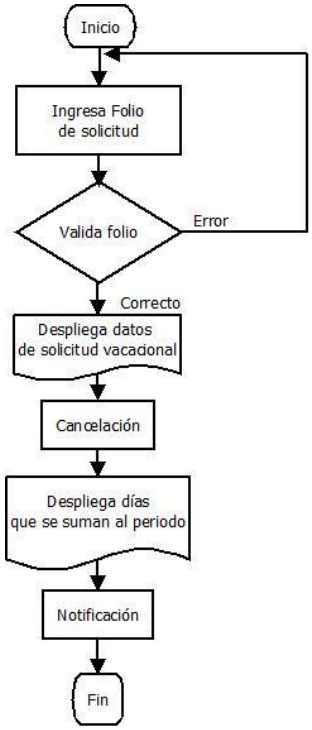

Figura 3.13 Diagrama de flujo de cancelación de solicitud

El rol de administrador es el único rol que puede ver esta opción, ningún otro usuario puede verla ni utilizarla.

(Véase Figuras 3.14 y 3.15)

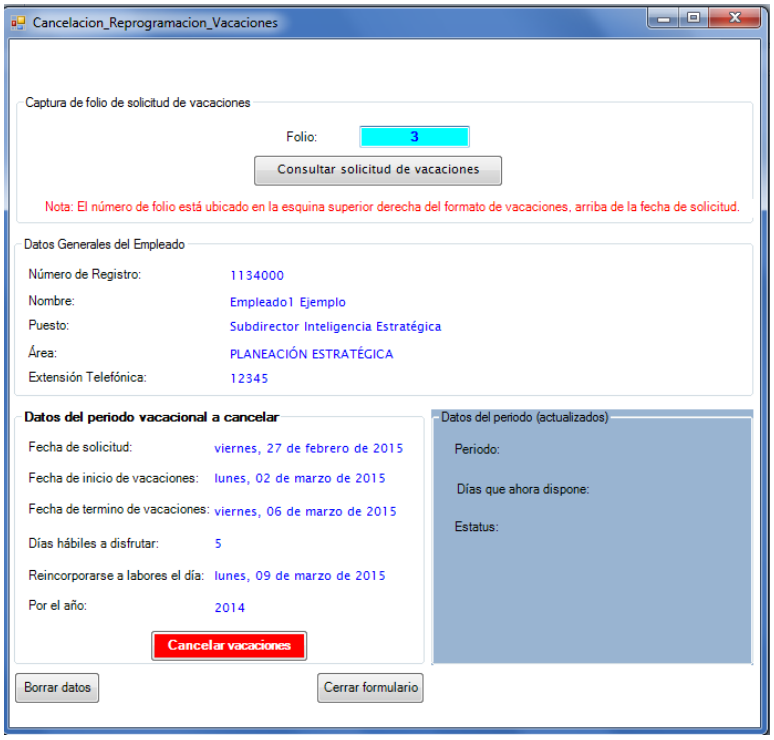

Figura 3.14 Formulario de cancelación de solicitud vacacional

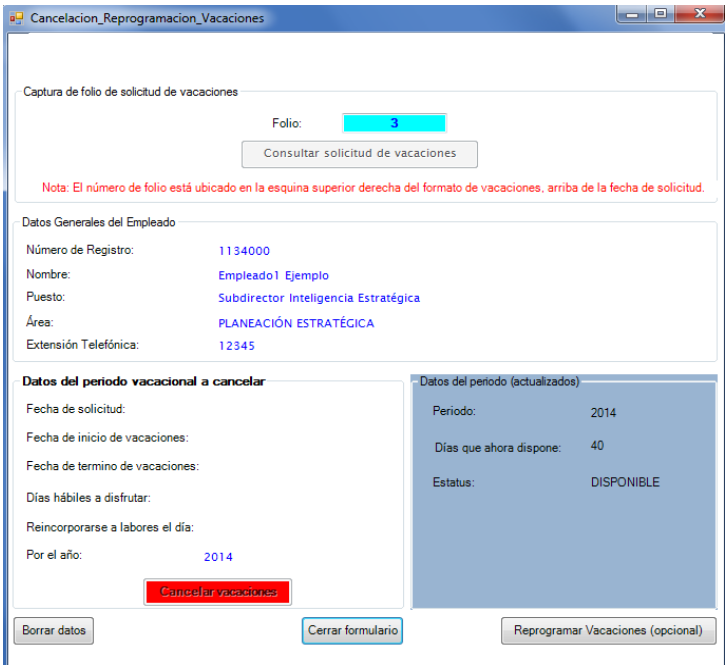

Figura 3.15 Formulario de cancelación de solicitud vacacional

Generación del reporte consulta de días y periodos

Este módulo es también uso solo del administrador, y dicho reporte se entrega a la Dirección del Área cada que sea solicitado (véase Figuras 3.16 y 3.17).

