UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

"SISTEMA DE INFORMACIÓN DE RECURSOS HUMANOS"

CASO: SENASIR – Servicio Nacional del Sistema de Reparto

Para optar al título de Licenciatura en Informática mención en Ingeniería de Sistemas Informáticos

Postulante: Dora Rosalía Quino Huasco

Tutor: MSc. Mario Loayza Molina

Revisor: Lic. Celia Elena Tarquino Peralta

La Paz – Bolivia

2009

Dedicatoria.

Con mucho amor dedico este proyecto a quien un día me dijo;"Mira que te mando que te esfuerces y seas valiente. No temas, ni desmayes; porque yo, tu Dios, estaré contigo donde quiera que vayas. "

A mi papá Nicolás y a mi mamá Lucia, quienes me apoyaron con amor y comprensión estos años de estudio. Gracias por hacer posible uno más de mis sueños.

A mis hermanos Martin, María, Marco y David, por la compañía y amor incondicional brindado en cada paso de mi vida.

Agradecimientos.

Contar con el conocimiento, experiencia y consejos de mi tutor, revisora y supervisores fue fundamental para el desarrollo de este proyecto.

Las instrucciones y consejos brindados alinearon el rumbo correcto para la conclusión de este proyecto, gracias MSc. Lic. Mario Loayza.

La paciencia, tiempo, conocimientos, enseñanzas, y consejos que compartió conmigo fueron pieza clave para la conclusión de este proyecto, de todo corazón muchas gracias Lic. Celia Tarquino Peralta.

El principio de una realidad son los sueños; la oportunidad y confianza que me brindaron hicieron de este principio una realidad. Ing. René Arduz gracias por su confianza y por abrirme las puertas en la institución. Lic. Eddy Vera gracias por compartir sus conocimientos, experiencia laboral, consejos, y su amistad. Hermosos recuerdos, experiencias y nuevas oportunidades son el resultado de lo vivido con todo el personal de la Unidad de Sistemas del SENASIR.

RESUMEN

"La administración de recursos humanos no constituyen un fin en si misma, sino un medio para alcanzar la eficacia y eficiencia de las organizaciones a través del trabajo de las personas, y para establecer condiciones favorables que les permitan conseguir los objetivos individuales" [1].

Servicio Nacional del Sistema de Reparto (SENASIR), es una institución gubernamental establecida en Bolivia, uno de los objetivos que pretende alcanzar es la de "ser la institución pública más transparente, eficiente y moderna del país contando con personal altamente capacitado y comprometido" [2].

El presente proyecto pretende coadyuvar a este fin, a través de la implementación de un Sistema Informático, el cual permite la automatización de procesos referentes a la administración de recursos humanos, facilitando el control del cumplimiento de normas y reglamentos vigentes en la institución, además del registro en una base de datos de los funcionarios de todas sus dependencias a nivel nacional, ofreciendo información ordenada, completa, oportuna para la toma de decisiones, de facil acceso, datos actualizados e información confiable.

De esta manera el Sistema de Información de Recursos Humanos (SIREHU) aporta a una administración transparente, eficiente, eficaz, moderna y la conformación de recursos humanos competente y calificado dentro la institución. Para este propósito se utilizará los recursos tecnólogicos y métodos apropiados para su desarrollo e implementación del proyecto presentado.

INDICE

Anexo B: Árbol de problemas

Anexo C: Árbol de objetivos

Anexo D: Marco lógico

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO I

INTRODUCCION

1.1. ANTECEDENTES

1.1.1. DE LA INSTITUCIÓN

El sistema de seguridad social en Bolivia fue creado en 1956 con la promulgación del Código de Seguridad Social el 14 de diciembre de 1956. El sistema de pensiones denominado de "Reparto Simple", delegó su administración a la Caja Nacional de Seguridad Social, además de Cajas de Pensiones y Salud y a los Fondos Complementarios Sectoriales alrededor de 30 años.

A partir de esa fecha, se sucedieron unos cúmulos de acciones que por diferentes motivos, entre ellos políticos, sociales, económicos, etc. este sistema tuvo varias denominaciones como FONARE, FOPEBA, Secretaría Nacional de Pensiones, Dirección General de Pensiones.

En Noviembre de 1996, los trabajadores discontinuaron sus contribuciones al antiguo sistema para empezar a hacerlas con un administrador de fondos de pensiones (AFP) y los recién incorporados al mercado laboral directamente comenzaron sus contribuciones a dichas AFPs.

Con la Ley de Organización del Poder Ejecutivo (LOPE) en fecha 16 de septiembre de 1997, se creó la Dirección General de Pensiones dependiente del Ministerio de Hacienda. Posteriormente a partir de la promulgación del decreto Supremo No. 26189 de

18 de mayo de 2001, pasa a ser una institución de Derecho Público con personería jurídica, desconcentrada del Ministerio de Hacienda.

Al no existir una mejora administrativa fundamental, el Gobierno promulga el Decreto Supremo No. 27066 de 6 de junio de 2003 creando el Servicio Nacional del Sistema de Reparto (SENASIR) bajo tuición del Ministerio de Hacienda estableciendo su naturaleza y funcionamiento institucional a cargo de un Director General Ejecutivo que ejerce como Máxima Autoridad Ejecutiva.

La Mision del SENASIR es la de facilitar, otorgar y administrar las prestaciones de largo plazo del sistema de reparto de manera eficiente y equitativa, preservando y velando por los derechos de los beneficiarios.

1.1.2. DE LA UBICACIÓN FÍSICA

El proyecto comprende la gestión de recursos humanos, de las diferentes unidades organizacionales del SENASIR a nivel nacional, estas reparticiones organizacionales se encuentran ubicadas geográficamente distantes, a continuación citamos las dependencias correspondientes en la ciudad de La Paz.

- Cuenta Individual - Edificio FOCSAP.
- Unidad de Fiscalización - Edificio Ex – BBA.
- Unidad Nacional de Operaciones - Miraflores.
- Central - Sopocachi.

En el Anexo A, se puede observar el organigrama de la institución, identificando su lugar de funcionamiento respectivamente. A nivel nacional se tiene las siguientes agencias regionales; Cochabamba, Santa Cruz, Camiri, Oruro, Llallagua - Uncía, Potosí, Atocha, Tupiza, Uyuni, Villazón, Sucre, Camargo, Tarija, Bermejo, Villamontes, Yacuiba, Trinidad, Riberalta, Guayaramerin, Cobija, Aiquile, Magdalena, Puerto Suarez, Robore, S.J. de Chiquitos, S.I. de Velasco, San Matías, Santa Ana, Vallegrande.

El sistema será desarrollado en la Unidad de Sistemas, ubicado en el 7mo piso del edificio Central en Sopocachi, ubicado entre las calles Presbítero Medina Nro. 2491 esq. Pedro Salazar.

1.2. ANTECEDENTES DE SISTEMAS AFINES

Resultados de una investigación preliminar, dieron a conocer la existencia de algunos sistemas informáticos realizados en gestiones anteriores, que hacen referencia a la administración de recursos humanos. A continuación citamos dos de los más relevantes.

- **Sistema de información para el departamento de recursos humanos administrativos – U.M.S.A.** Sistema que fue desarrollado por la egresada de Informática Adriana Sandra Balboa Paz en el año 2007, con los objetivos específicos de agilizar el manejo de información, registro, modificación y obtención de reportes de datos personales, laborales, educativos, formación académica y movilidad. El sistema esta conformado por los siguientes componentes; designación de funcionarios, movilidad de funcionarios y registro de evaluación.
- **Sistema de administración de recursos humanos para el CEMSE,** fue desarrollado por la univ. Felipa Chávez Alanoca en el año 2007, el sistema fue implementado con el objetivo de facilitar el flujo de la información para mejorar la administración de recursos humanos, los objetivos específicos alcanzados por el sistema son; registro, actualización y consulta de los datos del funcionario, reportes de horas trabajadas, cargos y elaboración de informes. Los módulos que conforman el sistema son; registro de personal, registro laboral, registro de asistencia, permisos, capacitación, informes y evaluación.

El presente proyecto presentado, Sistema de Información de Recursos Humanos (SIREHU), además del registro de personal, también realiza la automatización de los

procesos de control de asistencia, ya que el sistema cuenta con una estructura de base de datos que permite la utilización de la tecnología de "Control Biométrico – Huella Digital", permitiendo de esta forma controlar automáticamente procesos como; asistencia, atrasos, faltas, permisos, vacaciones, llamadas de atención, etc., también controla y realiza el seguimiento correspondiente a estructuras organizacionales, unidades, puestos, cargos, escalar salariales y seguimiento a movimientos horizontales y/ o verticales del funcionario.

1.3. PLANTEAMIENTO DEL PROBLEMA

Se identificaron varios problemas latentes en el área de recursos humanos, se presenta estos problemas estructurados en tres puntos principales, para un mejor entendimiento [3]. En el anexo B, se puede observar el árbol de problemas [4].

- Existen diferentes y variados procesos referentes a la gestión de recursos humanos.
  - Registro incompleto del 'file' del personal de la institución a nivel nacional, no disponible oportunamente, difícil acceso, puede contener información desactualizada, en consecuencia no todos los funcionarios que trabajan en la institución son personal calificado.
  - En el proceso de control de asistencia del personal se invierte tiempo y recursos innecesarios, debido a que el registro de sus horas de ingresos y salidas es manual. Proceso que se puede observar gráficamente en la Figura 1.1. Esta situación, conlleva un inadecuado control del personal; asistencias, faltas, atrasos, permisos y/o licencias, refrigerios son generados a través de información que puede tener errores, duplicidad, pérdida de datos, generando información incorrecta y bajo el riesgo de que no esté disponible al momento requerido.
  - El cómputo de vacaciones se realiza a nivel nacional y se entregan informes de asistencia, permisos y/o licencias de parte de los administradores de las oficinas regionales en el interior y dependencias en La Paz, en forma escrita (informes),

esto implica información poco confiable debido a la transcripción de datos y cálculo manual de vacaciones que se debe realizar.

- El volumen de la información que se gestiona es de un aproximado de 600 funcionarios a nivel nacional, que prestan sus servicios en las diferentes unidades organizacionales en La Paz y en el interior del País, situación que dificulta la administración y control del personal, no permitiendo obtener información oportuna, precisa, actualizada y correcta.

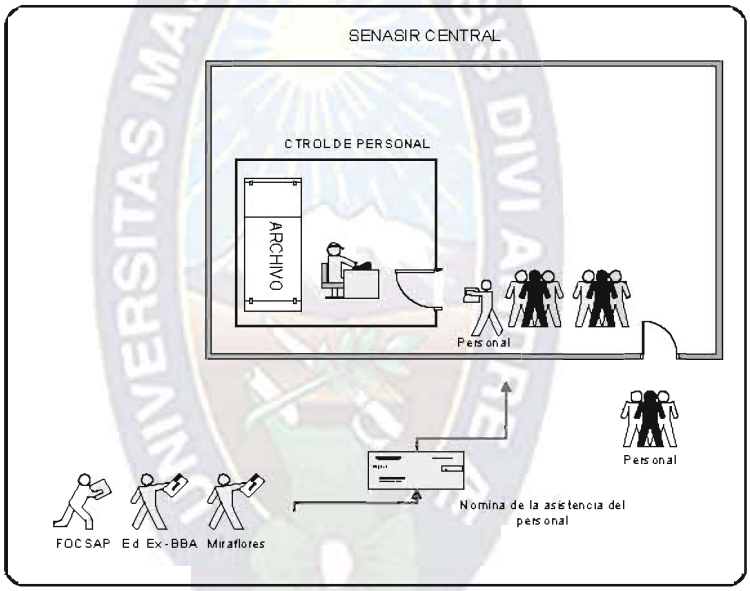


Figura 1.1. Proceso de Control de Asistencia Manual.
Fuente: [Elaboración propia]

- Se desconoce la trayectoria laboral del funcionario, debido a que no se realiza el seguimiento y control adecuado desde su ingreso hasta su salida de la institución, desconociendo los antecedentes históricos de movimientos organizacionales[1] del funcionario. En parte, debido a que no se cuenta con un histórico de estructuras organizacionales[2] y/o escalas salariales que cambiaron a través del tiempo en la institución.

1 *Designaciones, reasignaciones, retiros, renuncias, nuevos contratos, adendas, rescisiones.*
2 *Manejo de Unidades organizacionales, puestos, cargos y niveles salariales.*

- Los procesos, procedimientos y/o tareas administrativas de recursos humanos son propensos a cometer infracciones a lineamientos legales, no habiendo un control adecuado y automatizado para evitar estas faltas.
- No se tiene garantizada la seguridad lógica de los datos. La información de los 'files' del personal, documentación referente a designaciones, nuevos contratos entre otra documentación confidencial no se encuentra respaldada y/o resguardada debidamente.

1.4. OBJETIVOS

1.4.1 OBJETIVO GENERAL

Desarrollar e implementar un Sistema de Información de Recursos Humanos (SIREHU) que garantice una eficiente y eficaz administración de los recursos humanos, con el objeto de coadyuvar al alcance de los objetivos del SENASIR[3].

En el anexo C, se puede observar el árbol de objetivos y en el anexo D la matriz del marco lógico, para una mejor comprensión del objetivo que se espera alcanzar con la implementación del sistema [5].

1.4.2. OBJETIVOS ESPECÍFICOS

- Aportar al logro de una organización conformada por personal calificado, a través del seguimiento y control adecuado de la evaluación del personal, y además del registro actualizado, disponible y completo del 'file' del personal de planta y/o contrato de la institución a nivel nacional.

3 *Objetivos del SENASIR fueron mencionados en la Introducción de este Capítulo 1*

- Automatizar los procesos de control de personal, permitiendo a través de la estructura del sistema utilizar la tecnología de "Control Biométrico – Huella Digital", permitiendo de esta manera automatizar procesos como asistencias, atrasos, faltas, abandonos, permisos y/o licencias, refrigerios, generando sus correspondientes sanciones y/o llamadas de atención, considerando la definición de horarios y tolerancias.
- Automatización del cálculo de vacaciones, según permisos y/o licencias emitidas, Revolucionando de esta forma el manejo de la información de manera ágil, funcional, disminuyendo tiempos y aumentando la productividad del talento humano.
- Controlar y hacer el seguimiento adecuado de los movimientos laborales del funcionario, tanto vertical como horizontal, desde el momento de su ingreso hasta su retiro, llevando un seguimiento adecuado de los movimientos organizacionales; como designación, reasignación, retiro, renuncia, nuevos contratos, adendas y rescisiones. Permitiendo además el manejo de estructuras organizacionales (unidades, puestos, cargos, escalas salariales, ítems), generando de esta forma, información respecto a la línea de tiempo laboral del funcionario, estado de las unidades organizacionales; ocupadas o acéfalos y la conformación de personal en cada una de estas reparticiones.
- Controlar los procesos de gestión de recursos humanos, validando en cada uno de los módulos del sistema, el cumplimiento de los lineamientos legales[4] que atribuyen a la administración de recursos humanos.
- Garantizar la seguridad lógica de la información almacenada en el sistema de información, permitiendo solo el acceso a usuarios autorizados, según niveles de perfil.

4 Para el control de personal, es indispensable la correcta aplicación del Reglamento Interno de Personal del SENASIR – RIP. También se considera las Normas Básicas del Sistema de Administración de Personal

- Brindar reportes dinámicos, generación automática de Memorándums (en una interfaz Microsoft Word), exportación de datos a Microsoft Excel, según se requieran en cada módulo, para la toma de decisiones futuras, a nivel superior, operativo y ejecutivo. Ofreciendo disponibilidad de información al momento requerido.

1.5. JUSTIFICACIONES

1.5.1 JUSTIFICACIÓN ECONÓMICA

El funcionamiento del sistema permitirá mejorar, optimizar y disminuir riesgos en los procesos administrativos de los recursos humanos en el SENASIR.

En la institución existen todos los recursos y herramientas necesarias para el desarrollo, implementación y funcionamiento del sistema de información, desde equipos de computación adecuados hasta personal calificado para el manejo del sistema informático.

1.5.2. JUSTIFICACIÓN TÉCNICA

El proyecto se justifica por que existen computadoras actualizadas que cumplen los requisitos mínimos de hardware y software para la implementación del sistema tanto en la Unidad de Sistemas y como en las reparticiones de la Unidad de Desarrollo Organizacional, también se cuenta con la tecnología "Control Biométrico – Huella Digital" para el control de asistencia de los funcionarios,

A continuación detallamos los requisitos mínimos de software y hardware para el desarrollo, implementación y funcionamiento del sistema informático.

Hardware: Los requerimientos mínimos de hardware para el funcionamiento del sistema están directamente relacionados con la plataforma y el entorno de trabajo. Mínimamente se requiere PC Pentium IV y 256 de Memoria.

Software: Requerimientos mínimos de software para que el sistema pueda funcionar de manera adecuada.

SIREHU será compatible con cualquier máquina cuyo sistema operativo sea Windows XP (Service Pack 2) o Windows Vista (Service Pack 1), además que tenga instalado Microsoft Office 2003 o versiones posteriores.

1.5.3. JUSTIFICACIÓN SOCIAL

El desarrollo del SIREHU, ofrece beneficios a la institución, contribuyendo con mejoras en los procesos, procedimientos administrativos de recursos humanos para una correcta toma de decisiones. Beneficios relevantes que son mencionadas a continuación.

- ✓ Automatización, un 85% de los procesos de administración y control de personal están automatizados gracias al SIREHU.
- ✓ Optimización de tiempos, obtención de resultados en tiempos más cortos e información precisa y confiable, permitiendo la inversión de tiempo en más tareas, logrando de esta manera un trabajo satisfactorio.
- ✓ Disponibilidad de información al momento requerido, las consultas al sistema son fáciles, y de acceso rápido, permitiendo tener información actualizada, confiable y oportuna.
- ✓ Aporte al cumplimiento de leyes, normas y/o reglamentos referentes a los recursos humanos.

Permitiendo con estas ventajas un apoyo trascendental para toma de decisiones referente a la gestión de recursos humanos en el SENASIR. Aportando de esta forma al cumplimiento de los objetivos trazados por la institución.

1.6. METODOLOGÍA Y HERRAMIENTAS

Para el desarrollo del Sistema de Información de Recursos Humanos (SIREHU), se aplicará la metodología de desarrollo de software; Feature Driven Development - FDD [6] es una metodología ágil guiada por rasgos o características. Entre otras herramientas utilizadas para el diseño del sistema; Power Designer 11.0 y VP SDE Software Design Engineer for Visual Studio 2005.

SIREHU estará implementada en una Plataforma .NET; SQL Server 2005 Gestor de la Base de Datos y el lenguaje de programación C Sharp NET con reportes en Crystal Report 11 (Visual Studio NET 2005).

1.7. ALCANCES Y APORTES

SIREHU, es un sistema informático clasificado dentro los sistemas de apoyo a las decisiones, debido que ayuda a tomar decisiones inteligentes y documentadas acerca de diversos aspectos de operación, también es operacional por que ayuda a llevar a cabo los detalles del trabajo cotidiano de la organización. Este sistema cuenta con tres módulos, descritos a continuación [7].

- **Registro de Personal;** Este módulo permite tener el registro completo de toda la información referente al funcionario y además realiza el proceso de evaluación del personal. Ver figura 1.2.
- **Control de Personal;** Permite realizar las validaciones necesarias para el control y cumplimiento de las normativas que rigen la administración de los recursos humanos. Este módulo automatiza el control de asistencia, a través del cargado y

tipificación del reloj de control de las tarjetas magnéticas. Las tareas que realiza el Módulo de Control de Personal son presentadas a continuación. Ver figura 1.3.

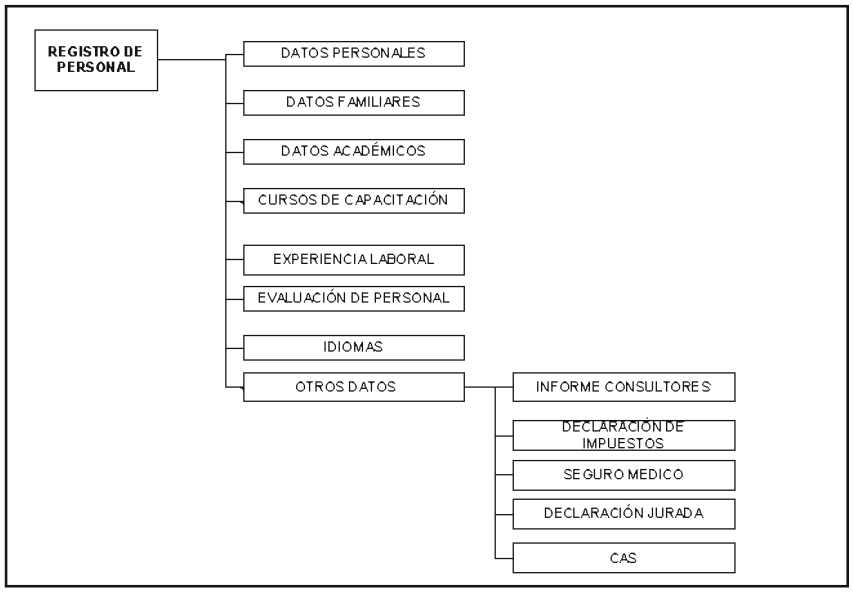


Figura 1.2. Esquema del Módulo Registro de Personal
Fuente: [Elaboración propia]

- **Procesos Organizacionales.** Encargada del manejo de estructuras organizacionales a través del tiempo, permitiendo la creación de nuevas unidades, puestos, cargos, escalas salariales e ítems respectivos. Además, realiza los movimientos (verticales y horizontales) delos funcionarios; designaciones, reasignaciones, retiros, nuevos contratos, adendas y rescisiones, este modulo permite conocer puestos acéfalos, escalas salariales históricos, recorrido laboral del funcionario; desde su ingreso hasta su retiro de la institución y otros mas reportes. Ver figura 1.4.
- **Administración de Usuarios.** Control de los usuarios del sistema de informático; permitiendo la asignación de Perfiles de Usuarios, perfiles que se

crearon bajo niveles de acceso según las necesidades en cada uno de los módulos del sistema.