Figura 3.16 Diagrama de flujo de consulta de días y periodos del área

	A	B	C	D	E	F
1		Generar Reporte				
2	Muestra información detallada de los días de vacaciones correspondientes que tiene el personal de GTB.					
3	Especifica los días pendientes, disponibles, vencidos y su estado actual, así mismo muestra el periodo correspondiente.					
4						
5	Numero_Registro	Nombre	Dias_pendientes_2012	Dias_pendientes_2013	Dias_vacaciones_2014	Status_periodo_2012
6	12345	Empleado1	0	0	17	TERMINADO
7	123345	Empleado2	0	0	0	NO DISPONIBLE
8	1234544	Empleado3	0	24	30	NO DISPONIBLE
9	272342	Empleado4	0	0	0	NO DISPONIBLE
10	12345345	Empleado5	0	0	6	NO DISPONIBLE
11	1239895	Empleado6	0	0	0	NO DISPONIBLE
12	12345009	Empleado7	0	0	40	DISPONIBLE
13	12345994	Empleado8	0	0	0	DISPONIBLE
14	12345009	Empleado9	0	25	30	VENCIDO
15	12345234	Empleado10	0	0	10	NO DISPONIBLE
16						

Figura 3.17 Reporte

Altas, Bajas, carga de días, carga de historial y configuración de días festivos.

(Véase Figuras 3.18-3.23)