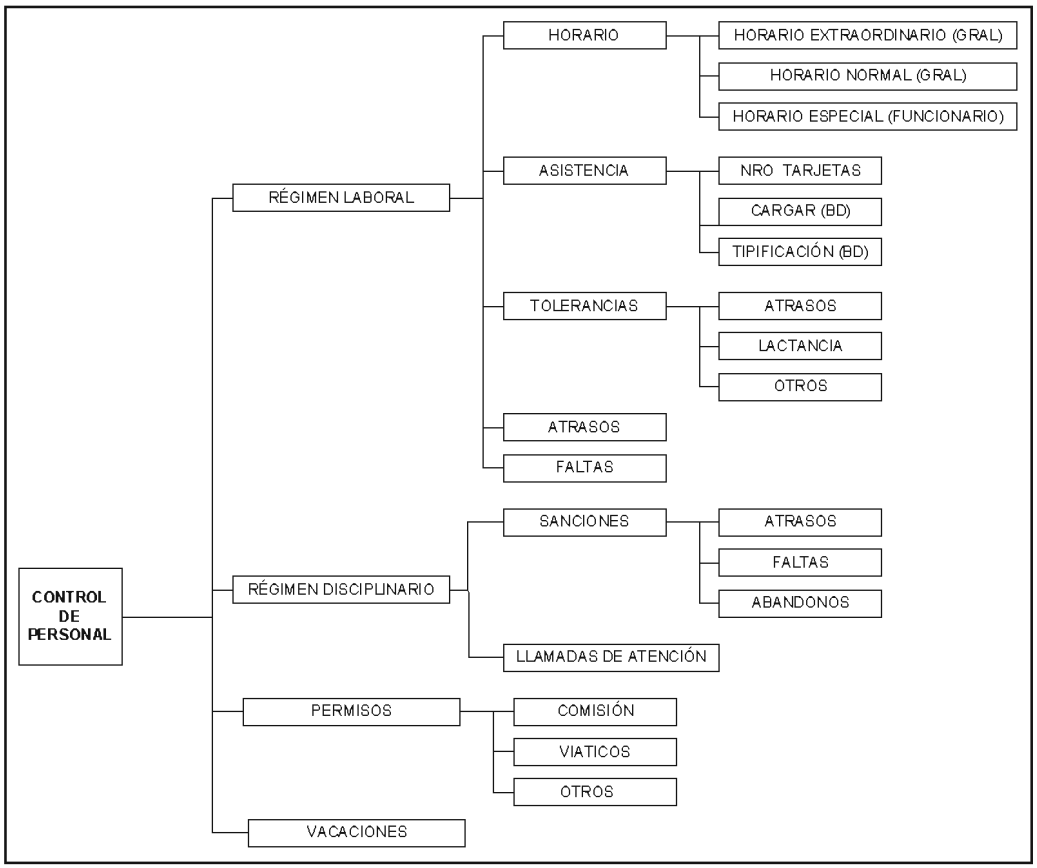


Figura 1.3. Esquema del Módulo Control de Personal
Fuente: [Elaboración propia]

Figura 1.4. Esquema del Módulo Procesos Organizacionales
Fuente: [Elaboración propia]

CAPÍTULO II
MARCO TEÓRICO

2.1. INGENIERÍA DE SOFTWARE

"Software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo" [8] y "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y costo-efectivos" [9].

El proceso de desarrollo de software "es aquel en que las necesidades del usuario son

traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "*define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo*" [10]. El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición.

2.2. UML - LENGUAJE DE MODELADO UNIFICADO

UML son las siglas de Unified Modeling Language (Lenguaje Unificado de Construcción de Modelos). Impulsado por el Object Management Group (OMG). UML se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software". Es un sistema notacional destinado a los sistemas de modelado que utilizan conceptos orientados a objetivos.

Comienzos de UML.-

UML es un estándar incipiente de la industria para construir modelos orientados a objetos. Nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar los dos famosos métodos: el de Booch y el OMT (Object Modeling Technique, Técnica de Modelado de Objetos). Más tarde se les unió Ivar Jacobson, creador del método OOSE (Object – Oriented software Engineering, Ingeniería de Software Orientada a Objetos). En respuesta a una petición de OMG (Object Management Group, asociación para fijar los estándares de la industria) para definir un lenguaje y una notación estándar del lenguaje de construcción de modelos, en 1997 propusieron el UML como candidato. Prescindiendo de la aceptación que pueda tener, este lenguaje recibió la aprobación *de facto* en la industria, pues sus creadores representan métodos muy difundidos de la primera generación del análisis y diseño orientado a objetos. [11]

Modelos y diagramas.-

- Un modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completa-mente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.
- Diagrama: una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos
- Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés
- El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos.
- En la Figura 2.1 se presenta gráficamente la relación entre diagramas.
    - Diagrama de Casos de Uso
    - Diagrama de Clases[5]
    - Diagramas de Interacción[5]; Diagrama de Secuencia y Diagrama de Colaboración
    - Diagramas de implementación[5]; Diagrama de Componentes y Diagrama de Despliegue.

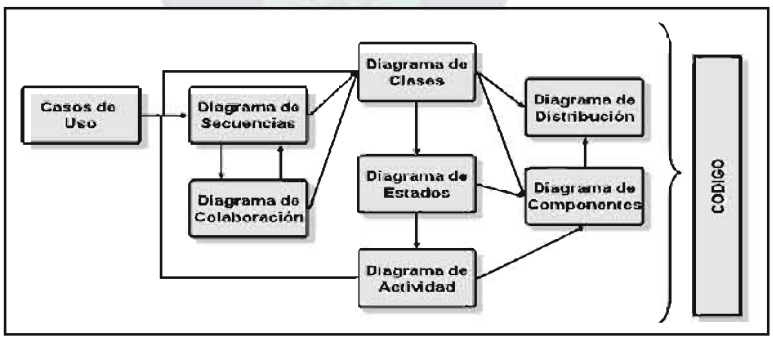


Figura 2.1. Relación entre Diagramas
Fuente: [Larman 1999]

2.3. METODOLOGÍA DE DESARROLLO DE SOFTWARE

Si se necesita de un software, ¿qué conviene?, comprar un software de mercado masivo

(25 %), adaptar una solución ya hecha (48%), desarrollar a partir de cero (180%), entre un rango de 0 a 200 %, se puede observar en la siguiente figura 2.2.

5 *agilemanifesto.org*

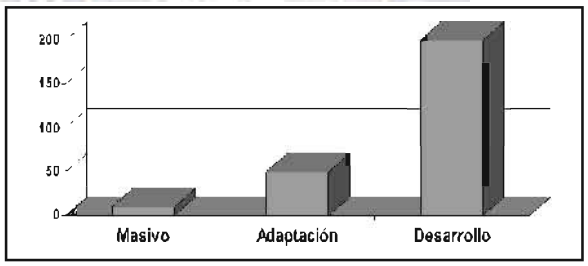


Figura 2.2.Estadistica sobre el uso de metodologías
Fuente: [AgileShift-Systems Enginneering]

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más

restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al des arrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Metodologías Agiles.- En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "*ágil*" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó "*The Agile Alliance*", dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil , un documento que resume la filosofía "*ágil*".

Principio de Manifiesto Ágil.-

- ***Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.*** La gente es el principal factor de éxito de un proyecto software.

Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- ***Desarrollar software que funciona más que conseguir una buena documentación***. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- ***La colaboración con el cliente más que la negociación de un contrato***. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- ***Responder a los cambios más que seguir estrictamente un plan***. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Estos principios son mencionados a continuación:

1) La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
2) Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

3) Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4) La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5) Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6) El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7) El software que funciona es la medida principal de progreso.
8) Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9) La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10) La simplicidad es esencial.
11) Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por si mismos.
12) En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Técnicas que permiten los Métodos Agiles.-

- **Refactorización.** Cambios en el diseño sobre el sistema implementado
- **Pruebas automáticas.** Pruebas exhaustivas del sistema cuyos resultados se comparan con resultados esperados.
- **Integración continúa.** Automatización de la integración de sistemas de modo de permitir pruebas automáticas muy frecuentes.
- **Gestión de configuración.** Hecha de una forma especial para apoyar la interacción y la integración continúa.

La Tabla 2.1. se aprecia las convergencias y divergencias en la definición de metodologías ágiles de primera línea, así como resumir sus características claves, los nombres de los promotores iníciales y sus fechas de aparición.

Entre las metodologías agiles más populares en las empresas [6] están las siguientes y sus respectivos porcentajes de uso:

- Extreme Programming (XP): 38%
- Feature Driven Development (FDD): 23%
- Adaptive Software Development (ASD): 22%
- Dynamic Systems Development Method (DSDM): 19%
- Crystal Clear (CC): 8% (CC): 8%
- Lean Development (LD): 7%
- Scrum: 3%

6 En Estados Unidos, enero de 2006, CM Crossroads realizó una encuesta sobre el uso de metodologías ágiles en el desarrollo de software, entre los lectores de Configuration Management Journal.

Metodología	Acrónimo	Tipo de Modelo	Características
Adaptive Software Development	ASD	Prácticas + Ciclo de vida	Inspirado en sistemas adaptativo complejos
Agile Modeling	AM	Metodología basada en la práctica	Suministra modelo ágil a otros métodos
Crystal Methods	CM	Familia de Metodologías	MA con énfasis en modelo de ciclos
Agil RUP	Dx	Framework / Disciplina	XP dado vuelta con artefactos RUP
Dynamic Solutions Delivery Model	DSDM	Framework / Modelo de ciclo de vida	Creado por 16 expertos en RAD
Evolutionary Project Management	Evo	Framework adaptativo	Primer método ágil existente
Extreme Programming	XP	Disciplina en prácticas de ingeniería	Método ágil radical
Feature - Driven development	FDD	Metodología	Método ágil de diseño y construcción
Lean Development	LD	Forma de pensar - Modelo logístico	Metodología basada en procesos productivos

Microsoft Solutions Framework	MSF	Lineamientos, Disciplinas, practicas	Framework de desarrollo de soluciones
Rapid Development	RAD	Surey de técnicas y modelos	Selección de best practices, no método
Rational Unified Process	RUP	Proceso Unificado	Método ¿ágil? Con modelado
Scrum	Scrum	Proceso Framework de management	Complemento de otros métodos, ágil o no

Tabla 2.1. Metodologías Agiles
Fuente: [AgileShift-Systems Enginneering]

2.4. FDD FEATURE DRIVEN DEVELOPMENT - DESARROLLO GUIADO POR FUNCIONALIDADES

FDD es una metodología de desarrollo de software ágil, FDD es también traducido como "*desarrollo guiado por rasgos y/o características*". Características importantes de esta metodología son mencionadas a continuación.

- ✓ Es un proceso ágil para el desarrollo de sistemas.
- ✓ Fue diseñado por Peter Coad, Eric Lefebvre y Jeff DeLuca. FDD es marca registrada en la empresa, Nebulón Pty. [12]
- ✓ No hace énfasis en la obtención de los requerimientos sino en como se realizan las fases de diseño y construcción.
- ✓ Disminuye el riesgo de los proyectos, pues gracias a sus entregas (resultados) periódicas, tangibles y al constante monitoreo de su calidad, se asegura el firme avance del mismo.
- ✓ Se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorear. Con iteraciones cortas, más o menos 2 semanas
- ✓ Esta pensado para proyectos con tiempo de desarrollo relativamente cortos, menos de un año.
- ✓ Centrada en el usuario, no en el programador; su objetivo es sintetizar un programa conforme a los rasgos requeridos [13]. Estos pequeños bloques son llamados "*features*", los cuales contienen la funcionalidad del sistema.

- ✓ Es adaptativo, pues permite realizar cambios de último momento debido a nuevos requerimientos y a las necesidades del negocio.

Roles y Responsabilidades.- La descripción de los roles responsabilidades de la metodología son presentadas en la Tabla 2.2.

Comparación entre FDD, XP, RUP.-

- ✓ **Tamaño de los equipos:** RUP está pensado para proyectos y equipos grandes, en cuanto a tamaño y duración. FDD y XP se implementan mejor para proyectos cortos y equipos más pequeños, siendo quizás FDD más escalable que XP.
- ✓ **Evaluación del estado del proyecto:** FDD es posiblemente el proceso más adecuado para definir métricas que definan el estado del proyecto, puesto que al dividirlos en unidades pequeñas es bastante sencillo hacer un seguimiento de las mismas. XP también define esos componentes pequeños. RUP por su parte, es tan grande y complejo en este sentido como en el resto, por lo que manejar el volumen de información que puede generar requiere mucho tiempo.

Roles Claves	Director del Proyecto	Líder administrativo y financiero del proyecto. Proteger el equipo de situaciones externas.
	Arquitecto Jefe	Diseño global del sistema Ejecución de todas las etapas
	Director de desarrollo	Lleva diariamente las actividades de desarrollo Resuelve conflictos en el equipo Resuelve problemas referentes a recursos
	Programador Jefe	Analiza los requerimientos Diseña el proyecto Selecciona las funcionalidades a desarrollar de la ultima fase del FDD
	Propietario de Clases	Responsable del desarrollo de las clases que se le asignaron Participa en la decisión de que clase será incluida en la próxima iteración.
	Expertos de dominio	Puede ser un usuario, un cliente, analista o una mezcla de estos. Poseen el conocimiento de los requerimientos del sistema Pasan el conocimiento a los desarrolladores para que se asegure la entrega de un sistema completo
Roles de Soporte	Domain Manager	Líder del grupo de expertos del dominio Resuelve sus diferencias de opinión sobre requerimientos del sistema
	Release Manager	Controla el avance del proceso mediante revisión de los reportes del Chief Programmer
	Guru del lenguaje	Responsable de tener conocimiento en lenguajes de programación o tecnología
	Ingeniero de Construcción	Responsable de preparar, mantener y correr el proceso de construcción Realiza el mantenimiento de las versiones y la publicación de la documentación
	Herramentista	Rol para la construcción de herramientas especificas para el desarrollo, conversión de datos y testeo Puede trabajar en la preparación y mantenimiento tanto de bases de datos o sitios web destinados al proyecto
	Administrador del sistema	Configura, administra y repara los servidores, estaciones de trabajo equipos de desarrollo
Roles Adicionales	Tester	Verifica que el sistema recién creado cumpla con los requerimientos del cliente
	Deployer	Es el encargado de convertir la información existente requerido por el nuevo sistema
	Escritores de documentos técnicos	Prepara la documentación para los usuarios, que pueden formar parte o no del equipo del proyecto

Tabla 2.2. Roles y Responsabilidades
Fuente: [Elaboración propia]

- ✓ **Carga de trabajo:** XP es un proceso ligero, esto es, que los creadores del proceso han tenido cuidado de no poner demasiadas tareas organizativas sobre los desarrolladores. RUP es un proceso pesado, basado mucho en la documentación, en la que no son deseables todos esos cambios volátiles. FDD es por su parte un proceso intermedio, en el sentido de que genera más documentación que XP pero menos que RUP.
- ✓ **Relación con el cliente:** Con RUP se presentarán al cliente los artefactos del final de una fase, en contrapartida, la aseguración de la calidad en XP y FDD no se basa en formalismos en la documentación, si no en controles propios y una comunicación fluida con el cliente.
- ✓ **Conocimiento sobre la arquitectura:** En RUP se intentará reducir la complejidad del software a producir a través de una planificación intensiva. En XP se conseguirá a través de la programación a pares que ya en la creación del código se puedan evitar errores y malos diseños. En FDD sin embargo se usan las sesiones de trabajo conjuntas en fase de diseño para conseguir una arquitectura sencilla y sin errores.

Punto flaco del FDD.- FDD presenta su talón de Aquiles en la necesidad de tener en el equipo miembros con experiencia que marquen el camino a seguir desde el principio, con la elaboración del modelo global.

2.4.1. FASES DE LA METODOLOGIA

El proceso consiste de cinco pasos secuénciales; las fases de ingeniería de requerimientos se encuentran fusionadas con las fases de la Metodología FDD, gráficamente se lo puede observar en la figura 2.3. La parte iterativa soporta desarrollo ágil con rápidas adaptaciones a cambios en requerimientos y necesidades del negocio. Cada fase del proceso tiene un criterio de entrada, tareas, pruebas y un criterio de salida [14].

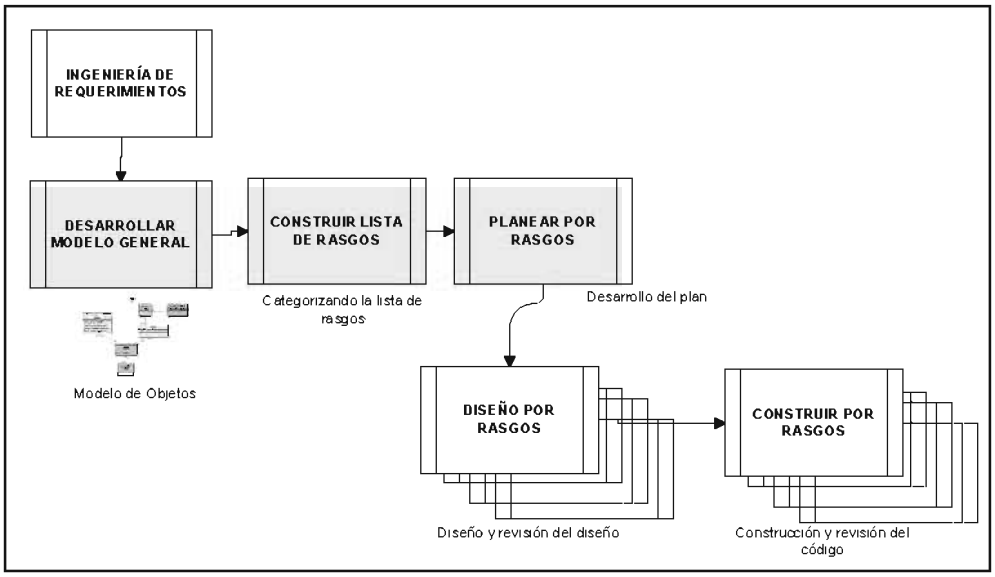


Figura 2.3. Procesos FDD
Fuente: [Stephen 2002]

2.4.1.1. FASE 1: INGENIERIA DE REQUERIMIENTOS

Los requerimientos son una descripción de las necesidades o deseos de las características que debe tener el sistema, la meta es identificar y documentar lo que en realidad se necesita, existen varias herramientas para la obtención de los mismos, cuestionarios, entrevistas, muestreo entre otras. [15].

La especificación del sistema es el producto final sobre los requisitos del sistema obtenido por el ingeniero. Sirve como fundamento para la ingeniería del hardware, ingeniería de software, la ingeniería de base de datos y la ingeniería humana. Describe la función y características de un sistema de computación y las restricciones que gobierna su desarrollo. La especificación delimita cada elemento del sistema. Describe la información (datos y control) que entra y sale del sistema. [16]

2.4.1.1.1. REQUISITOS NO FUNCIONALES.- Describe aspectos del sistema visibles por el usuario que no se relacionan en forma directa con el comportamiento funcional del sistema.

2.4.1.1.2. REQUISITOS FUNCIONALES.- Describe las interacciones entre el sistema y su entorno (usuario u otros sistemas), sin tener en cuenta cuestiones de implementación. S estudian y representan en el Modelo de Casos de Uso.

2.4.1.1.2.1. DIAGRAMA DE CASOS DE USO

Una vez recopilados los requisitos, en ingeniero de software (analistas) puede crear un conjunto de escenarios que identifiquen una línea de utilización para el sistema que va ser construido. Los escenarios, algunas veces llamados Casos de Uso [17], facilitan una descripción de cómo el sistema se usará. [18]

- Los Casos de Uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario
- Es una técnica para capturar información respecto de los servicios que un sistema proporciona a su entorno. Es una técnica para captura y especificación de requisitos.
- Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno

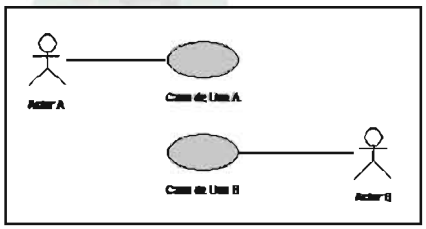


Figura 2.4. Representación de Casos de Uso
Fuente: [Larman 1999]

Actores.-

- Principales: son las personas que usan el sistema. La representación grafica se puede ver en la figura 2.4.
- Secundarios: personas que mantienen o administran el sistema
- Material externo: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros sistemas: sistemas con los que el sistema interactúa
- La misma persona física puede interpretar varios papeles como actores distintos
- El nombre del actor describe el papel desempeñado

Tipos de relación.-

- **Comunicación:** comunicación normal entre el actor y el caso de uso. Figura 2.5.
- **Inclusión**: una instancia del Caso de Uso origen incluye también el comportamiento descrito por el Caso de Uso destino. Figura 2.6.
- **Extensión**: el Caso de Uso origen extiende el comportamiento del Caso de Uso destino. Figura 2.7.
- **Herencia**: el Caso de Uso origen hereda la especificación del Caso de Uso destino y posiblemente la modifica y/o amplía. Figura 2.8.

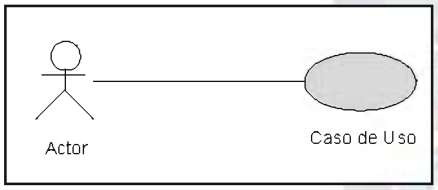


Figura 2.5. Diagrama de Caso de Uso - Comunicación
Fuente: [Elaboración Propia]

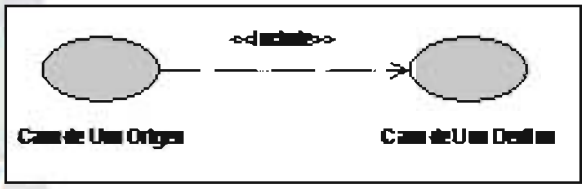
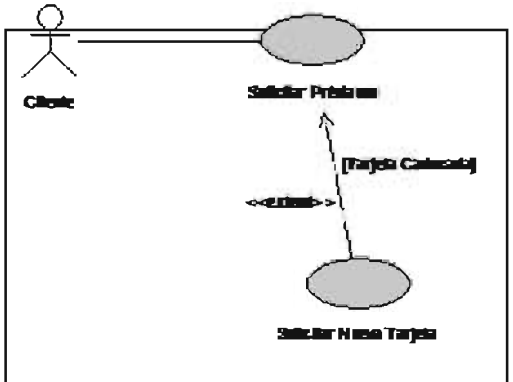


Figura 2.6 Diagramas de Caso de Uso – Inclusión
Fuente: [Elaboración Propia]



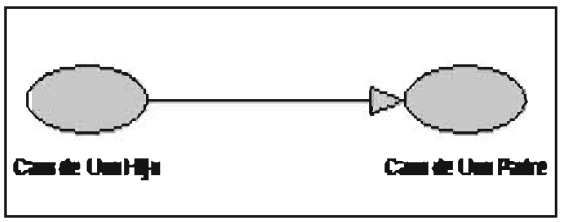


Figura 2.7 Diagramas de Caso de Uso – Extensión
Fuente: [Elaboración Propia]
Figura 2.8. Diagramas de Caso de Uso – Herencia
Fuente: [Elaboración Propia]

Construcción.-

- ✓ Un caso de uso debe ser simple, inteligible, claro y conciso
- ✓ Generalmente hay pocos actores asociados a cada Caso de Uso
  - Preguntas clave: ¿Cuáles son las tareas del actor?, ¿Qué información crea, guarda, modifica, destruye o lee el actor?, ¿Debe el actor notificar al sistema los cambios externos?, ¿Debe el sistema informar al actor de los cambios internos?
- ✓ La descripción del Caso de Uso comprende:
  - El inicio: ¿cuándo y qué actor lo produce?
  - El fin: ¿cuándo se produce y qué valor devuelve?
  - La interacción actor-caso de uso: ¿qué mensajes intercambian ambos?
  - Objetivo del caso de uso: ¿qué lleva a cabo o intenta?
  - Cronología y origen de las interacciones
  - Repeticiones de comportamiento: ¿qué operaciones son iteradas?
  - Situaciones opcionales: ¿qué ejecuciones alternativas se presentan en el caso de uso?

2.4.1.2. FASE 2: DESARROLLO DE UN MODELO GENERAL.

El análisis y diseño orientado de objetos busca ante todo descomponer un espacio de un problema por objetos y no por funciones, como se advierte en la figura 2.9.

Expertos del dominio y desarrolladores construye un modelo global del sistema. Detallados a continuación.

Entradas: El cliente está listo para comenzar con la construcción del sistema. Además se espera que existan requerimientos tales como casos de uso o especificaciones funcionales.

Verificación: Se realizará evaluaciones sobre el equipo para clarificar el entendimiento de los aspectos del dominio, la funcionalidad necesaria y el alcance. Además, se realiza controles y verificaciones sobre el modelo desarrollado. Se divide el dominio global en áreas que son analizadas detalladamente.

Salida:

- ✓ Diagrama de clases del sistema o Diagrama de Objetos por cada área.
- ✓ Lista informal de rasgos funcionales
- ✓ Registro de las alternativas más importantes sobre el modelo

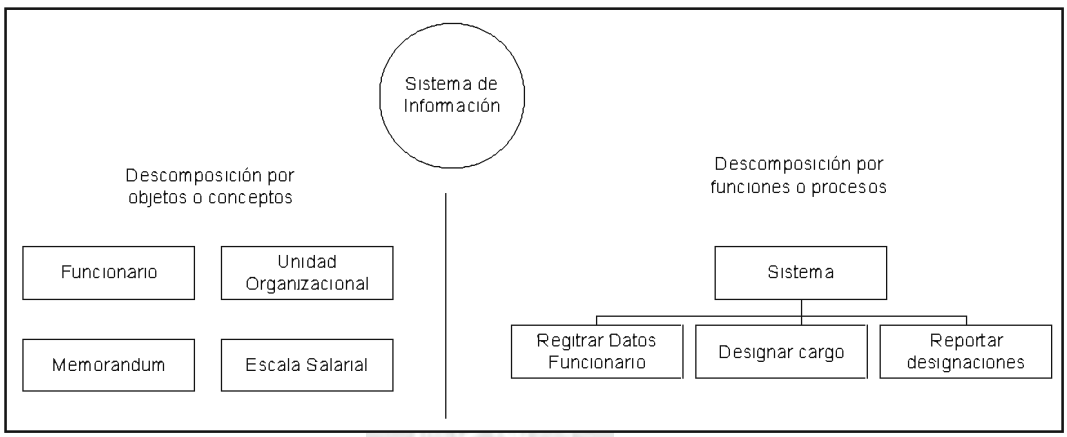


Figura 2.9. Comparación entre la descomposición OO y la descomposición orientada a funciones. Fuente: [Elaboración Propia]

2.4.1.2.1. DIAGRAMA DE CLASES

Un diagrama de clases es una descripción de las clases en un sistema y sus relaciones. Cada clase se representa en un rectángulo con tres compartimientos: nombre de la clase, atributos de la clase, métodos de la clase. Como se muestra en la figura 2.10.

Los atributos y los métodos pueden tener diferente visibilidad. Es visible si puede ser referenciado desde otras clases diferentes a donde esta definido, se definen como públicos (+), privados (-) ó protegidos (#).

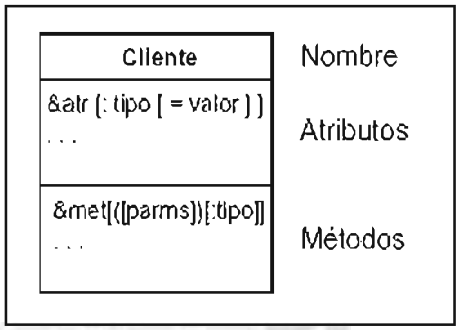


Figura 2.10. Una clase
Fuente: [Elaboración propia]

Generalización.- Indica que una "clase hijo" hereda los métodos y atributos especificados por una "clase padre", por ende la "clase hijo" además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la "clase padre", ver figura 2.11.

Agregación y Composición.- Para modelar objetos complejos, no bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades.

Composición: Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye. Es decir el objeto base se construye a partir del objeto incluido, es decir, es "parte/todo".

Agregación: Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Es decir el objeto base utiliza al objeto incluido para su funcionamiento. Ver Figura 2.12.

Asociaciones.- Este tipo de relación permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro. Ver Figura 2.13.

Dependencia.- El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana, la creación del Objeto Ventana está condicionado a la instanciación proveniente desde el objeto Aplicación. Ver figura 2.14.

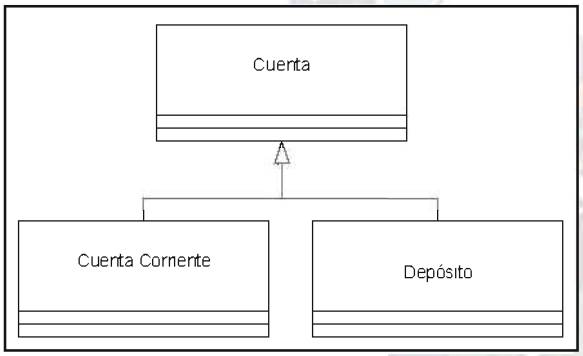


Figura 2.11 Diagramas de Clase – Generalización
Fuente: [Elaboración Propia]

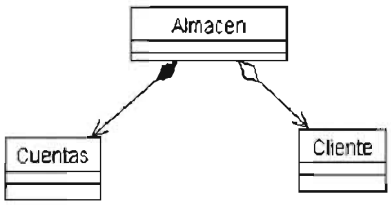
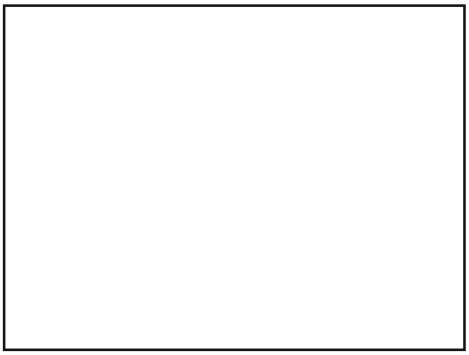


Figura 2.12 Diagramas de Clase - Composición y Agregación
Fuente: [Elaboración Propia]



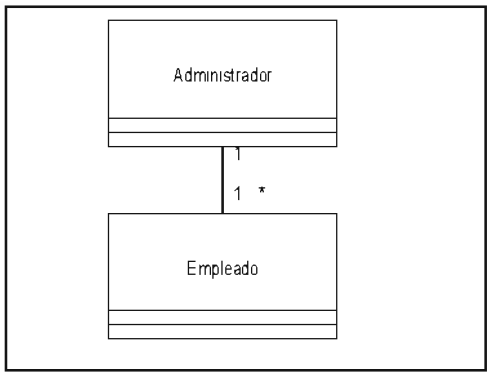


Figura 2.13 Diagrama de clases – Asociación
Fuente: [Elaboración Propia]

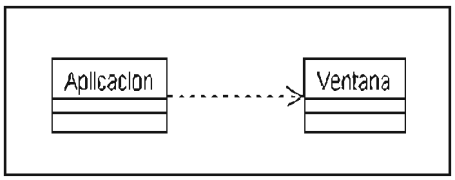


Figura 2.14 Diagrama de clases – Dependencia
Fuente: [Elaboración Propia]

2.4.1.3. FASE 3: CONSTRUCCIÓN DE LA LISTA DE FUNCIONALIDADES.

Los ensayos, modelos de objeto y documentación de requerimientos proporcionan la base para construir una amplia lista de rasgos. Los rasgos o funcionalidad son pequeños ítems útiles a los ojos del cliente, se escriben en un lenguaje que todas las partes puedan entender.

Entrada: Diagrama global del sistema y lista informal de rasgos o funcionalidades de la etapa anterior.

Verificación: Se realizan revisiones de los rasgos o funcionalidades mediante expertos del dominio.

Salida:

- ✓ Se elabora una lista de funcionalidades que resuma la funcionalidad general del sistema.
- ✓ Un rasgo o funcionalidad en FDD es una función pequeña expresada en la forma:

  <Acción> <*resultado*> <por | para | de | a> <objeto>

  con los operadores adecuados entre los términos. Por ejemplo;

  Calcular el *importe total* de una venta;

  Determinar la *última operación* de un cajero;

  Validar la *contraseña* de un usuario.
- ✓ La lista es elaborada por los desarrolladores y es evaluada por el cliente.

- ✓ Se divide la lista en subconjuntos según la afinidad y la dependencia de las funcionalidades.
- ✓ La lista es finalmente revisada por los usuarios y los responsables para su validación y aprobación.

2.4.1.4. FASE 4: PLANEAR POR FUNCIONALIDADES.

Entrada: La lista de rasgaos o funcionalidades de la etapa anterior.

Verificación: La verificación del plan se debe realizar tomando en cuenta la opinión de todos los miembros posibles.

Salida:

- ✓ Cronograma completo
- ✓ Ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia, y se asigna a los programadores jefes.
- ✓ Estas listas priorizadas en secciones que se llaman paquetes de diseño. Luego se asignan las clases definidas en la selección del modelo general a programadores individuales, o sea propietarios de clases. Se pone fechas de inicio y fin para los conjuntos de rasgos.

2.4.1.5. FASE 5: DISEÑAR POR FUNCIONALIDADES

Durante el diseño orientado a objetos, se procura definir los objetos lógicos del software que finalmente serán implementados en el lenguaje de programación orientada a objetos. Los objetos tienen atributos y métodos. Por ejemplo, el sistema de información de recursos humanos un objeto de software *funcionario* puede tener atributos de *nombres, edad* y un método de *designar*.

Entrada: Como entrada se necesita los documentos desarrollados en la fase anterior.

Verificación: Se realiza tareas de verificación de los diagramas construidos.

Salida:

- ✓ Se realizan diagramas de secuencia y colaboración para la funcionalidad.
- ✓ Se refina e actualiza el diagrama de clases.
- ✓ Descripción de clases y métodos.
- ✓ Notas del equipo que consideren significativas para el diseño.

2.4.1.5.1. DIAGRAMA DE SECUENCIAS Y DIAGRAMAS DE COLABORACIÓN

Los diagramas de interacción se realizan en la fase de diseño de un ciclo de desarrollo. Explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de éstas. El UML define dos tipos de estos diagramas; ambos sirven para expresar interacciones semejantes o idénticas de mensaje [19].

Diagramas de Secuencia.- Describe las interacciones en una especie de formato de cerca o muro.

- ✓ Muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo. Constan de objetos y representando en una línea vertical el tiempo, se indican las operaciones que ejecuta el objeto o activación se representan mediante un rectángulo cuya altura va en relación a la duración de la operación.

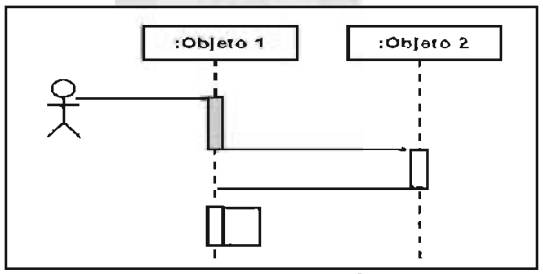


Figura 2.15. Diagrama de Secuencia
Fuente: [Letelier 2001]

- ✓ Los mensajes van de un objeto a otro se representan con líneas. Pueden ser simples (transfieren control), sincrónicos (esperan respuesta) o asincrónicos (no espera respuesta). Ver figura 2.15.

Diagrama de Colaboración.- Describe las interacciones entre los objetos entre los objetos en un formato de grafo o red. Un diagrama de colaboración permite decir más en un especio que los diagramas de secuencia y expresa además mas información contextual; tipo de visibilidad entre objetos. También resulta más fácil expresar la lógica condicional y la concurrencia. [20]

- ✓ El Diagrama de Colaboración ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos
- ✓ El Diagrama de Colaboración puede obtenerse automáticamente a partir del correspondiente Diagrama de Secuencia (o viceversa)
- ✓ El mensaje se representa como una flecha cerca de la línea de asociación entre dos objetos. Esta flecha apunta al objeto receptor. El tipo de mensaje se mostrará en una etiqueta cerca de la flecha. Ver figura 2.16.
- ✓ El mensaje le indicará al objeto receptor que ejecute una de sus operaciones.

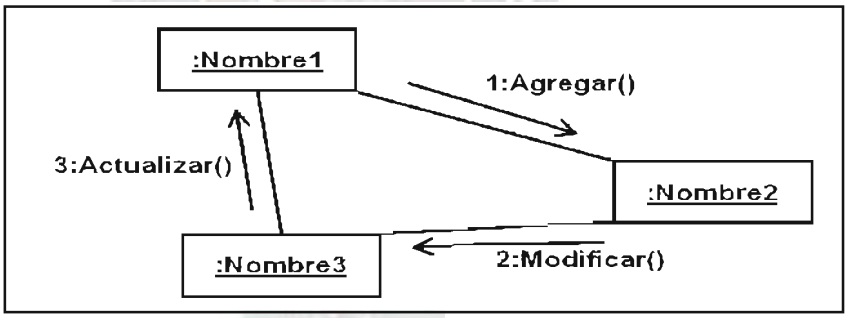


Figura 2.16. Diagrama de Colaboración
Fuente: [Letelier 2001]

2.4.1.6. FASE 6: CONSTRUCCIÓN POR FUNCIONALIDADES

Entradas: Como entrada se necesita los documentos desarrollados en la fase anterior.

Verificación: Se realiza tareas de inspección y verificación del código.

Salida:

- ✓ Se selecciona un pequeño conjunto de rasgos del conjunto y los propietarios de clases seleccionan los correspondientes equipos dispuestos por rasgos. Se procede luego iterativamente hasta que se producen los rasgos seleccionados.
- ✓ Puede haber varios grupos trabajando en paralelo. El proceso iterativo incluye inspección de diseño, codificación, prueba de unidad, integración e inspección de código. Luego de una iteración exitosa, los rasgos completos se promueven al build principal.
- ✓ Los Dueños de Clases codifican y prueban el código producido (Visualizar a través de los diagramas de Componentes y Despliegue).
- ✓ Construcción del sistema y verificación de integración.

2.4.1.6.1. DIAGRAMAS DE COMPONENTES Y DIAGRAMA DE DESPLIEGUE

Diagrama de Componentes.- Los diagramas de componentes muestra las dependencias del compilador y del "runtime" entre los componentes del software; por ejemplo, los archivos del código fuente y los DLL.

- ✓ Los diagramas de componentes describen los elementos fisicos del sistema y sus relaciones. [21]
- ✓ Muestran las opciones de realización incluyendo código fuente, binario y ejecutable
- ✓ Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc.
- ✓ Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente
- ✓ Hay diferentes tipos de componentes y cada uno con diferentes tipos de dependencia, clasificándolos en relación con el proceso de compilación un componente podría ser:

Código fuente, el cual depende de que otros componentes estén disponibles al momento de compilación.

Código objeto binario, (biblioteca de clases) que depende de cualquier código objeto al cuál se ligara desde un programa ejecutable.

Una **aplicación ejecutable**, la cuál puede depender de otros programas ejecutables al tiempo de ejecución.

- ✓ Cada clase del modelo lógico se realiza en dos componentes: la especificación y el cuerpo.
- ✓ La especificación contiene la interfaz de la clase mientras que el cuerpo contiene la realización de la clase.
- ✓ Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente se refiere a los servicios ofrecidos por otro.
- ✓ Los distintos componentes pueden agruparse en paquetes según un criterio lógico y con vistas a simplificar la implementación. Son paquetes estereotipados en «subsistemas» para incorporar la noción de biblioteca de compilación y de gestión de configuración. Se presentada en la figura 2.17.

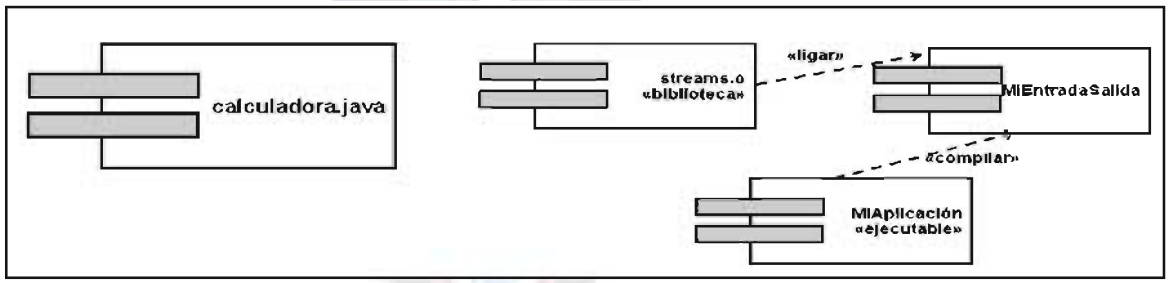


Figura 2.17. Diagrama de Componentes
Fuente:[Letelier 2001]

Diagrama de Despliegue.- Muestran a los nodos procesadores, la distribución de los procesos y de los componentes.

- ✓ Los diagramas de distribución muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos.
- ✓ Un nodo representa todo tipo de equipo de cómputo y se representa por un cubo.