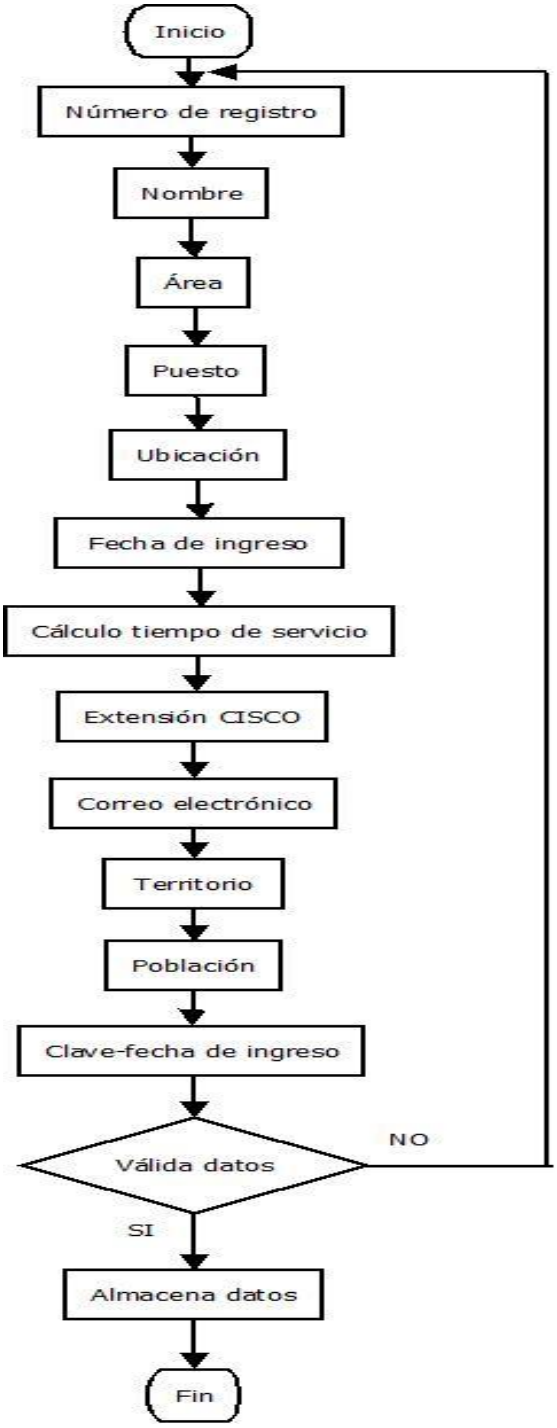

Figura 3.18 Diagrama de flujo alta de empleados

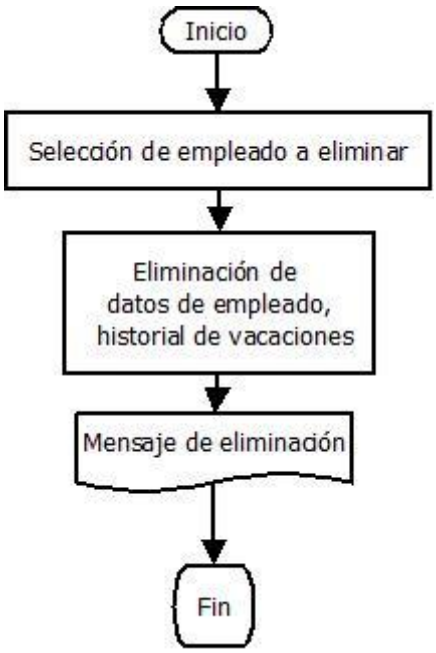

Figura 3.19 Diagrama de flujo baja de empleado y su historial

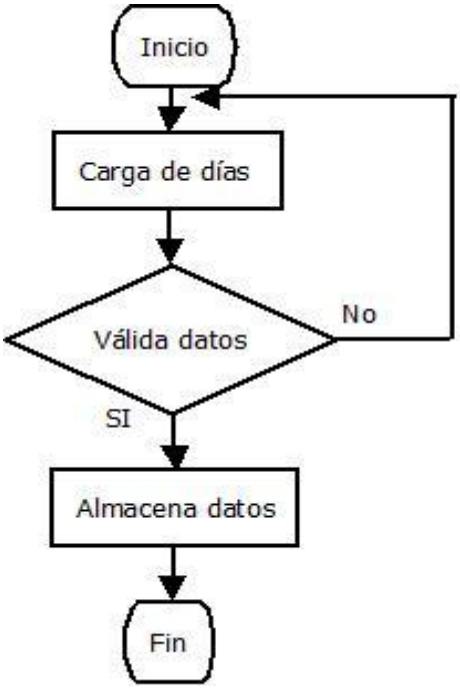

Figura 3.20 Carga de días

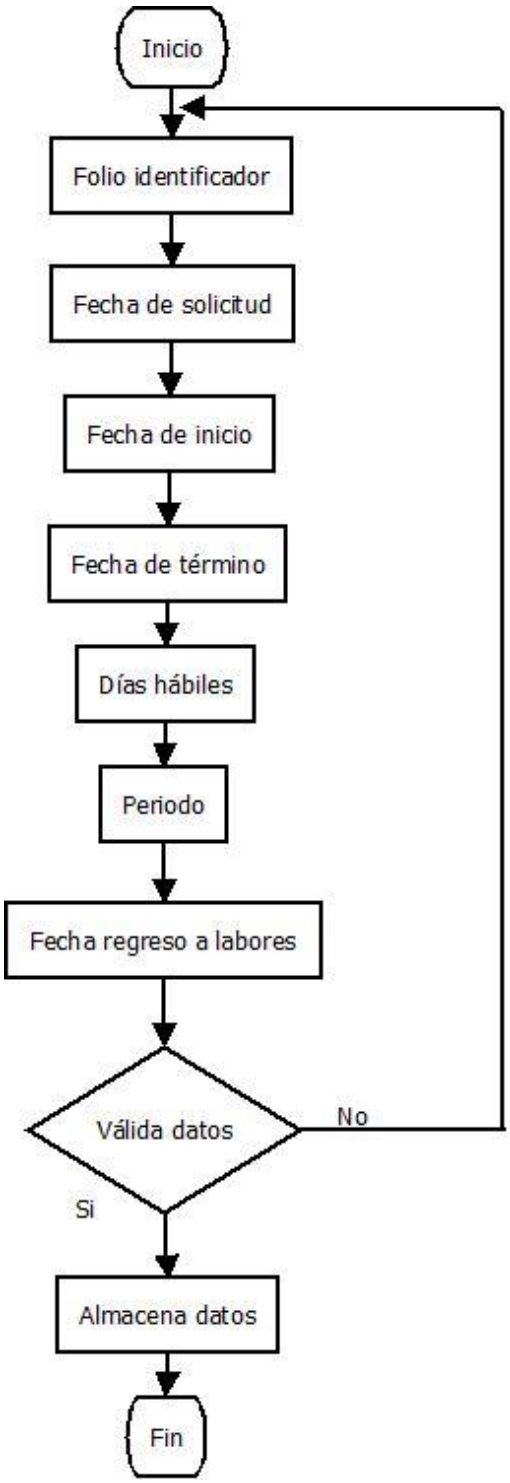

Figura 3.21 Diagrama de flujo de carga historial vacacional para empleados que se cambian de área