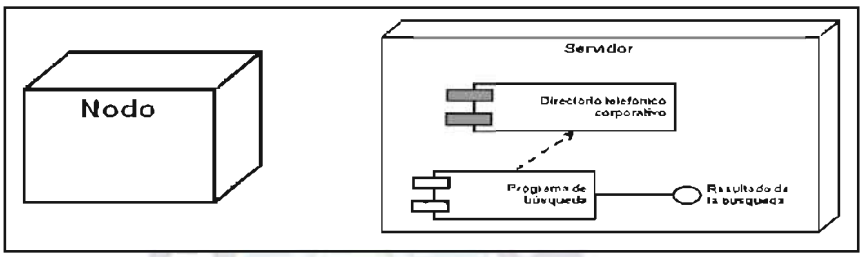


Figura 2.18. Diagrama de Despliegue
Fuente: [Letelier 2001]

- ✓ Los estereotipos permiten precisar la naturaleza del equipo: Dispositivos, Procesadores, Memoria, Etc.
- ✓ Los nodos se interconectan mediante soportes bidireccionales (en principio) que puede a su vez estereotiparse. Como se observa en la figura 2.18.
- ✓ Se pueden mostrar los componentes en relaciones de dependencia con un nodo:
- ✓ Las conexiones entre nodos se puede poner mediante una relación.

2.5. HERRAMIENTAS DE CONSTRUCCION DE SOFTWARE

2.5.1. VISUAL STUDIO NET 2005

La plataforma .NET en realidad no es algo radicalmente nuevo. Es un conjunto de tecnologías dispersas, que en muchos casos ya existían, que Microsoft ha integrado en una plataforma común con el objetivo de facilitar el desarrollo de este nuevo tipo de servicios de tercera generación. Estos son los pilares de esta nueva plataforma:

- ✓ Integración: Proporcionar mecanismos para que una empresa pueda ofrecer servicios a otras empresas o clientes de una forma sencilla y rápida. En general,

este tipo de servicios se suelen denominar B2B: *Business to Business* y B2C: *Business to Client*.

- ✓ Nuevos dispositivos: La forma más común de acceso a Internet hasta ahora ha sido el ordenador personal con sus limitaciones de movilidad. Pero recientemente han ido apareciendo una serie de dispositivos que permiten el acceso a servicios Internet de forma rápida y directa, como por ejemplo agendas electrónicas, teléfonos móviles, WebTV, videoconsolas, etc. Esto supone un cambio radical en la forma de acceder a este tipo de servicios.

.NET son aplicaciones de servidor (SQL Server 2000, Exchange 2000, etc.), es un entorno de desarrollo (Visual Studio .NET), son componentes clave que se integran al sistema operativo, son servicios, y es, finalmente, una plataforma de desarrollo denominada **framework .NET**.

Es una infraestructura de desarrollo que está compuesta por diversos recursos, entre los cuales se destaca el más importante, que es una máquina virtual conocida como **CLR** (*Common Language Runtime*), sobre la cual se ejecutan las aplicaciones.

- ✓ De este modo, nuestros programas ya no poseerán código nativo de ningún microprocesador en particular, sino instrucciones MSIL (Microsoft Intermediate Language) que serán traducidas a código nativo en el momento de su ejecución (por medio de un compilador Just In Time).

El framework también define una librería base de clases, **BCL** (*Base Class Library*), a la cual puede acceder cualquier desde lenguaje desarrollado para la plataforma. Por encima de la infraestructura se ubicará un conjunto de reglas básicas que debe implementar un lenguaje para poder ser parte de la familia .NET. Esta especificación es conocida como **CLS** (*Common Language Specificeation*).

Finalmente, se encuentra el conjunto de lenguajes que cumplan con la especificación CLS, como el C#, el VB.NET, Managed C++, etc. Es posible crear recursos en cualquiera de estos lenguajes que hagan uso de recursos escritos en otros; de hecho, es posible crear una clase en C# que herede de una clase creada en Managed C++. Todo esta estructura puede observarse en la figura 2.19.

Figura 2.19. Arquitectura de la infraestructura de desarrollo
Fuente: [Angel Ramos 2002]

Visual Studio 2005 es considerado en la actualidad como la nueva generación de tecnología de desarrollo y bases de datos de Microsoft, y fue presentado por primera vez en sociedad en el 2003 en una conferencia conocida como PDC (Professional Developers Conference).

Para poder desarrollar servicios para este tipo de arquitectura, Microsoft ha lanzado el nuevo entorno de desarrollo denominado *Visual Studio .NET*. El objetivo principal de este entorno de desarrollo es la de simplificar el desarrollo de aplicaciones Windows y servicios Web permitiendo la elección del lenguaje de programación más adecuado (Visual Basic, C++ o C#).

Además de todo este soporte, Microsoft ha desarrollado un lenguaje nuevo de programación basado en C++ denominado C# (*C sharp*). El concepto de C# es muy parecido a Java, un lenguaje que elimina las complicaciones innecesarias del C++ pero

manteniendo su potencia. El principal objetivo de C# es eliminar el uso de Java y C++, con el objetivo de reducir el coste de desarrollo de los servicios .NET.

Lenguaje C Sharp (C#) es parte de la plataforma .NET. Es un lenguaje que cumple con la especificación CLS. El código que se cree con él será traducido a instrucciones **MSIL** para entonces ser traducido, justo antes de su ejecución, a instrucciones nativas que correspondan a la plataforma concreta sobre la cual esta trabajando.

Cabe destacar que el compilador JIT (*Just In Time*) traduce el código MSIL a código nativo no de manera monolítica, sino por métodos, módulos y componentes. Por lo tanto, a grandes rasgos: código que no sea ejecutado no será compilado.

El código MSIL generado a partir de la compilación de código C# es idéntico al código MSIL generado a partir de cualquier otro lenguaje CLS. Cada lenguaje posee sus características que lo tornan ideal para ciertos usos; además, presenta diversos grados de expresividad que pueden permitir implementar el mismo algoritmo de maneras diversas, por lo que un modo puede resultar más eficiente que otro.

Características fundamentales del Lenguaje.-

- ✓ C# es un lenguaje moderno y orientado a objetos, con una sintaxis muy similar a la de C++ y Java. Combina la alta productividad de Visual Basic con el poder y la flexibilidad de C++.
- ✓ La misma aplicación que se ejecuta bajo Windows podría funcionar en un dispositivo móvil de tipo PDA. Con C#/.NET no nos atamos a ninguna plataforma en particular.
- ✓ Se puede crear una gran variedad de aplicaciones en C#: aplicaciones de consola, aplicaciones para Windows con ventanas y controles, aplicaciones para la Web, etc.

- ✓ C# gestiona automáticamente la memoria, y de este modo evita los problemas de programación tan típicos en lenguajes como C o C++.
- ✓ Mediante la plataforma .NET desde la cual se ejecuta es posible interactuar con otros componentes realizados en otros lenguajes .NET de manera muy sencilla.
- ✓ También es posible interactuar con componentes no gestionados fuera de la plataforma .NET. Por ello, puede ser integrado con facilidad en sistemas ya creados.
- ✓ Desde C# podremos acceder a una librería de clases muy completa y muy bien diseñada, que nos permitirá disminuir en gran medida los tiempos de desarrollo.

C# es un lenguaje moderno y altamente expresivo que se ajusta al paradigma de programación orientada a objetos. Su sintaxis es similar a C++ y Java. El lenguaje fue desarrollado en gran parte por Anders Hejlsberg (creador del mítico compilador Turbo Pascal1 y uno de los diseñadores líder del lenguaje de programación Delphi).

2.5.2. SQL SERVER 2005

Los clientes están buscando soluciones para sus problemas de negocios. La mayoría de las <<soluciones>> de bases de datos solamente traen múltiples niveles de costos y complejidad. La estrategia de Microsoft es la de hacer que SQL Server sea la base de datos más fácil de utilizar para construir, administrar e implementar aplicaciones de negocios. Esto significa tener que poner a disposición un modelo de programación rápido y sencillo para desarrolladores, eliminando la administración de base de datos para operaciones estándar, y suministrando herramientas sofisticadas para operaciones más complejas.

Comienzos del SQL.- A finales del año 1973 un grupo de personas trabajaba en los laboratorios de investigación de IBM Corporation, con el objetivo de desarrollar un lenguaje especifico que les permitiera acceder en forma estándar a aquellas bases de datos, que nacidas quizás bajo otro concept comenzaban a adecuarse al denominado

modelo relacional, el cual se perfilaba como un estándar de facto en la generación de modelos complejos de datos.

La primera etapa de este proyecto arrojo como resultado un lenguaje denominado SEQUEL[7] el cual fue evolucionando en forma acelerada convirtiéndose hacia el año 1977, primero en SEQUEL /2 y finalmente, por motivos legales, en SQL (Structural Query Language).

Pero no fue sino hasta entrados los años ochenta, momento en que IBM lanzara su producto de bases de datos relacional DB2 orientado a equipos de rango medio/ alto, que el lenguaje SQL se viera incorporado en la mayoría de los productos de este tipo (Sysbase, Oracle y mas tarde Informix) para finalmente convertirse en un estándar ANSI en 1986 y en estándar ISO a fines de 1987.

Hoy, "Pure SQL" ha demostrado que lejos de aquellos inicios como leguaje embebido se ha convertido en una poderosa herramienta, capaz de manejar estructuras lógicas complejas así como cualquier tipo de dato imaginado, de hecho la versión de desarrollo de denominado SQL 3 comparte tantos atributos con los lenguajes de programación convencionales, que hasta hay quienes se animan a considerar esta nueva versión como un verdadero lenguaje standalone.

---

[7] *"Structure English Query Languaje"*

Principales Características.-

- ✓ Rendimiento
- ✓ Escalabilidad
- ✓ Múltiples instancias y Failover
    - Múltiples instancias: Aislar el impacto en las aplicaciones ante fallas

- Failover: Clustering, Integrado en la instalación, Fácil aplicación de SP (Service Packs), Soporte para clusters de 4-nodos con Windows 2000 Data Center

✓ Administración de bases de datos simplificada

✓ Servicios de Transformación de datos (DTS).

✓ Servicios de análisis (OLAP)

✓ Integración con la Web; Acceso Web a datos y Soporte para XML (cláusula FOR XML)

2.6. SEGURIDAD

Desde el punto de vista de la seguridad informática, una red debe entenderse como un entorno de cómputo con más de un ordenador independiente. No obstante, la introducción de las redes informáticas no modifica tan solo cuantitativamente el problema de la seguridad, sino que supone un incremento cualitativo del mismo. No se trata tan solo de que debamos proteger un mayor número de ordenadores de un mayor número de atacantes potenciales, sino que se introducen toda una nueva serie de vulnerabilidades y amenazas, y se hacen necesarias toda una nueva serie de técnicas y herramientas para protegernos de ellas. Gráficamente en la figura 2.20 se observa algunas de las preguntas a responder en cuanto a la seguridad de la información.

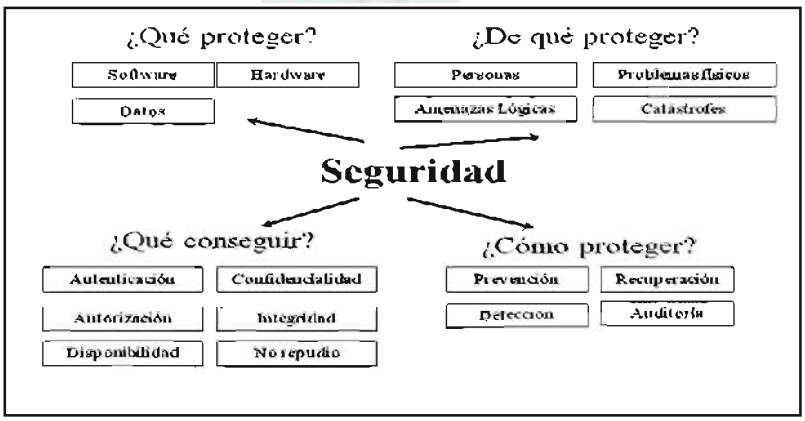


Figura 2.20. Seguridad de los Datos
Fuente: [Valdez 2005]

2.6.1. FUNCIONES HASH EN CRIPTOGRAFÍA

Probablemente el mecanismo para proporcionar seguridad en redes más utilizado es la criptografía. Este mecanismo permite crear conexiones seguras sobre canales inseguros. El uso de la criptografía nos puede proporcionar propiedades tales como la privacidad, la autenticidad, la integridad y el acceso limitado a los datos, entre otras.

Las funciones hash asignan cadenas binarias de una longitud arbitraria a cadenas binarias pequeñas de una longitud fija. Una función hash criptográfica tiene la propiedad de que, mediante el cálculo, no es posible encontrar dos entradas distintas que generen aleatoriamente el mismo valor; es decir, los valores hash de dos conjuntos de datos deben coincidir si los datos correspondientes también coinciden. Pequeñas modificaciones en los datos ocasionan grandes cambios imprevisibles en el valor hash. El tamaño del valor hash del algoritmo MD5 es de 128 bits.

Dentro del proceso de envío de login de usuario y clave al servidor de MSN, la clave debemos de encriptar para evitar que sea capturada con algún sniffer en la red. Y se usa un método un tanto artesanal de hacerlo, utilizando MD5.

Propiedades de las Funciones Hash.- h(M) será segura si tiene las siguientes características:

1. **Unidireccionalidad:** conocido un resumen h(M), debe ser computacionalmente imposible encontrar M a partir de dicho resumen.
2. **Compresión**: a partir de un mensaje de cualquier longitud, el resumen h(M) debe tener una longitud fija. Lo normal es que la longitud de h(M) sea menor que el mensaje M.
3. **Facilidad de cálculo:** debe ser fácil calcular h(M) a partir de un mensaje M.

4. **Difusión:** el resumen h(M) debe ser una función compleja de todos los bits del mensaje M: si se modifica un solo bit del mensaje M, el hash h(M) debería cambiar la mitad de sus bits aproximadamente.

Etapas del MD5.-

Gráficamente estas etapas pueden ser observadas en las figuras 2.21, figura 2.22 y figura 2.23 respectivamente.

1. Añadir bits para congruencia módulo 512, reservando los últimos 64 bits para un indicador de longitud.
2. Añadir indicación de la longitud del mensaje en los 64 bits reservados para ello.
3. Inicializar el vector ABCD de claves con un valor que no es secreto.
4. Procesar bloques de 512 bits, entregando una salida de 128 bits que formarán nuevamente el vector ABCD.
5. Obtener el resumen de los últimos 128 bits.

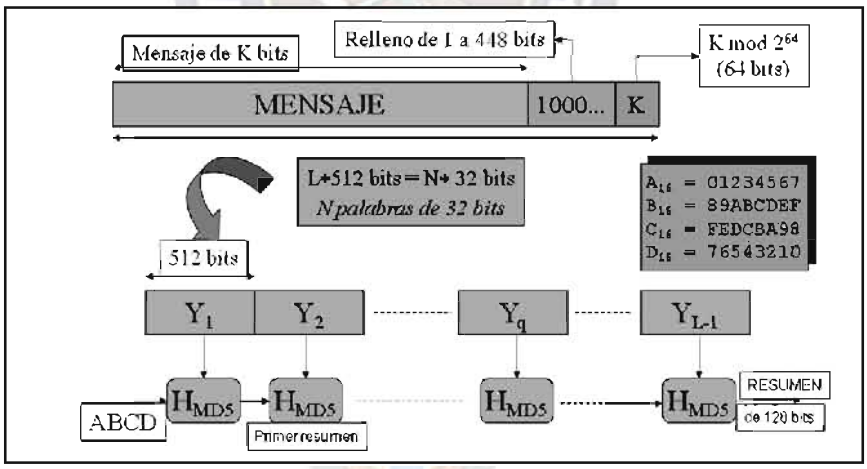


Figura 2.21. Esquema de la Función MD5
Fuente: [J Ramió 2006]

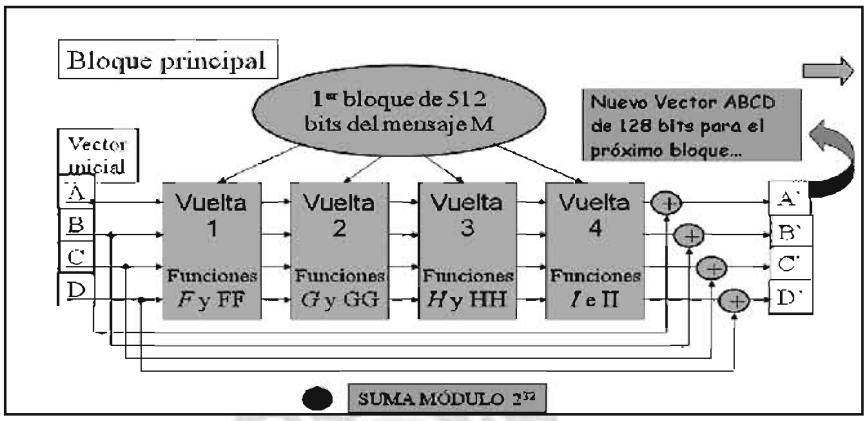


Figura 2.22. Bloque Principal del MD5
Fuente: [J Ramió 2006]

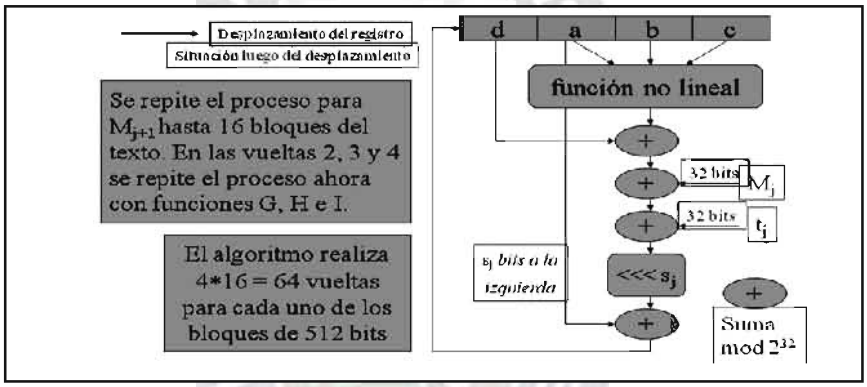


Figura 2.23. Algoritmo de las Funciones en MD5
Fuente: [J Ramió 2006]

Funciones Hash para la Autenticación.-

- ✓ Las funciones hash vistas (MD5, SHA-1, etc.) pueden usarse además para autenticar a dos usuarios.
- ✓ Como carecen de una clave privada no pueden usarse de forma directa para estos propósitos. No obstante, existen algoritmos que permiten añadirles esta función.
- ✓ Entre ellos está HMAC, una función que usando los hash vistos y una clave secreta, autentica a dos usuarios mediante sistemas de clave secreta.
- ✓ HMAC se usa en plataformas IP seguras como por ejemplo en Secure Socket Layer, SSL.

2.6.2. SEGURIDAD EN SQL SERVER

Tipos de Autenticación.-

- ✓ Windows (integrada o de confianza)
  - SQL Server delega en Windows para realizar la autenticación.
  - Sujetas a las directivas de Windows
- ✓ SQL Server (o estándar)
  - Nombre del Usuario y contraseña (hash) guardad en SQL Server. Requeridos en cada establecimiento de conexión.

Autorización: Gestión de Permisos.-

- ✓ Cuenta de inicio de sesión requeridos para establecer conexión.
- ✓ La autorización puede establecerse por medio de:
  - Roles de Servidor: Definidos a nivel Servidor independientes de las Bases de Datos.
  - Roles de Base de Datos: Cada Rol agrupa un conjunto de permisos. Un usuario puede pertenecer a uno o más roles.
  - Permiso de acceso de base de datos
  - Permiso de ejecución de sentencias: A un usuario, rol definido por el usuario, y a public, puede concederse o denegarse permiso para ejecutar sentencias.
  - Permisos específicos para objetos de base de datos: A un usuario, rol definido por el usuario, y a public, se les puede conceder o denegar permiso sobre objetos.

Estrategias de Seguridad.-

- ✓ Dar permiso a vistas y funciones en línea en lugar de a las propias tablas
  - Ocultan la complejidad de la base de datos

- Permiten fácilmente gestionar el acceso a nivel de columna

✓ Uso de Procedimientos Almacenados

- Los usuario no necesitan tener permisos para acceder alas tablas, solo permisos de ejecución de los procedimientos almacenados

✓ Cuando se diseña un sistema se tiene básicamente dos opciones para controlar la seguridad:

- Basarse en la seguridad de SQL Server. Ventajas: facilidad de auditoría. Inconvenientes: los usuarios pueden acceder a la base de datos por otros medios
- La aplicación cliente controla la seguridad. Ventajas: los usuarios solo pueden acceder a la base de datos por medio de la aplicación. Inconvenientes: dificultad de auditoría.

2.7. EVALUACIÓN / PRUEBAS DEL SOFTWARE

Encontrar defectos en el software, considerando que la prueba tuvo éxito si se encontró al menos un defecto, podemos definir prueba fracasada si existen defectos que no se detectaron.

Concretamente la Prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (configuración de la prueba), registrándose los resultados obtenidos. Seguidamente se realiza un proceso de Evaluación Dinámica en el que los resultados obtenidos se comparan con los resultados esperados para localizar fallos en el software. Estos fallos conducen a un proceso de Depuración en el que es necesario identificar la falta asociada con cada fallo y corregirla, pudiendo dar lugar a una nueva prueba. Como resultado final se puede obtener una determinada Predicción de Fiabilidad, tal como se indicó anteriormente, o un cierto nivel de confianza en el software probado.

El objetivo de las pruebas no es asegurar la ausencia de defectos en un software, únicamente puede demostrar que existen defectos en el software. Nuestro objetivo es pues, diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Para ser más eficaces (es decir, con más alta probabilidad de encontrar errores), las pruebas deberían ser realizadas por un equipo independiente al que realizó el software. El ingeniero de software que creó el sistema no es el más adecuado para llevar a cabo las pruebas de dicho software, ya que inconscientemente tratará de demostrar que el software funciona, y no que no lo hace, por lo que la prueba puede tener menos éxito.

Una prueba de software, comparando los resultados obtenidos con los esperados. A continuación se presentan algunas características de una buena prueba:

- Una buena prueba ha de tener una alta probabilidad de encontrar un fallo. Para alcanzar este objetivo el responsable de la prueba debe entender el software e intentar desarrollar una *imagen mental* de cómo podría fallar.
- Una buena prueba debe centrarse en dos objetivos: 1) *probar si el software no hace lo que debe hacer*, y 2) *probar si el software hace lo que no debe hacer*.
- Una buena prueba no debe ser redundante. El tiempo y los recursos son limitados, así que *todas las pruebas deberían tener un propósito diferente*.
- Una buena prueba debería ser la "mejor de la cosecha". Esto es, se debería emplear la prueba que tenga la *más alta probabilidad de descubrir una clase entera de errores*.
- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja, pero si se quieren combinar varias pruebas a la vez se pueden enmascarar errores, por lo que en general, *cada prueba debería realizarse separadamente*.

Veamos ahora cuáles son las tareas a realizar para probar un software:

1. ***Diseño de las pruebas.*** Esto es, identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejercitan diferentes criterios como guía para realizar las pruebas. Seguidamente veremos algunas de estas técnicas.

2. ***Generación de los casos de prueba.*** Los casos de prueba representan los datos que se utilizarán como entrada para ejecutar el software a probar. Más concretamente los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Como veremos posteriormente, cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba. Por lo tanto, durante la tarea de generación de casos de prueba, se han de confeccionar los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso (como se verá más adelante, esto es necesario para detectar un posible fallo en el programa).

3. ***Definición de los procedimientos de la prueba.*** Esto es, especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar, cuándo, etc.

4. ***Ejecución de la prueba,*** aplicando los casos de prueba generados previamente e identificando los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

5. ***Realización de un informe de la prueba,*** con el resultado de la ejecución de las pruebas, qué casos de prueba pasaron satisfactoriamente, cuáles no, y qué fallos se detectaron. Tras estas tareas es necesario realizar un proceso de depuración de las faltas asociadas a los fallos identificados. Nosotros nos centraremos en el segundo paso, explicando cómo distintas técnicas de pruebas pueden proporcionar criterios para generar distintos datos de prueba.

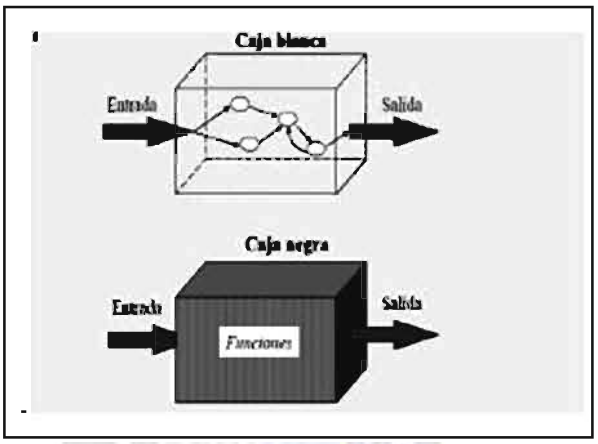


Figura 2.24. Representación de pruebas de Caja Blanca y Caja Negra
Fuente: [N Juristo 2006]

2.7.1. PRUEBAS DE CAJA BLANCA O ESTRUCTURALES

Este método se centra en cómo diseñar los casos de prueba atendiendo al ***comportamiento interno y la estructura del programa.*** Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa. Gráficamente se representa en la figura 2.24. ambas prueba caja blanca y caja negra.

Como se ha indicado ya, puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa. Por ello se han definido distintos criterios de cobertura lógica, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Estos criterios son: ***Cobertura de Sentencias*****:** Se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez. ***Cobertura de Decisión*****:** Se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con el falso. ***Cobertura de Condiciones*****:** Se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra falso. ***Cobertura Decisión/Condición*****:** Se escriben casos de prueba suficientes para que cada condición en una decisión tome

todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez. ***Cobertura de Condición Múltiple***: Se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen al menos una vez. ***Cobertura de Caminos*:** Se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa. Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida. Este último criterio es el que se va a estudiar.

Cobertura de Caminos.- La aplicación de este criterio de cobertura asegura que los casos de prueba diseñados permiten que todas las sentencias del programa sean ejecutadas al menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Una de las técnicas empleadas para aplicar este criterio de cobertura es la **Prueba del Camino Básico**. Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa. Los pasos a realizar para aplicar esta técnica son:

- ✓ Representar el programa en un grafo de flujo
- ✓ Calcular la complejidad ciclomática
- ✓ Determinar el conjunto básico de caminos independientes
- ✓ Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación, se detallan cada uno de estos pasos.

Representar el programa en un grafo de flujo.- El grafo de flujo se utiliza para representar flujo de control lógico de un programa. Para ello se utilizan los tres elementos siguientes, ver figura 2.25:

- ✓ *Nodos*: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).
- ✓ *Aristas*: líneas que unen dos nodos.
- ✓ *Regiones*: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.
- ✓ *Nodos predicados*: cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

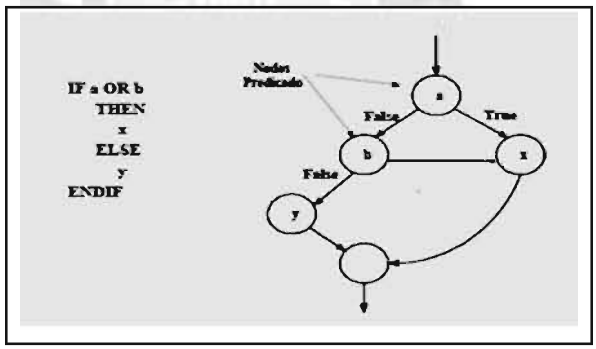


Figura 2.25 Representación de condición múltiple
Fuente: [N Juristo 2006]

Calcular la complejidad ciclomática.- a complejidad ciclomática es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba del camino básico, el valor de la complejidad ciclomática define el número de caminos independientes de dicho programa, y por lo tanto, el número de casos de prueba a realizar. Posteriormente veremos cómo se identifican esos caminos, pero primero veamos cómo se puede calcular la complejidad ciclomática a partir de un grafo de flujo, para obtener el número de caminos a identificar. Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo, ver figura 2.26:

1. El número de regiones del grafo coincide con la complejidad ciclomática, V(G).

2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como

V(G) = Aristas – Nodos + 2

3. La complejidad ciclomática, V(G), de un grafo de flujo G se define como

V(G) = Nodos Predicado + 1

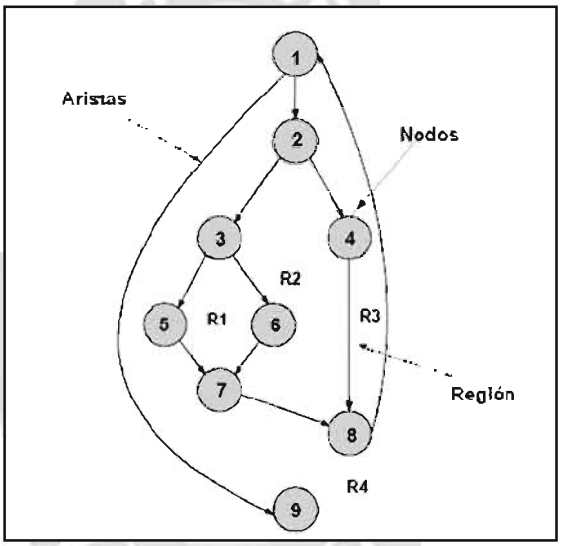


Figura 2.26. Propiedades del grafo
Fuente: [N Juristo 2006]

Esta complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa.

Determinar el conjunto básico de caminos independientes.- Un camino independiente es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una condición, respecto a los caminos existentes. En términos del diagrama de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino. En la

identificación de los distintos caminos de un programa para probar se debe tener en cuenta que cada nuevo camino debe tener el mínimo número de sentencias nuevas o condiciones nuevas respecto a los que ya existen. De esta manera se intenta que el proceso de depuración sea más sencillo.

El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación, se muestran algunas heurísticas para identificar dichos caminos:

- ✓ Elegir un *camino principal* que represente una función válida que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.
- ✓ Identificar el segundo camino mediante la localización de la *primera decisión* en el camino de la línea básica alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
- ✓ Identificar un tercer camino, colocando la primera decisión en su valor original a la vez que se altera la *segunda decisión* del camino básico, mientras se intenta mantener el resto de decisiones originales.
- ✓ Continuar el proceso hasta haber conseguido *tratar todas las decisiones*, intentando mantener como en su origen el resto de ellas.

Este método permite obtener V(G) caminos independientes cubriendo el criterio de cobertura de decisión y sentencia.

Derivar los casos de prueba que fuerzan la ejecución de cada camino.- El último paso es construir los casos de prueba que fuerzan la ejecución de cada camino. Una forma de representar el conjunto de casos de prueba es mencionando; Nro. De caminos, casos de prueba, resultado esperado.

2.8. EVALUACION / CALIDAD DEL SOFTWARE

2.8.1. ISO 9126/IEC 9126 es un estándar internacional para la evaluación del software. Este estándar proviene desde el modelo establecido en 1977 por McCall y sus colegas, los cuales propusieron un modelo para especificar la calidad del software

El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso. Solo la parte primera, modelo de calidad, es un estándar aprobado y publicado siendo el resto de partes de la norma informes que se encuentran en la fase denominada Technical Report (TR)

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas de la siguiente manera, ver figura 2.27:

- **Funcionalidad** - Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen lo indicado o implica necesidades. Idoneidad, Exactitud, Interoperabilidad, Seguridad, Cumplimiento de normas.
- **Fiabilidad** - Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período de tiempo establecido. madurez, recuperabilidad, tolerancia a fallos.
  Es especificado al momento en que comienza a funcionar, determinado por $t_0=0$. A partir de este tiempo se observa el trabajo del sistema hasta que se produzca una falla en el instante T que se aproxima a una variable aleatoria continua, que determina la fiabilidad en términos probabilístico. Entonces se tiene las siguientes probabilidades:

$P(T <= t) = F(t)$ …(1) Probabilidad de fallas

$P(T > t) = 1 - F(t)$ …(2) Probabilidad de trabajo sin fallas

Debido a que se tiene tiempos de inicio y fin, para el cálculo de estas probabilidades se utiliza la distribución exponencial. Entonces,

$$F(t) = PuntoFuncion\ e^{-\square * t}$$ Ecuación de Confiabilidad

- **Usabilidad** - Un conjuntos de atributos relacionados con el esfuerzo necesitado para el uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios. Aprendizaje, Comprensión, Operatividad
- **Eficiencia** - Conjunto de atributos relacionados entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas. Comportamiento en el tiempo, Comportamiento de recursos
- **Mantenimiento** - Conjunto de atributos relacionados con el esfuerzo necesitado para modificar las especificaciones. Estabilidad, Facilidad de análisis, Facilidad de cambio, Facilidad de pruebas

  Para calcular la estabilidad del sistema, es decir índice de madurez del software (IMS), se debe considerar los cambios que ocurrieron con cada versión del producto. El IMS se calcula de la siguiente manera:

  $$IMS = [\ MT - (\ Fc + Fa + Fe\ )\ ] / MT$$

  donde:

  MT: Número de módulos en la versión actual

  Fc: Número de módulos en la versión actual que se cambiaron

  Fa: Número de módulos en la versión actual que se adicionaron.

  Fe: Número de módulos en la versión actual que se eliminaron.

- **Disponibilidad**. La disponibilidad del software es la probabilidad de que un programa funcione de acuerdo con los requisitos en un momento dado, y se define como.

  $$Disponibilidad = [TMDF / (TMDF + TMDR)] \times 100\%$$

  Donde:

  TMDF: Tiempo medio de fallo

TMDR: Tiempo medio de reparación

- **Movilidad** - Conjunto de atributos relacionados con la habilidad del software para ser transferido desde un entorno a otro. Capacidad de instalación, capacidad de reemplazamiento, y adaptabilidad.

  A través de la métrica de facilidad de instalación se calcula el factor de portabilidad. La facilidad de instalación esta dada por la siguiente relación:

  X = A / B

  Donde:

  A: Número de casos de éxitos de la operación de instalación por parte del usuario.

  B: Número total de operaciones de instalación que realizo el usuario.

Calidad del sw. (Interna y externa)					
Funcionalidad	**Fiabilidad**	**Facilidad de uso**	**Eficiencia**	**Mantenimiento**	**Movilidad**
Idoneidad	Madurez	Fácil comprensión	Comportamiento frente al tiempo	Facilidad de análisis	Adaptabilidad
Exactitud	Tolerancia a fallos	Fácil aprendizaje	Uso de recursos	Capacidad para cambios	Facilidad de instalación
Interoperatividad	Capacidad de recuperación	Operatividad	Adherencia a normas	Estabilidad	Coexistencia
Seguridad	Adherencia a normas	Software atractivo		Facilidad para pruebas	Facilidad de reemplazo
Adherencia a normas		Adherencia a normas		Adherencia a normas	Adherencia a normas

Figura 2.27. Modelo de calidad según ISO9126
Fuente: [Z Carranza 2001]

2.8.2. METRICA PARA EL TAMAÑO: PUNTO FUNCION

Esta métrica se define como una métrica funcional, dado que se enfoca a la funcionalidad que el software proporciona al usuario. *"Es una métrica para establecer el tamaño y complejidad de los sistemas informáticos basada en la cantidad de funcionalidad requerida y entregada a los usuarios"*, o *"Los Puntos Función miden el tamaño lógico o funcional de los proyectos o aplicaciones de software basados en los requerimientos funcionales del usuario*" [22].

Se debe determinar los siguientes puntos:

- ✓ **Número de entradas de usuario**, entradas de usuario que proporciona al software diferentes datos orientados a la aplicación.
- ✓ **Número de salidas de usuario**, salidas que proporcionan al usuario información orientada a la aplicación. Como ser informes, mensajes de error, etc.
- ✓ **Número de peticiones de usuario**, una petición esta definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva.
- ✓ **Número de archivos,** se cuenta cada archivo maestro lógico.

- ✓ **Número de interfaces externas**. Se cuenta todas las interfaces legibles por el ordenador que son solicitados para transmitir información a otro sistema.

La relación que permite calcular los puntos de función es la siguiente:

PF = CuentaTotal * (Grado de Confiabilidad + tasa de error * Σ Fi)

Donde:

PF: Medida de funcionalidad.

Cuenta Total: es la suma del valor de las entradas, salidas, peticiones, interfaces externas y archivos.

Grado de confiabilidad: Es la confiabilidad estimada del sistema.

Tasa de error: Probabilidad subjetiva estimada del dominio de la información.

Fi: son valores de ajuste de complejidad.

CAPITULO III
MARCO APLICATIVO

En este capítulo se desarrolla la ingeniería de requerimientos y las fases del ciclo de desarrollo del sistema de información correspondientes a la metodología Feature Driven Development (FDD), además del uso del Lenguaje de Modelo Unificado (UML) para el modela de los datos, serán utilizados algunos diagramas que la metodología así lo requiera.

3.1. FASE 1: INGENIERIA DE REQUERIMIENTOS

Los requerimientos básicamente fueron recopilados de los funcionarios que trabajan en la Unidad Desarrollo Organizacional (RRHH) y la Unidad de Sistemas del SENASIR. Se realizaron las siguientes actividades:

- ✓ Cuestionarios, con un formato sencillo que contiene preguntas abiertas y cerradas [23]. Un ejemplo de estos cuestionarios puede observarse en el Anexo E.
- ✓ Entrevistas, se programo realizar entrevistas con los jefes de unidad, y algunos funcionarios cuyo trabajo está y/o estará relacionado con funcionalidades del sistema. También se visito algunas otras dependencias de la institución.
- ✓ Se recolecto documentos, formatos, información en medios magnéticos, reglamentos, etc. Necesarios para el desarrollo del sistema

3.1.1. REQUISITOS NO FUNCIONALES

Los aspectos del sistema visibles por el usuario que no se relacionan en forma directa con el comportamiento funcional del sistema están clasificados de la siguiente manera.

Requerimientos del Producto
Usabilidad: Que se de fácil uso **Eficiencia:** Que el sistema tenga un rendimientos óptimo, es decir realice lo que debe hacer utilizando pocos recursos necesarios **Disponibilidad:** Que se pueda utilizar en el momentos requerido. **Portabilidad:** que tenga facilidad de ser transferido de un entorno a otro.
Requerimientos Organizacionales **Entrega** **Implementación** **Estándares**
Requerimientos Externos
Éticos: Todo proceso deberá estar regido bajo los reglamentos y normas correspondientes a la institución. **Legislativos:** Deberá existir seguridad en el acceso a la información gestionada por el sistema.

Tabla 3.1 Requisitos No Funcionales
Fuente: [Elaboración Propia]

3.1.2. REQUISITOS FUNCIONALES

La Unidad de Desarrollo Organizacional, responsable de la correcta aplicación de las Normas Básicas del Sistema de Administración Personal (NB SAP) y del Reglamento Interno del Personal (RIP) del SENASIR, requiere de un sistema de información que contenga las siguientes características. Los requisitos funcionales que describen las interacciones entre el sistema y el entorno son mencionados a continuación.

- ✓ Se requiere el registro completo de los datos de los funcionarios que trabajan en las diferentes dependencias de la institución a nivel nacional.
- ✓ Seguimiento adecuado de la presentación de declaraciones juradas, declaración de compatibilidad, informe de actividades, seguro medico, declaración de impuestos, CAS.
- ✓ Realizar el proceso de evaluación de desempeño; permitiendo programar las evaluaciones, contar con el registro de las evaluaciones realizadas, de los resultados obtenidos, permitiendo emitir reportes correspondientes.
- ✓ Creación y almacenamiento histórico de estructuras organizacionales, permitiendo crear nuevas unidades, edificios, puestos, cargas, ítems y escalas salariales.
- ✓ Conocer estructuras organizacionales que estuvieron vigentes en gestiones pasadas.
- ✓ Conocer la estructura organizacional a la que estuvo asociada un funcionario específico (en que unidad, cargo, ítem y con que escala salarial estuvo).
- ✓ Control de movimientos (verticales y horizontales) de procesos organizacionales como; designaciones, reasignaciones, destituciones, renuncias, nuevos contratos, adendas y rescisiones.
- ✓ Conocer puestos acéfalos en cada unidad o área de unidad correspondiente.
- ✓ Validar lineamientos legales para realizar algún movimiento de transferencia, designación o retiro.
- ✓ Conocer el recorrido laboral del funcionario a través del tiempo, desde el momento de su designación hasta su estado actual.
- ✓ Generación automática de memorándums de cada proceso organizacional. Generación automática del CITE correspondiente para cada tipo de memorándum.
- ✓ Automatización en los procesos de control de personal.
- ✓ Contar con una estructura que permita el uso de la tecnología "Control biométrico – Huella Digital" para el control de asistencia.

- ✓ Llevar el control diario de asistencia, permitiendo reportes respectivos sobre atrasos, faltas. Identificando a funcionarios con horarios especiales, tolerancias o permisos solicitados.
- ✓ En cada corte de fecha (mensual) generar automáticamente las sanciones correspondientes según el RIP- Reglamento Interno de Personal del SENASIR.
- ✓ Definición de horarios para cada funcionario y para la institución en general.
- ✓ Control y registro de licencias (sin cargo a vacaciones) y permisos (con cargo a vacaciones) solicitas por el funcionario.
- ✓ Calculo, programación de vacaciones.
- ✓ Conocer el número de días trabajados por el funcionario (años de antigüedad).
- ✓ Según el RIP programar sus vacaciones (considerando la presentación de CAS)
- ✓ Generar el cálculo de días hábiles solicitados, llevar la cuenta de su saldo de vacaciones en el periodo correspondientes.

3.1.3. CASOS DE USO

Para la especificación de los requerimientos funcionales se utiliza Casos de Uso del UML. A continuación se describe los actores, casos de uso y relaciones que contiene este modelado de datos.

Actores;

- **Responsable de Kardex:** Este usuario será el encargo del registro de datos personales; como datos familiares, declaraciones de incompatibilidad, datos académicos, datos sobre cursos de capacitación, datos de experiencia laboral, y además de las continuas actualizaciones de los Informes presentados, seguros médicos, declaraciones juradas, calificación de años de servicio, registros del personal de planta y contrato.
- **Responsable de Evaluación:** Usuario encargado del registro, proceso de evaluación del personal, también encargado de la emisión de memorándums de evaluación con resultados del proceso.