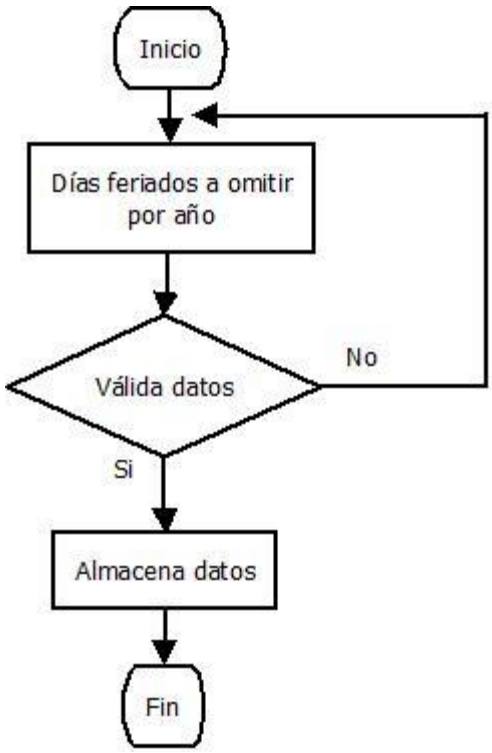

Figura 3.22 Diagrama de flujo de la configuración para omitir días feriados en el cálculo de días a elegir por solicitud vacacional

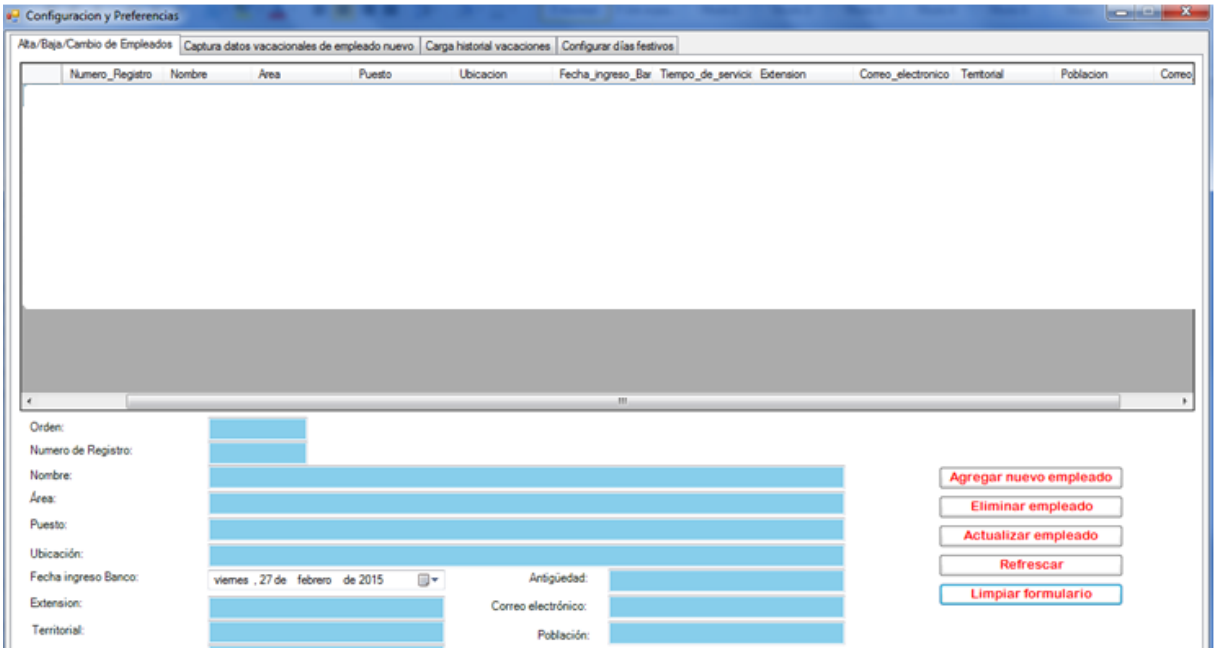

Figura 3.23 Formulario de altas, bajas, cambios, captura de días, carga historial y configuración de días feriados.