- **Responsable de Planillas:** Registro de declaraciones de impuestos, obtención de reportes de declaraciones de impuestos, además es el encargado del registro y modificaciones de las escalas salariales, puestos organizacionales, y repartición de unidades organizacionales.
- **Responsable de Movimientos:** Encargado de la elaboración y control de las designaciones, reasignaciones, retiros, renuncias, nuevos contratos, adendas, y rescisiones correspondientes a personal de planta y contrato a nivel nacional.
- **Responsable de Control:** Usuario responsable del registro y control de asistencia de los funcionarios; definición de tipos de horarios por funcionario y a nivel institución, también es responsable del control de atrasos, faltas y la generación de sanciones respectivas. Encargado del control y registro de licencias y vacaciones normadas según el Reglamento Interno del SENASIR.
- **Responsable de Aprobaciones EO Y ES:** Responsable de aprobaciones de Estructuras Organizacionales.
- **Funcionario:** Persona de palta o contrato que trabaja en la institución.
- **Responsable de RRHH:** Jefe de la Unidad de Recursos Humanos
- **Administración de Sistema:** Usuario con privilegios de modificar, adicionar y eliminar accesos de usuarios al sistema.
- **Administración de Base de Datos:** Responsable de la Administración de la Base de Datos del Sistema.

A continuación se muestra gráficamente un conjunto de casos de uso principales, agrupados por módulos y la descripción narrativa de cada caso de uso.

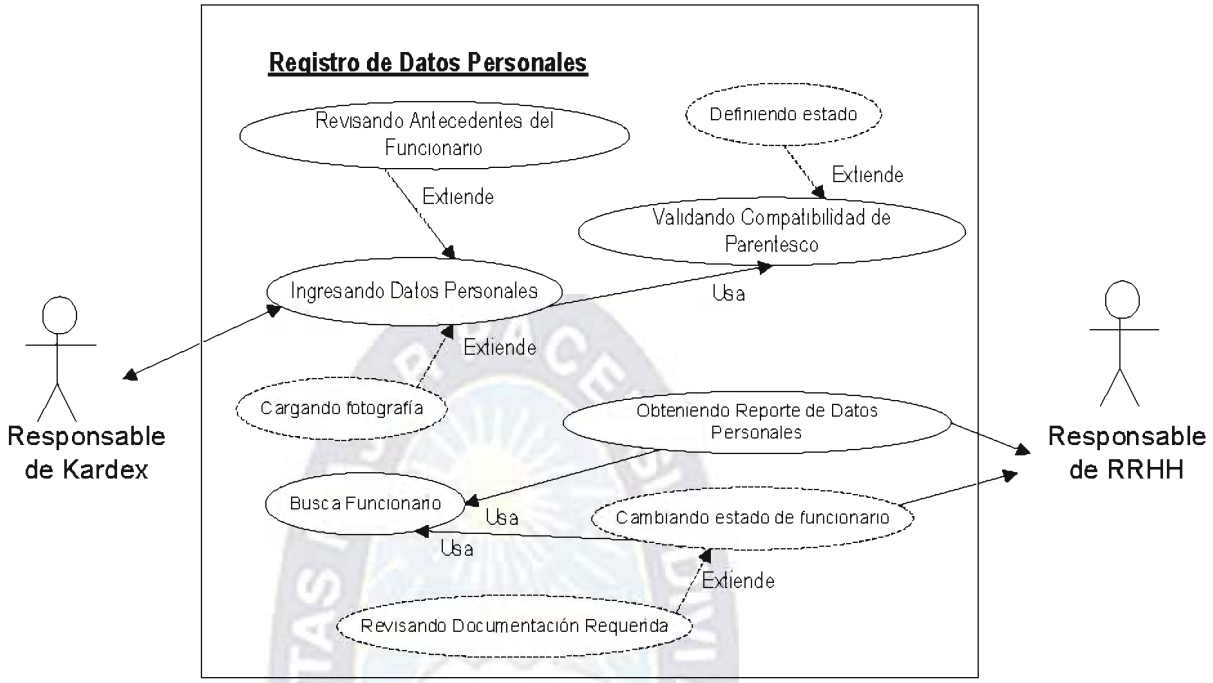


Figura 3.1. Diagrama de Casos de Uso del Módulo de Registro de Personal
Fuente: [Elaboración Propia]

Caso de Uso: Cambiando estado del funcionario	
Actor: Responsable de RRHH	
Descripción: Permite habilitar/ deshabilitar estado del funcionario, controlando que para esta tarea debe presentarse documentación requerida.	
Activación: Cuando el sistema detecta automáticamente la incompatibilidad de parentesco se cambia a un estado de inhabilitado, la habilitación de este funcionario debe ser manual.	
Curso Normal	**Curso Alternativo**
1.- El responsable de RRHH desea cambiar el estado del funcionario.	
2.- El responsable de RRHH busca funcionario	2.1. Si son presentados los documentos requeridos se cambia de estado.
Precondición: No debe existir registro en la Base de Datos y presentar documentos requeridos.	
Postcondición: Actualizar el estado del funcionario	
Observación y datos:	

Tabla 3.2. Descripción narrativa – Cambiando estado del funcionario
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Datos Personales	
Actor: Responsable de Kardex	
Descripción: Permite el registro de datos personales, datos familiares, datos académicos, cursos de capacitación, experiencia laboral, conocimiento de idiomas.	
Activación: El caso de uso se activa cuando el responsable de kardex seleccionar Adicionar nuevo Funcionario.	
Curso Normal	**Curso Alternativo**
1.- El nuevo funcionario presenta la documentación requerida.	
2.- El encargo de Kardex elabora su File Personal del nuevo funcionario.	2.1. Si el funcionario no tiene un registro anterior en la base de datos, entonces se le asigna un nuevo código de File.
3.- Se realiza el registro de los datos personales de nuevo funcionario.	3.1. El sistema valida la compatibilidad de parentesco (comprueba si existe el registro del familiar en la BD)
	3.2. Si existirá incompatibilidad el sistema cambia el estado del funcionario a estado Inhabilitado.
Precondición: No debe existir registro en la Base de Datos	
Postcondición: No se permite duplicidad de CI, ni valor null	
Observación y datos:	

Tabla 3.3. Descripción narrativa – Ingresando Datos Personales
Fuente: [Elaboración Propia]

Caso de Uso: Obteniendo reporte de datos personales	
Actor: Responsable de RRHH	
Descripción: Obtener Reportes del registro de datos personales	
Activación: Después del registro correspondiente el usuario puede ver los reportes que necesita.	
Curso Normal	**Curso Alternativo**
1.- El responsable de RRHH busca funcionario.	
2.- Verifica estado valido para visualizar los reportes.	
3.- El responsable de RRHH elige tipo de reporte.	
Precondición: El funcionario debe existir en la base de datos	
Postcondición: Visualizar el reporte solicitado.	
Observación y datos:	

Tabla 3.4. Descripción narrativa – Obteniendo reporte de datos personales
Fuente: [Elaboración Propia]

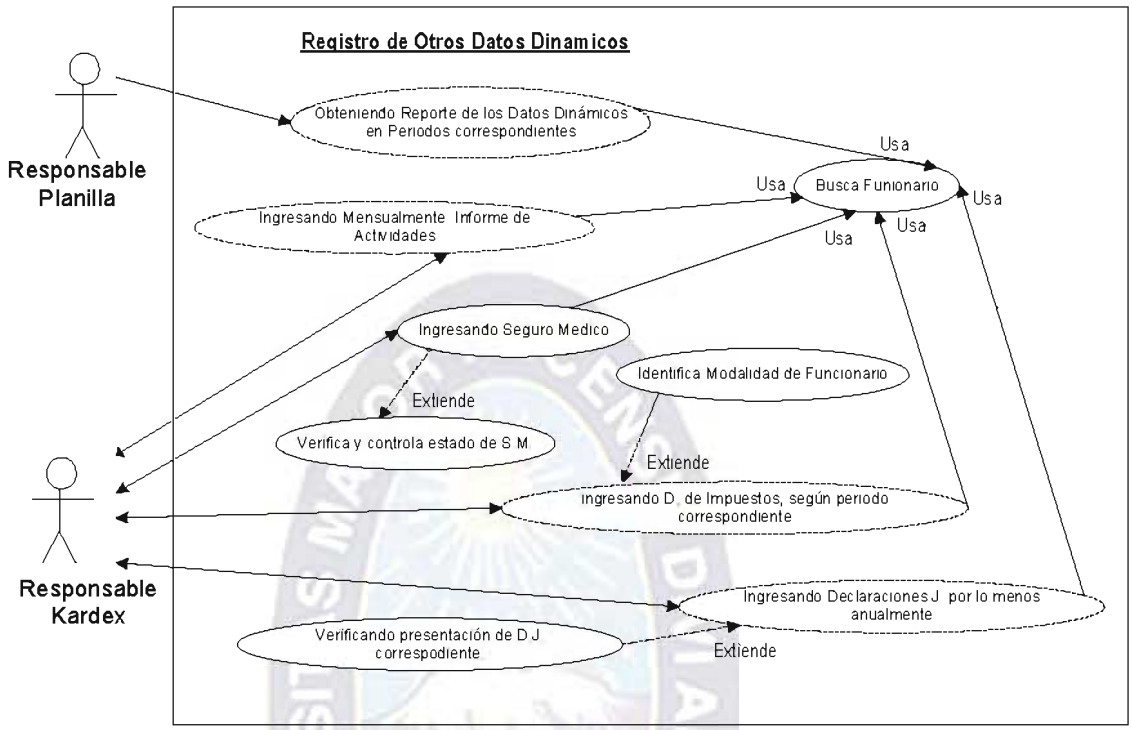


Figura 3.2. Diagrama de Casos de Uso del Módulo de Registro de Personal – Registro de datos dinámicos
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Seguro Medico	
Actor: Responsable de Kardex	
Descripción: Registrar y controlar las altas y bajas de los seguros médicos.	
Activación: El caso de uso se activa cuando el responsable de kardex selecciona Adicionar Seguro Medico.	
Curso Normal	**Curso Alternativo**
1.- El encargado de registro ingresa registro de seguro medico.	
2.- El sistema busca funcionario.	
3.- El sistema verifica estado de seguro medico.	3.1. Si se desea dar de baja, solo se debe actualizar la "fecha de baja"
	3.2. Si se desea dar de "alta " se llena todos los datos solicitado, excepto la "fecha de baja" .
Precondición: Debe existir el registro de la institución a la que se desea asegurar al funcionario.	

Tabla 3.5. Descripción narrativa – Ingresando Seguro Medico
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Mensualmente informe de actividades	
Actor: Responsable de Kardex	
Descripción: Registro y control de informes presentados por el personal de contrato y/o planta.	
Activación: El caso de uso se activa cuando el responsable de kardex selecciona Adicionar informe de actividades.	
Curso Normal	**Curso Alternativo**
1.- El encargado de registro ingresa la información de Informe de Actividades	
2.- El sistema busca funcionario.	2.1. Si no existe el registro de datos personales del funcionario se le informa que debe primero registrar al nuevo funcionario.
3.- El sistema valida las fechas ingresadas y el tipo de informe presentado	
Precondición: El funcionario debe existir en la base de datos.	
Postcondición: Registro de datos del funcionario	
Observación y datos:	

Tabla 3.6. Descripción narrativa – Ingresando Mensualmente informe de actividades
Fuente: [Elaboración Propia]

Caso de Uso: Obteniendo reporte de datos dinámicos	
Actor: Responsable de Planilla	
Descripción: Facilitar al responsable de planilla la información que requiere respecto a los datos dinámicos registrados.	
Activación: El caso de uso se activa cuando el Responsable de Planilla selecciona generar reportes.	
Evento Actor	**Evento Sistema**
1.- El encargado de planilla datos de referencia para localizar al funcionario.	
2.- El sistema busca funcionario bajo los parámetros introducidos.	
3.- El encargo de planilla elige el tipo de reporte.	
Precondición: Se debe tener el registro completo de los datos del funcionario	
Postcondición: Visualizar el reporte solicitado.	
Observación y datos:	

Tabla 3.7. Descripción narrativa – Obteniendo reporte de datos dinámicos
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Declaración de Impuestos	
Actor: Responsable de Kardex	
Descripción: Registro y control declaración de impuestos.	
Activación: El caso de uso se activa cuando el responsable de kardex selecciona Adicionar Declaración Impuestos.	
Curso Normal	**Curso Alternativo**
1.- El encargado de registro datos de referencia para localizar al funcionario.	
2.- El sistema busca funcionario bajo los parámetros introducidos.	
3.- El encargado de registro ingresa los datos de declaración de impuestos del funcionario.	
4.- El sistema valida periodo de presentación de declaraciones según la modalidad del funcionario.	4.1. Si es Funcionario de Contrato: mensualmente.
	4.2. Si es Funcionario de Planta: trimestralmente
Precondición: Se debe conocer la modalidad (contrato o planta) del funcionario.	
Postcondición: Adicionar los datos de declaración de impuestos	
Observación y datos:	

Tabla 3.8. Descripción narrativa – Ingresando Declaración de Impuestos
Fuente: [Elaboración Propia]

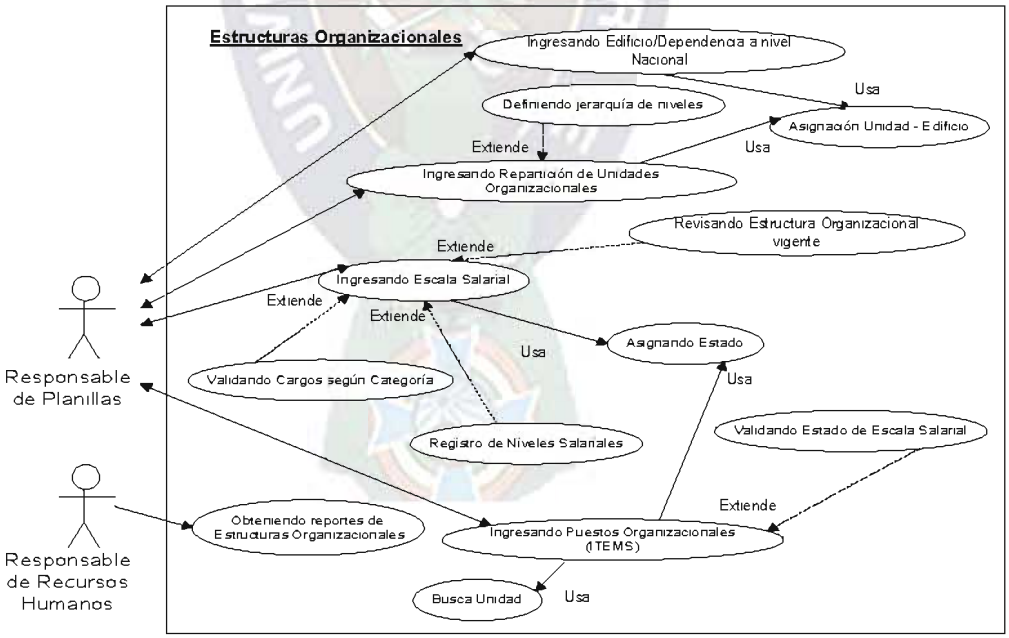


Figura 3.3. Diagrama de Casos de Uso del Módulo de Procesos Organizacionales – Estructuras Organizacionales
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando unidades organizacionales	
Actor: Responsable de Planilla	
Descripción: Crear nuevas unidades organizacionales	
Activación: El caso de uso se activa cuando el Responsable de Planilla selecciona Adicionar edificio o unidad.	
Curso Normal	**Curso Alternativo**
1.- El encargado de planilla ingresa datos de la nueva unidad	
2.- El sistema registra datos y define la jerarquía de niveles.	
3.- También se puede hacer la asignación Edificio / Unidad.	
Precondición: Debe existir el registro de edificios.	
Postcondición: Adicionar nueva unidad organizacional	
Observación y datos:	

Tabla 3.9. Descripción narrativa – Ingresando Unidades Organizacionales
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Puestos Organizacionales (ítems)	
Actor: Responsable de Planilla	
Descripción: Crear nuevas escalas salariales y guardando un histórico de escalas pasadas	
Activación: El caso de uso se activa cuando el Responsable de Planilla selecciona Adicionar edificio o unidad.	
Curso Normal	**Curso Alternativo**
1.- El encargado de planilla registra los niveles salariales con sus respectivos salarios	
2.- El encargado de planilla revisa la anterior escala salarial vigente.	2.1. Si la nueva escala tiene registros similares al anterior entonces se realiza una copia.
3.- El sistema valida los cargos según la categoría a la que corresponda.	3.1. Si se desea registrar un nuevo cargo o nivel del cargo, debe estar asociado correctamente a la categoría a la que corresponde.
4.- El sistema le asigna estado de "edición".	4.1. Si se desea trabajar con esta escala salarial para la creación de ítems se le cambiara a estado de "ejecución"
Precondición: Se debe tener el registro de escala salarial en estado de "ejecución"	
Postcondición: Conformar estructuras organizacionales (crear ítems.)	
Observaciones y datos:	

Tabla 3.10. Descripción narrativa – Ingresando Puestos Organizacionales (ítems)
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando escalas salariales	
Actor: Responsable de Planilla	
Descripción: Crear nuevas ítems y guardando un histórico de las otras escalas salariales que estuvieron vigentes.	
Activación: El caso de uso se activa cuando el Responsable de Planilla selecciona Adicionar estructura organizacional.	
Curso Normal	**Curso Alternativo**
1.- El encargado de planilla selecciona una unidad en la que desea crearle ítems.	1.1. Se debe seleccionar específicamente la unidad o área de la unidad para la creación del ítem.
2.- El sistema carga la escala salarial que este en estado de ejecución.	2.1. Si solo existiera una escala salarial en estado de "edición", se le informa que necesita una escala salarial en estado de "ejecución".
3.- El responsable de planilla ingresa los datos solicitados para su registro.	3.1. Si se registrara un ítem duplicado el sistema validará este error, informando al usuario.
4.- El sistema crea estos ítems en estado "acéfalo" y la conformación de esta estructura en estado de "ejecución"	4.- Solo se pueden modificar registros de la estructura organizacional cuando estén en estado de "ejecución"
Precondición: Se debe tener el registro de Unidades (con su dependencia de edificio respectiva) y solo es permitido tener una sola escala salarial en estado "edición".	
Postcondición: Adicionar una nueva escala salarial.	
Observación y datos:	

Tabla 3.11. Descripción narrativa – Ingresando Escalas Salariales
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando edificio / dependencia a nivel nacional	
Actor: Responsable de Planilla	
Descripción: Adición de nuevo edificio.	
Activación: El caso de uso se activa cuando el Responsable de Planilla selecciona Adicionar edificio.	
Curso Normal	**Curso Alternativo**
1.- El encargado de planilla ingresa datos del nuevo edificio.	1.1. Si ya existe el registro del edificio solo se actualiza el registro.
2.- El sistema realiza la asignación de unidades al edificio registrado.	
2.- El sistema lista todas las unidades que ya estén asociadas a ese edificio.	
3.- El sistema en la contraparte lista todas las unidades disponibles para una asignación de edificio.	3.1. Si se requiere asignar a una unidad doble pertenecía (una unidad funciona en mas de un edificio), entonces se cambia a un estado que permita esta operación.
4.- El encargo de planilla selecciona una unidad organizacional del listado presentado.	
5.- Se guarda la asignación realizada.	
Precondición: Debe existir el registro de las unidades organizacionales.	
Postcondición: Adicionar nuevo edificio y asignación de unidades.	
Observación y datos:	

Tabla 3.12. Descripción narrativa – Ingresando edificio / dependencia a nivel nacional
Fuente: [Elaboración Propia]

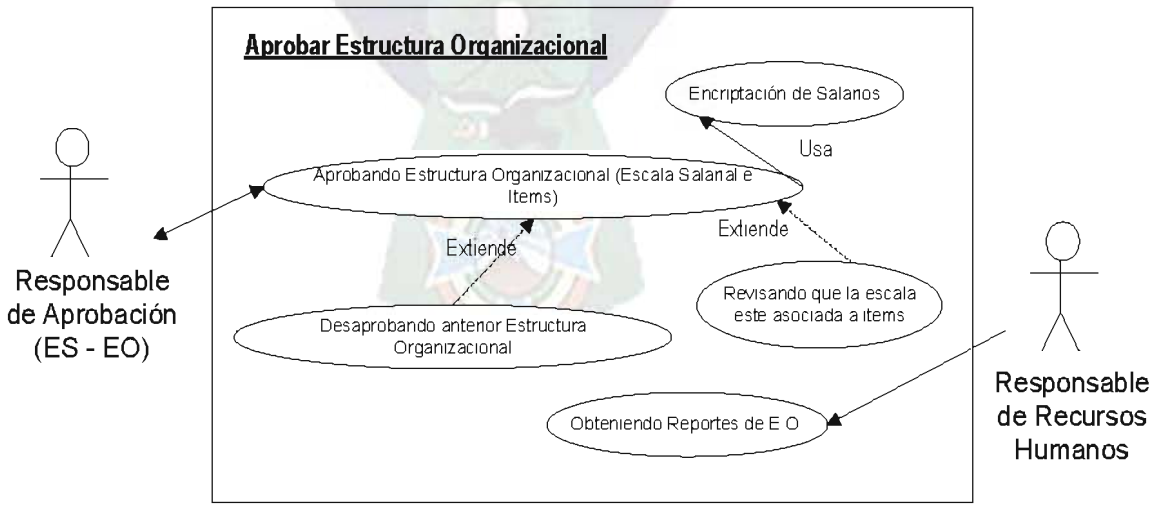


Figura 3.4. Diagrama de Casos de Uso del Módulo de Procesos Organizacionales – Aprobar E.O.
Fuente: [Elaboración Propia]