El formato de emisión de la solicitud vacacional es impreso 3 veces, un formato para el empleado, otro para el jefe inmediato y otro para el expediente (véase Figura 3.24).

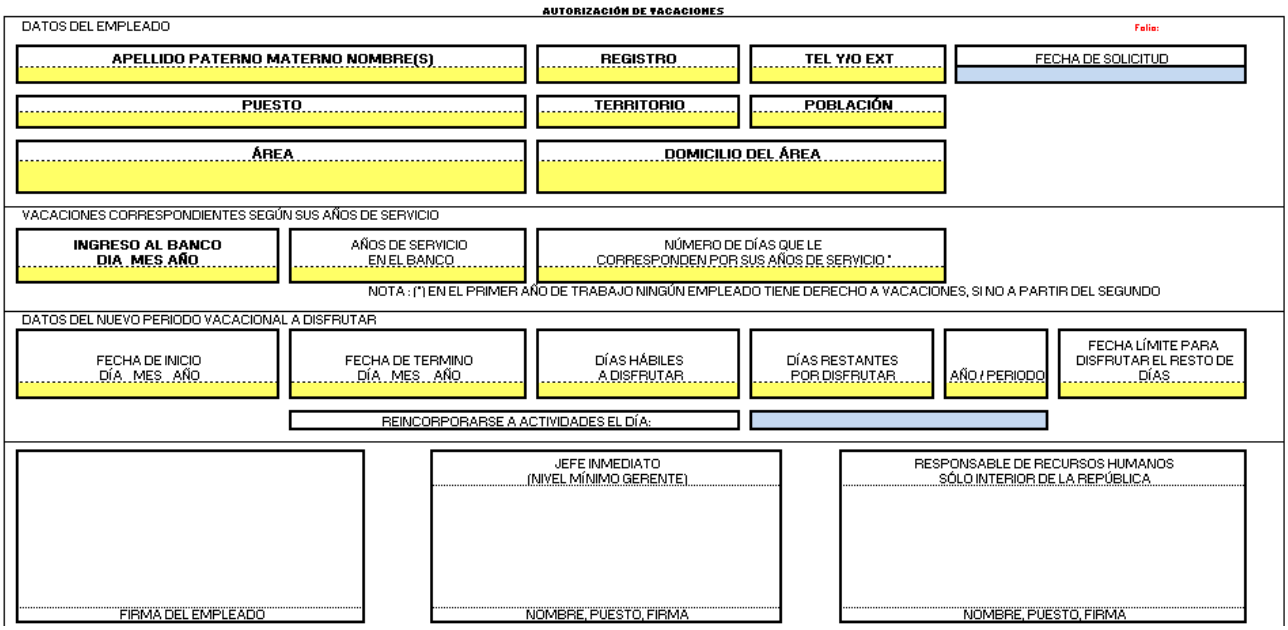

Figura 3.24 Formato de solicitud de vacaciones

Pantalla inicial

(Véase Figura 3.25)

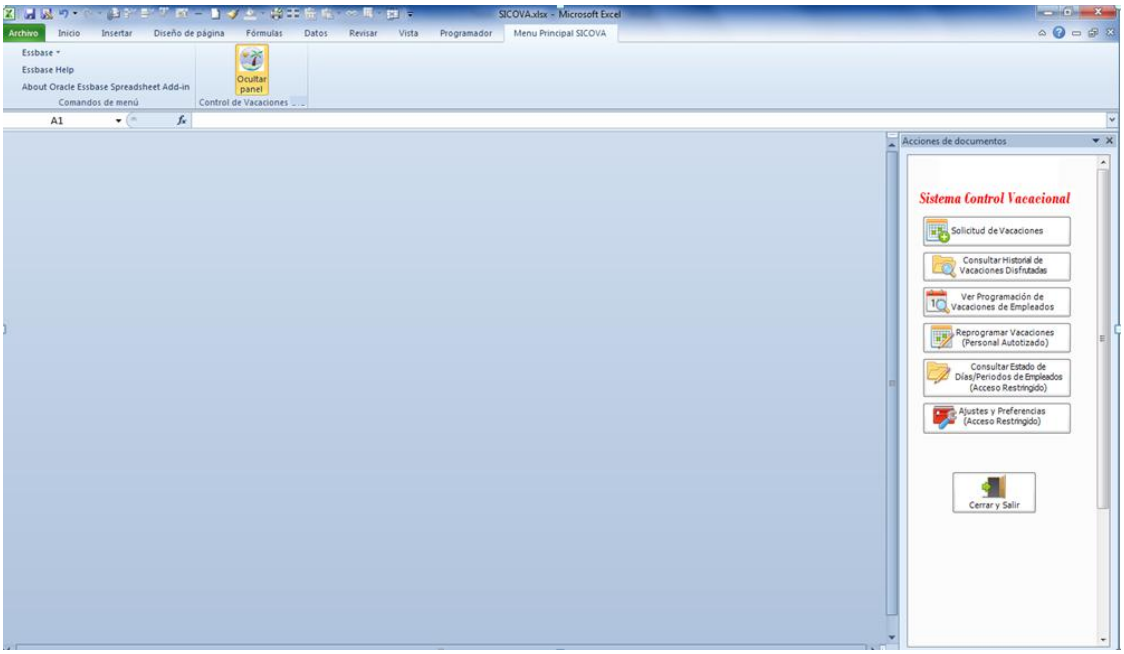

Figura 3.25 Add-in Sistema Control Vacacional

Finalmente el seguimiento del Sprint quedo como lo muestra la siguiente figura (véase Figura 3.26 y 3.27):

Sprint 1				Día1	Día2	Día3	Día4	Día5	Día6	Día7	Día8	Día9	Día10	Día11	Día12	Día13	Día14	Día15	Día16	Día17	Día18	Día19	Día20	Día21	Día22	Día23	Día24
		Hrs.Pendientes		127	119	110	102	93	85	76	68	60	52	44	37	29	20	18	16	14	12	10	8	6	4	2	0
# Actividad	Descripción	Prioridad	Estado	Horas de trabajo																							
1	Armar la información	1	Terminado	20	12	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Base de datos	2	Terminado	5	5	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	Menú en panel de acceso	3	Terminado	4	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Inicio de sesión y roles	4	Terminado	10	10	10	9	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	Altas, cambios, bajas	5	Terminado	15	15	15	15	15	12	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Solicitud de vacaciones	6	Terminado	15	15	15	15	15	15	15	10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	Historial de vacaciones	7	Terminado	6	6	6	6	6	6	6	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	Programación de empleados	8	Terminado	8	8	8	8	8	8	8	8	8	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0
7	Cancelación de solicitud de vacaciones	9	Terminado	6	6	6	6	6	6	6	6	6	6	3	0	0	0	0	0	0	0	0	0	0	0	0	0
8	Consulta de días y periodos	10	Terminado	4	4	4	4	4	4	4	4	4	4	3	3	0	0	0	0	0	0	0	0	0	0	0	0
11	Documentación	11	Terminado	6	6	6	6	6	6	6	6	6	6	6	6	1	0	0	0	0	0	0	0	0	0	0	0
12	Pruebas unitarias	12	Terminado	8	8	8	8	8	8	8	8	8	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0
13	Pruebas integración	13	Terminado	20	20	20	20	20	20	20	20	20	20	20	20	20	20	18	16	14	12	10	8	6	4	2	0