Caso de Uso: Aprobando estructura organizacional	
Actor: Responsable de Aprobación (ES - EO)	
Descripción: Aprobar escalas salariales y estructuras organizacionales.	
Activación: Se activa cuando el Responsable de Planilla selecciona Aprobar EO.	
Curso Normal	**Curso Alternativo**
1.- El encargado de aprobación, seleccionar la escala salarial que desea aprobar	1.1. Si desea modificar algún registro puede hacerlo solo en este estado (ejecución)
2.- El Sistema realiza la encriptación de los niveles salariales	
3.- El sistema cambia a estado "desaprobado" la estructura organizacional vigente.	
4.- El sistema cambia a estado de "aprobación" la nueva estructura organizacional.	
Precondición: Se debe tener editado una escala salarial y conformación de ítem s en estado "ejecución"	
Postcondición: Aprobar estructuras organizacionales (crear ítems.)	
Observaciones y datos:	

Tabla 3.13. Descripción narrativa – Aprobando Estructuras Organizacionales
Fuente: [Elaboración Propia]

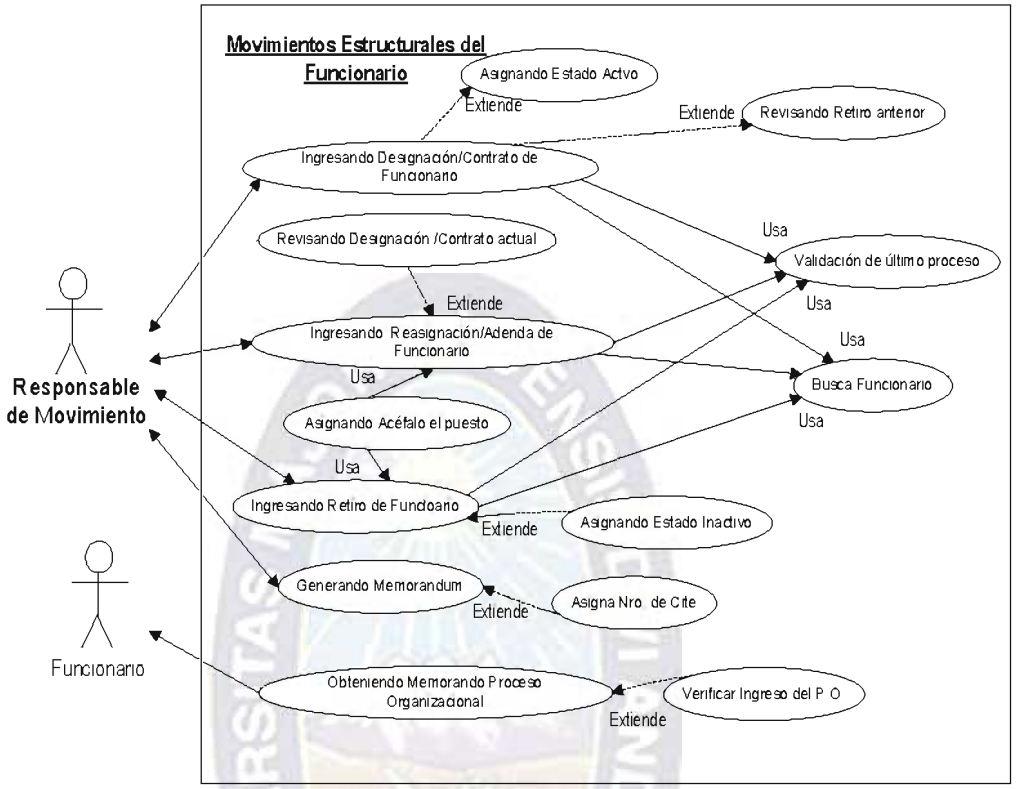


Figura 3.5. Diagrama de Casos de Uso del Módulo de P.O. – Movimientos estructurales del funcionario
Fuente: [Elaboración Propia]

Caso de Uso: Generando Memorándums	
Actor: Responsable de Movimiento de funcionario.	
Descripción: Registrar y controlar el ingreso de un nuevo funcionario.	
Activación: El caso de uso se activa cuando el Responsable de Movimiento selecciona Designación (para personal de planta) o Contrato (para personal a contrato).	
Curso Normal	**C. Alternativo**
1.- El encargado de aprobación busca funcionario.	
2.- El sistema valida que el funcionario tenga un retiro (destitución o renuncia para personal de planta – rescisión para personal a contrato) registrado, en su último proceso realizado.	
3.- El sistema cambia el estado del funcionario a un estado "activo" y al número de ítem asignado cambia a estado "ocupado"	
Precondición: Se debe tener registrado el retiro del funcionario.	
Postcondición: Ingreso de nuevos funcionarios.	
Observaciones y datos:	

Tabla 3.14. Descripción narrativa – Generando Memorándums
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Designación / Contrato de Funcionario	
Actor: Responsable de Movimiento de funcionario.	
Descripción: Generar memorándums para cada uno de los procesos.	
Activación: El caso de uso se activa cuando el Responsable de Movimiento selecciona imprimir una designación, reasignación, retiro, contrato, adenda, rescisión.	
Curso Normal	**Curso Alternativo**
1.- El encargado de aprobación guarda los datos de algún proceso organizacional (designación, reasignación, retiro, contrato, adenda, rescisión)	
2.- El sistema genera el correspondiente CITE del memorándum, dependiendo del tipo de memorándums	2.1. En caso de que se trate de una renuncia o un nuevo contrato se autogenera un correlativo.
3.-El memorándum generado puede ser modificado y también se lo puede mandar a imprimir.	
Precondición: Se debe guardar el memorándum en un directorio elegido por el usuario antes de visualizarlo y/o imprimir.	
Postcondición: Generación del memorándums en documento Word con opción de impresión del memorándum	
Observaciones y datos:	

Tabla 3.15. Descripción narrativa – Ingresando designación
Fuente: [Elaboración Propia]

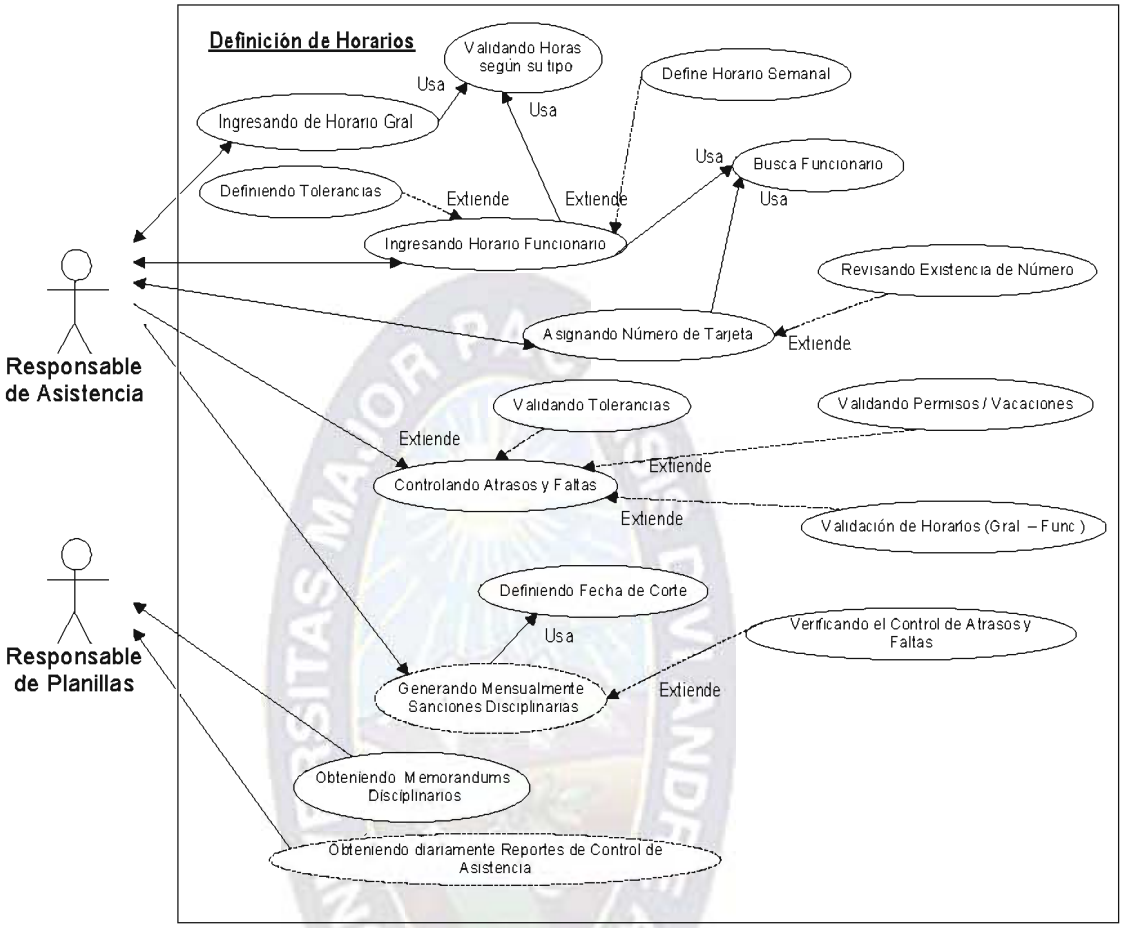


Figura 3.6. Diagrama de Casos de Uso del Módulo de Control de Personal - Definición de horarios
Fuente: [Elaboración Propia]

Caso de Uso: Ingresando Horario Funcionario

Actor: Responsable de Asistencia	
Descripción: Registrar y controlar la definición de horarios de los funcionario.	
Activación: El caso de uso se activa cuando el Responsable de Asistencia selecciona definir horario de funcionario.	
Curso Normal	**Curso Alternativo**
1.- El responsable de asistencia busca funcionario.	
2.- Ingresa datos de horario especial del funcionario. Se ingresa el detalle por días del horario definido	2.1. Si el funcionario ya tiene un horario especial definido. Se le informa al usuario para primero dar de baja esta horario.
3.- Se ingresa la toleración de ingreso, según el horario definido.	
4.- El sistema valida el horario definido según el reglamento interno de la institución.	
Precondición: Se debe tener definido los tipos de horarios.	
Postcondición: definición de horario especial para funcionarios.	
Observaciones y datos:	

Tabla 3.16. Descripción narrativa – Ingresando horario funcionario
Fuente: [Elaboración Propia]

Caso de Uso: Generando Mensualmente sanciones disciplinarias.	
Actor: Responsable de Asistencia	
Descripción: Generación automática de las sanciones disciplinarias	
Activación: El caso de uso se activa cuando el Responsable de Asistencia selecciona Generar sanciones disciplinarias.	
Curso Normal	**Curso Alternativo**
1.- El responsable selecciona el mes correspondiente.	
2.- El sistema valida los datos según la definición de corte de fecha.	2.1. Si no se tiene la definición de corte de fecha, se le informa al usuario. Para que ingrese el dato requerido.
3.- El sistema genera automáticamente las sanciones establecidas por el reglamento interno de la institución según los registros de los minutos de atrasos y/o faltas.	
Precondición: Se debe tener definido el corte de fecha y tener el registro correspondiente de los minutos de atrasos y/o faltas.	
Postcondición: Generar Sanciones disciplinarias.	
Observaciones y datos:	

Tabla 3.17. Descripción narrativa – Generando mensualmente sanciones disciplinarias
Fuente:[Elaboración Propia]

Caso de Uso: Controlando atrasos y faltas	
Actor: Responsable de Asistencia	
Descripción: Control y generación de los atrasos y faltas.	
Activación: El caso de uso se activa cuando el Responsable de Asistencia selecciona Control de atrasos y faltas	
Curso Normal	**Curso Alternativo**
1.- El responsable selecciona fecha de control.	
2.- El sistema realiza el control de asistencia.	2.1. Si el funcionario tiene horario especial definido para esa fecha será validado para el control.
	2.2. Si la institución tiene definido un horario extraordinario para esa fecha es validado para el control
	2.3. Si se tiene definido los minutos de tolerancia son validados para el control.
	2.4. Si el funcionario tiene licencia o vacaciones para esa fecha son validados para el control.
3.- El sistema generar automáticamente los atrasos y faltas correspondientes.	
Precondición: Se debe tener definido los tipos de horarios (Gral. y por funcionario si fue solicitado), las licencias, vacaciones y tolerancias de horario respectivamente. Y también se debe tener el registro de las horas de ingreso marcadas por los funcionarios en la base de datos.	
Postcondición: Calculo de atrasos y faltas.	
Observaciones y datos:	

Tabla 3.18. Descripción narrativa – Controlando Atrasos y Faltas
Fuente: [Elaboración Propia]

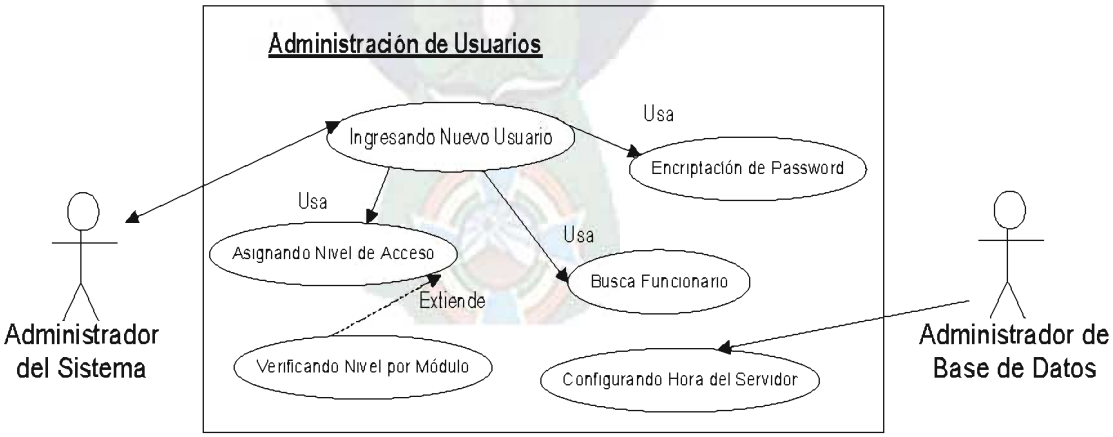


Figura 3.8. Diagrama de Casos de Uso Administración de Usuarios
Fuente: [Elaboración Propia]

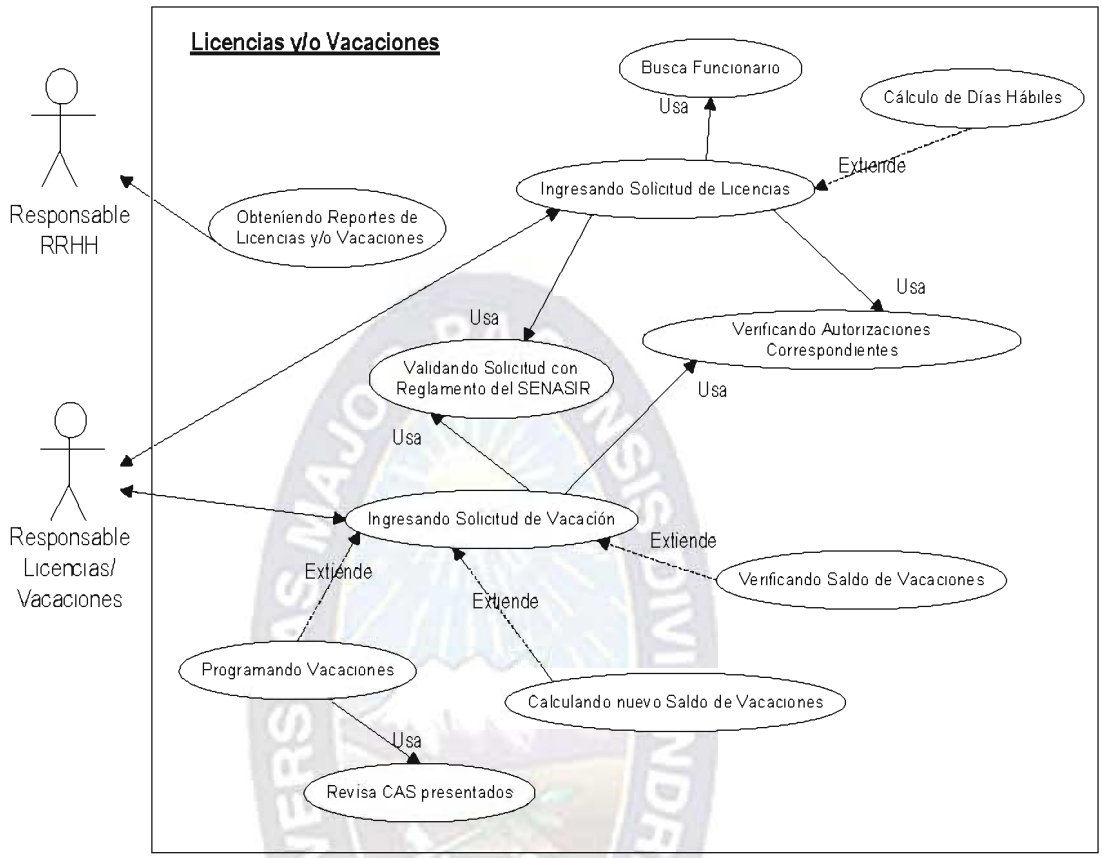


Figura 3.9. Diagrama de Casos de Uso del Módulo de Control de Personal- Licencias y/ o vacaciones
Fuente: [Elaboración Propia]

3.2. FASE 2: DESARROLLO DE UN MODELO GENERAL

Presentamos el diagrama de clases del sistema con las clases más importantes y además divididas por áreas en este caso por los tres módulos siguientes; Registro de Personal, Procesos Organizacionales y Control de Personal.

3.2.1.1. DIAGRAMA DE CLASES

A continuación se presenta el diagrama de clases del sistema, con las clases más relevantes, este diagrama se presenta dividido por módulos, Registro de Personal en la figura 3.9, Procesos Organizacionales en la figura 3.10, y Control de Personal en la figura 3.11.