Figura 3.26 Seguimiento del Sprint

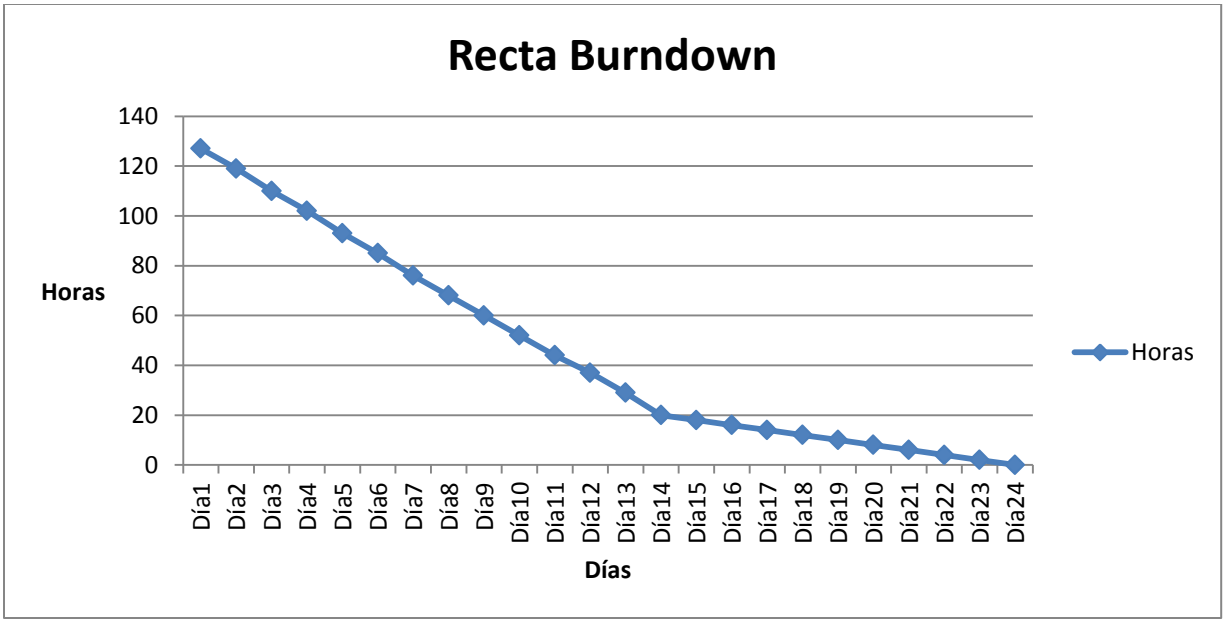

Figura 3.27 Recta Burndown

CAPÍTULO 4:

PRUEBAS, IMPLEMENTACIÓN Y RESULTADOS

4.1 Pruebas

Antes de implementar el desarrollo, realicé pruebas unitarias y de integración para comprobar el buen funcionamiento de cada uno de los módulos que conforman la aplicación.

Cuando terminé de programar cada módulo, realicé las pruebas de carga de información, para determinar si las entradas y salidas de datos eran las correctas.

Estas pruebas unitarias fueron correctas y satisfactorias.

Cuando terminé de programar toda la aplicación, realizamos en conjunto con algunos compañeros del área pruebas de integración.

Para ello participaron durante dos semanas, algunos compañeros del área y la asistente, a efecto de encontrar errores en el proceso que pudieran llegar a surgir en la utilización del sistema.

De lo anterior no fueron encontrados errores ni en la captura de datos de los empleados ni en el registro de sus días, tampoco en la solicitud de días, ni en su seguimiento.

Únicamente modifique la interfaz gráfica del formulario de la solicitud de vacaciones puesto que causaba confusión en los empleados, además de que las interfaces gráficas tenían una letra pequeña lo que hacía difícil la lectura para algunos empleados.

Durante el tiempo mencionado, el sistema en pruebas, funcionó con los requerimientos establecidos al inicio.

Después de este tiempo, el sistema tuvo una aceptación satisfactoria por todos los empleados.

4.2 Implementación

Después subí al servidor compartido, el paquete final con el ejecutable, para que los empleados lo descargaran, instalaran y lo utilizaran.

La base de datos la subí a otro servidor donde están únicamente las bases de datos del área.

Capacité a los empleados del área en 2 sesiones diferentes, y en dicha capacitación, mostré como descargar, instalar el add-in, así como usar todos y cada uno de los módulos.

4.3 Resultados

A partir del momento en que el sistema quedo en uso por el personal, pude observar el gran cambio que representaba usar el add-in, frente a la problemática que existía anteriormente.

Ahora el proceso es automático, y los empleados pueden usar el sistema con mucha facilidad, evitando la confusión y problemas que surgían con el proceso realizado de manera manual.

Las ventajas y beneficios del nuevo sistema frente al proceso anterior, destacan:

- Mejor disponibilidad de la información.
- Mejor control de la información
- Mejor integridad y organización de la información.
- Menor tiempo de búsqueda de formatos vacacionales.
- Reducción significativa de errores en la elaboración del formato y la solicitud de días de vacacionales.
- Simplificación significativa para actualizar datos del personal.
- Mejora en manipulación de información al tener una base de datos relacional.
- Histórico de solicitudes vacacionales.
- Mayor control y atención en los periodos vacacionales a vencer del personal a efecto de reducir su carga laboral para que disfrute sus vacaciones.

En relación a los diferentes roles establecidos, existe un mejor mecanismo de acceso a la información, puesto que cada empleado puede ver y manipular información en relación a su nivel.

CAPÍTULO 5:

CONCLUSIONES

La carrera de ingeniería en computación ofrece una gran variedad de conocimientos de ciencias básicas, ciencias de la ingeniería, ingeniería aplicada, humanidades y ciencias sociales, mismas que permiten al egresado estudiar problemas de diversa índole, a efecto de proponer una solución desde una perspectiva tecnológica, a través del análisis, diseño, construcción e implementación de algún desarrollo.

El estudiar Ingeniería en computación en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, me ha permitido contar con conocimientos sólidos acerca de asignaturas que utilicé a lo largo de mi experiencia laboral como, Ingeniería de software, Administración de proyectos de software, Negocios electrónicos, mismos que me han permitido modelar, planificar el proyecto a través de la metodología que implementé en la empresa, la planificación y el seguir una metodología de desarrollo es fundamental puesto que sin ella sería imposible que el producto final fuera terminado correctamente. Así como Bases de datos que me permitió modelar, diseñar y hacer la estructura donde residen los datos del programa.

Las asignaturas de computación para ingenieros, programación avanzada y métodos numéricos y computo móvil me dieron las bases para desarrollar software, si bien, en dichas asignaturas los lenguajes de estudio con C y Java, pero me otorgaron el conocimiento y estudio del paradigma orientado a objetos, además de que la sintaxis de Java es similar a la de C#.

Es importante que el ingeniero en su actividad profesional, se relacione, comunique, trabaje y colabore con personas en equipo, debido a que en una empresa, difícilmente puede satisfacer por sí mismo, las necesidades que se tengan para llevar a implementar un proyecto.

Si bien, en el proyecto explicado en el presente informe fue desarrollado por una persona, sin embargo, en la etapa de pruebas colaboré con demás miembros del área a efecto de determinar entre otras cosas si era realmente lo que el usuario necesitaba.

Además de que al inicio escuche los comentarios de los empleados para analizar y diseñar una adecuada solución.

Me siento contento y agradecido por haber ingresado al Grupo Financiero y al área de Planeación estratégica por otorgarme la oportunidad de laborar en ella. En estos tiempos de gran avance tecnológico es importante la automatización de actividades, para que las personas de negocio en vez de procesar información, estudien y tomen decisiones estratégicas de negocio a efecto de crecimiento en ventas y participación de mercado de una empresa y sus productos.

Aprendí una nueva herramienta como el uso de Visual Studio Tools for Office, para la creación de add-ins y comprendí la importancia que tienen los add-ins a la ofimática en una empresa donde el ambiente es principalmente de negocio, planeación y administración.

Referencias

Bibliográficas

Date, C.J., Introducción a los SISTEMAS DE BASES DE DATOS; Pearson Prentice Hall, 7ma Edición, México, 2001.

Dewson, Robin, Beginning SQL Server 2012 for Developers, 3rd Edition, Apress, 1st Edition, USA 2012.

Atkinson Paul, Vieira Robert, Beginning Microsoft SQL Server 2012 Programming, Wrox, 1st Edition, USA 2012.

Carter Eric, Lippert Eric, Visual Studio Tools for Office 2007, VSTO for Excel, Word, and Outlook, Addison-Wesley, 1st Edition, USA 2009.

Greene Jennifer, Stellman Andrew Head First C# A Learner's Guide to Real-World Programming with C#, XAML, and .NET, O'Reilly Media 3rd Edition, USA 2013.

Palacio Juan, Flexibilidad con Scrum Principios de diseño e implantación de campos Scrum, 2da edición, España 2008. Navegapolis.

Digitales

https://msdn.microsoft.com/es-ES/office/hh133430.aspx Página de desarrollo de aplicaciones para office Oficial de Microsoft Corporation visitada durante todo el desarrollo.

https://msdn.microsoft.com/es-ES/office Página del centro de desarrollo para Office 2013, última visita 11/12/2014.

http://msdnfan.blogspot.mx/2005/02/pltica-de-visual-studio-tools-for.html Blog de desarrolladores de Microsoft, última visita 15/12/2014.

https://msdn.microsoft.com/es-ES/ Página de Microsoft Developer Network visitada durante todo el desarrollo, para información de c#, .NET, VSTO, visual studio 2012.

https://msdn.microsoft.com/es-es/library/d2tx7z6d.aspx Página de Desarrollo para Office y Sharepoint, última visita en 12/11/2014.

http://blogs.msdn.com/b/officeapps/archive/2013/06/18/roadmap-for-apps-for-office-vsto-and-vba.aspx Blog de Office Developers, última visita 21/11/2014.

https://msdn.microsoft.com/es-es/magazine/cc507632.aspx Página de MSDN Magazine, última visita en 7/01/2015.