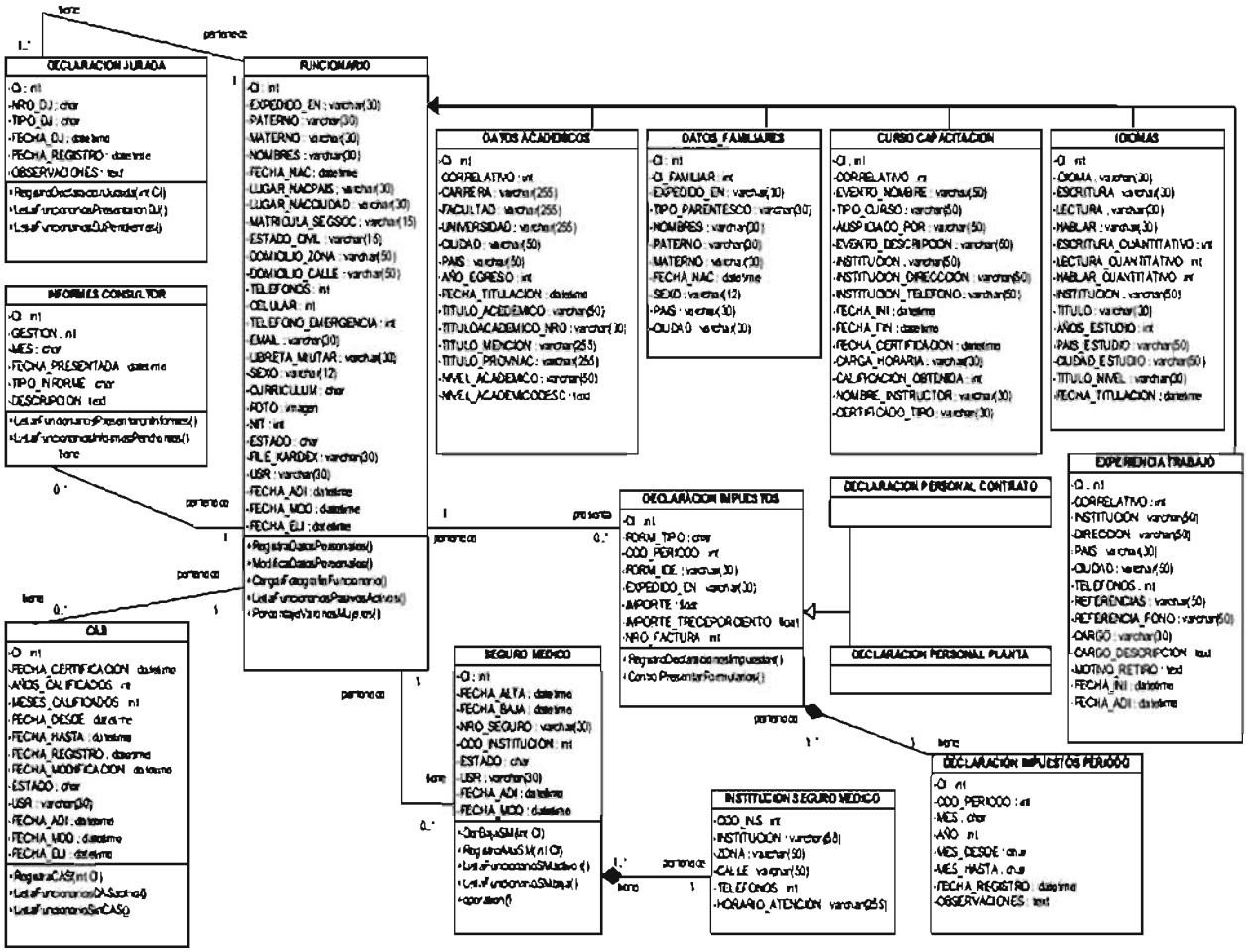


Figura 3.9. Diagrama de Clases: Modulo Registro de Personal
Fuente: [Elaboración Propia]

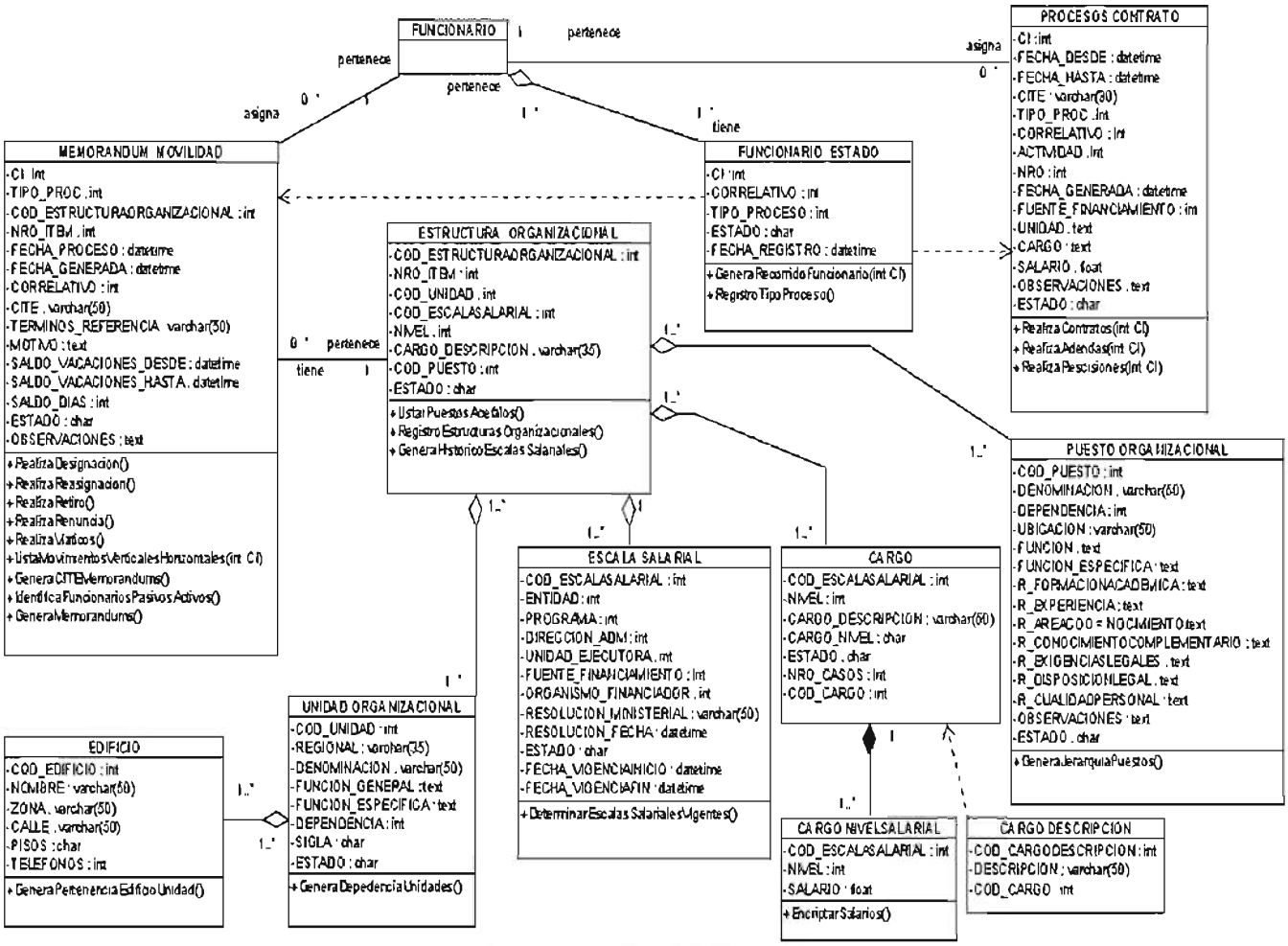


Figura 3.10 Diagrama de Clases: Modulo Proceso Organizacionales
Fuente: [Elaboración Propia]

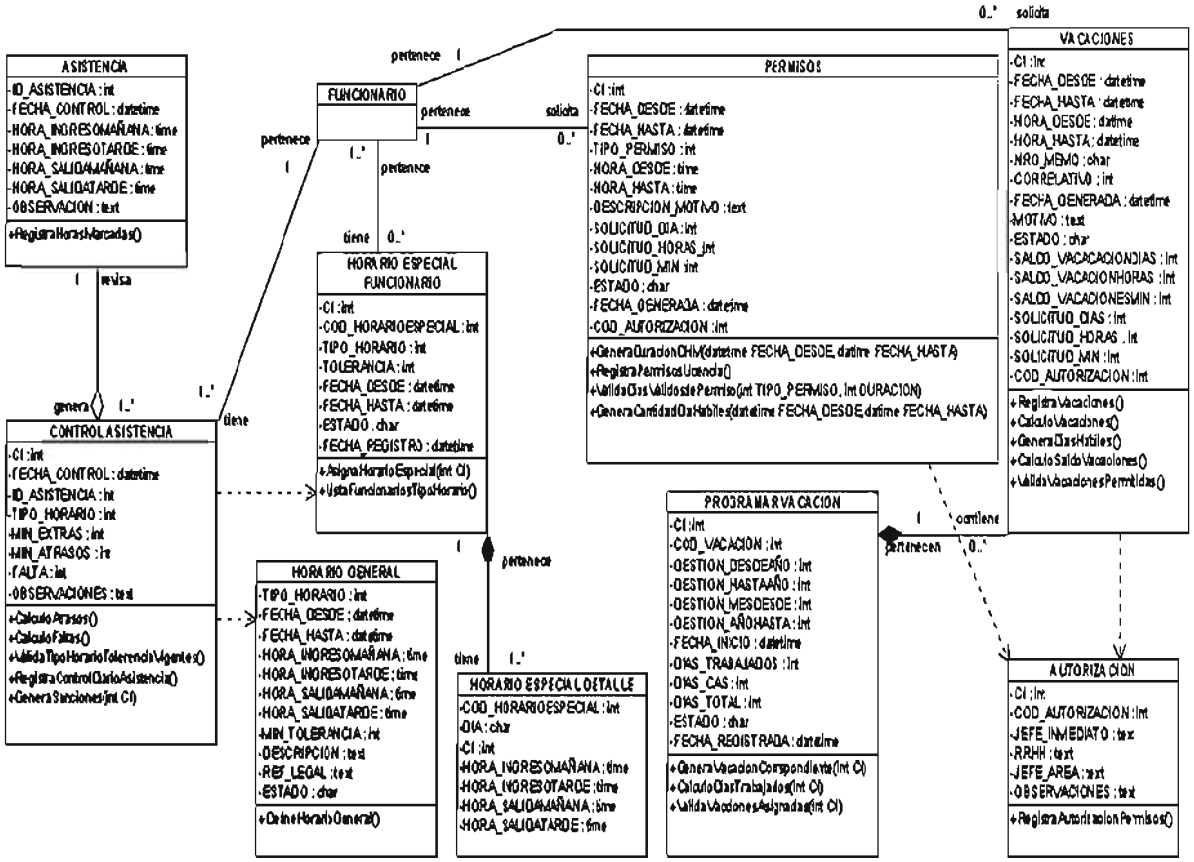


Figura 3.11 Diagrama de Clases: Modulo Control de Personal
Fuente: [Elaboración Propia]

3.3. FASE 3: CONSTRUCCIÓN DE LA LISTA DE FUNCIONALIDADES

En esta fase presentamos las listas de algunas funcionalidades o rasgos del sistema, son presentadas en subconjuntos según la afinidad y la dependencia de cada funcionalidad.

Control de Personal:

- ✓ Conocer el saldo de vacaciones de los funcionarios.
- ✓ Validar todo proceso de control de personal para cumplir los lineamientos legales establecidos por la institución.
- ✓ Generar el cálculo de atrasos, faltas o abandono en cuanto a la asistencia del personal.
- ✓ Contar con una estructura de la base de datos adecuada para el uso de tecnología de control de asistencia "biométrico".
- ✓ Controlar y registrar la asignación de permisos y/o vacaciones de los funcionarios.
- ✓ Generar el cálculo automático de vacaciones para los funcionarios de planta.
- ✓ Calcular el número de días de trabajo del funcionario. Considerando el CAS de cada funcionario.
- ✓ Definir horarios especiales y tolerancias para los funcionarios.
- ✓ Definir horarios generales para la institución.
- ✓ Generar sanciones correspondientes a los funcionarios

Procesos Organizacionales:

- ✓ Conocer cargos e ítems acéfalos
- ✓ Conformar estructuras organizacionales (unidades, puestos, cargos, niveles salariales, ítem) para procesos organizacionales.
- ✓ Reportar el histórico de estructuras organizacionales de la institución.
- ✓ Controlar y hacer el seguimiento a los movimientos verticales y horizontales del funcionario. Desde el momento de su ingreso hasta su retiro de la institución.

- ✓ Realizar designación de nuevos funcionarios, reasignaciones, retiros, renuncias, nuevos contratos, adendas y/o rescisiones de los funcionarios.
- ✓ Generar Memorándums para cada proceso organizacional.
- ✓ Conformar niveles de escalas salariales para las estructuras organizacionales.
- ✓ Encriptar salarios de las escalas salariales.
- ✓ Validar la jerarquía de cargos y puestos al momento de su creación, según la categoría a la que corresponda.
- ✓ Crear unidades organizacionales realizando la asignación de dependencia jerárquica constituyendo el organigrama de la institución.
- ✓ Reportar línea de tiempo del funcionario

Registro de Personal:

- ✓ Realizar el proceso de evaluación del funcionario.
- ✓ Contar con el registro completo del personal de la institución a nivel nacional.
- ✓ Registrar las actividades y cumplimiento de los funcionarios, como presentación de informes, declaraciones de impuestos, declaraciones juradas, seguro médico, etc.
- ✓ Realizar el llenado de formularios de evaluación y generar resultados automáticamente.

Administración de Usuarios:

- ✓ Validar y encriptar la contraseña del usuario.
- ✓ Contar con niveles de acceso para cada módulo.
- ✓ Identificar el usuario, fechas de cada modificación, adición y/o eliminación de cada registro.
- ✓ Crear nuevos usuarios para el sistema.
- ✓ Asignar niveles de usuario según se requiera en cada módulo.

3.4. FASE 4: PLANEAR POR FUNCIONALIDADES.

La lista de funcionalidades elaborado en la fase anterior, en esta fase es presentada de manera ordenada conforme a su prioridad y dependencia, y se asigna a los programadores jefes.

Id.	Descripción	Prog. Jefe	Prog. Clase	Código		Inspección de Código	
				Plan	Actual	Plan	Actual
ID001	Conformar estructuras organizacionales (unidades, puestos, cargos, niveles salariales, ítem) para procesos organizacionales.	CJ	ABC	05/01/2009	05/01/2009	19/01/2009	19/01/2009
ID002	Conformar niveles de escalas salariales para las estructuras organizacionales	CJ	ABC	14/01/2009	14/01/2009	21/01/2009	21/01/2009
ID003	Realizar designación de nuevos funcionarios, reasignaciones, retiros, renuncias, nuevos contratos, adendas y/o rescisiones de los funcionarios	CJ	ABC	20/01/2009	21/01/2009	23/01/2009	24/01/2009
ID004	Controlar y hacer el seguimiento a los movimientos verticales y horizontales del funcionario. Desde el momento de su ingreso hasta su retiro de la institución	CJ	ABC	20/01/2009		29/01/2009	
ID005	Encriptar salarios de las escalas salariales	CJ	ABC	20/01/2009		29/01/2009	
ID006	Validar la jerarquía de cargos y puestos al momento de su creación, según la categoría a la que corresponda.	CJ	ABC	20/01/2009		29/01/2009	

Id.	Descripción	Prog. Jefe	Prog. Clase	Código		Inspección de Código	
				Plan	Actual	Plan	Actual
ID007	Crear unidades organizacionales realizando la asignación de dependencia jerárquica constituyendo el organigrama de la institución.	CJ	ABC	20/01/2009	20/012009	29/01/2009	29/01/2009
ID008	Generar Memorándums para cada proceso organizacional.			20/01/2009	20/012009	29/01/2009	29/01/2009
ID009	Conocer cargos e ítems acéfalos	CJ	ABC	20/01/2009	20/012009	29/01/2009	29/01/2009
ID010	Reportar el histórico de estructuras organizacionales de la institución.			20/01/2009	20/012009	29/01/2009	29/01/2009
ID011	Reportar línea de tiempo del funcionario	CJ	ABC	20/01/2009	20/012009	29/01/2009	29/01/2009
ID012	Contar con una estructura de la base de datos adecuada para el uso de tecnología de control de asistencia "biométrico".	CJ	ABC	02/02/2009	20/012009	12/02/2009	
ID013	Generar el cálculo de atrasos, faltas o abandono en cuanto a la asistencia del personal.	CJ	ABC	10/02/2009		20/02/2009	
ID014	Generar sanciones correspondientes a los funcionarios	CJ	ABC	10/02/2009		27/02/2009	
ID015	Controlar y registrar la asignación de permisos y/o vacaciones de los funcionarios	CJ	ABC	10/02/2009	11/02/2009	27/02/2009	
ID016	Calcular el número de días de trabajo del funcionario. Considerando el CAS de cada funcionario.	CJ	ABC	10/02/2009		27/02/2009	
ID017	Generar el cálculo automático de vacaciones para los funcionarios de planta.	CJ	ABC	10/02/2009		27/02/2009	

Id.	Descripción	Prog. Jefe	Prog. Clase	Código		Inspección de Código	
				Plan	Actual	Plan	Actual
ID018	Conocer el saldo de vacaciones de los funcionarios.	CJ	ABC	10/02/2009	10/02/2009	27/02/2009	27/02/2009
ID019	Validar todo proceso de control de personal para cumplir los lineamientos legales establecidos por la institución	CJ	ABC	10/02/2009	10/02/2009	27/02/2009	27/02/2009
ID020	Definir horarios especiales y tolerancias para los funcionarios.	CJ	ABC	10/02/2009	10/02/2009	27/02/2009	27/02/2009
ID021	Definir horarios generales para la institución.	CJ	ABC	10/02/2009	10/02/2009	27/02/2009	27/02/2009
ID022	Realizar el proceso de evaluación del funcionario.	CJ	ABC	02/03/2009		24/03/2009	
ID023	Realizar el llenado de formularios de evaluación y generar resultados automáticamente.	CJ	ABC	02/03/2009		24/03/2009	
ID024	Registrar las actividades y cumplimiento de los funcionarios, como presentación de informes, declaraciones de impuestos, declaraciones juradas, seguro médico, etc.	CJ	ABC	02/03/2009		24/03/2009	
ID025	Contar con el registro completo del personal de la institución a nivel nacional.	CJ	ABC	02/03/2009	02/03/2009	24/03/2009	24/03/2009
ID026	Validar y encriptar la contraseña del usuario.	CJ	ABC	23/03/2009		28/03/2009	
ID027	Identificar el usuario, fechas de cada modificación, adición y/o eliminación de cada registro	CJ	ABC	23/03/2009	23/03/2009	28/03/2009	
ID028	Contar con niveles de acceso para cada módulo.	CJ	ABC	23/03/2009	23/03/2009	28/03/2009	

Id.	Descripción	Prog. Jefe	Prog. Clase	Código		Inspección de Código	
				Plan	Actual	Plan	Actual
ID029	Asignar niveles de usuario según se requiera en cada módulo	CJ	ABC	23/03/2009	23/03/2009	28/03/2009	28/03/2009
ID030	Crear nuevos usuarios para el sistema.	CJ	ABC	23/03/2009	23/03/2009	28/03/2009	28/03/2009

Tabla 3. 19 Lista de Funcionalidades ordenados por prioridad.
Fuente: [Elaboración Propia]

3.5. FASE 5: DISEÑAR POR FUNCIONALIDADES

3.5.1. DIAGRAMA DE SECUENCIAS Y DIAGRAMAS DE COLABORACIÓN

Los diagramas de Secuencia y diagramas de colaboración nos ayudaran en el diseño de las funcionalidades citadas anteriormente, a través de estos diagramas se tendrá una mejor visualización de las interacciones entre objetos, debido a que se muestra la secuencia cronológica en que ocurren las cosas.

En el diagrama de secuencia se indicarán los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada.

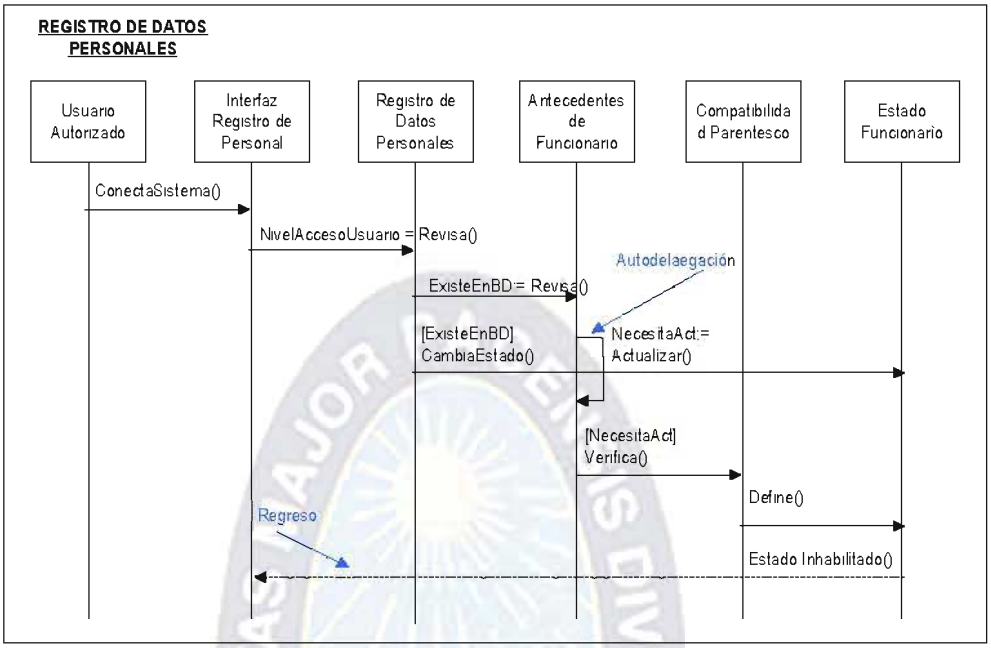


Figura 3.12. Diagrama de Secuencia del Registro de Datos Personales
Fuente: [Elaboración Propia]

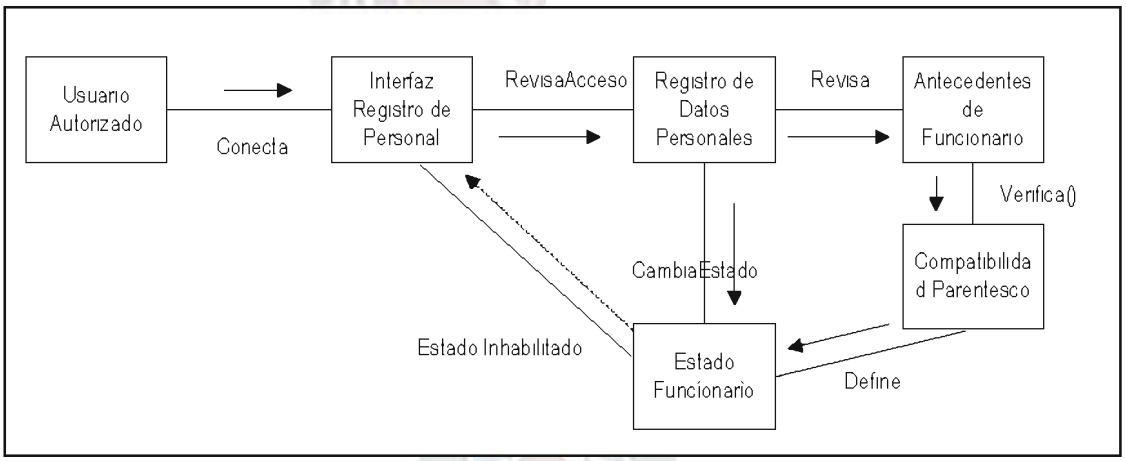


Figura 3.13. Diagrama de Colaboración del Registro de Datos Personales
Fuente: [Elaboración Propia]

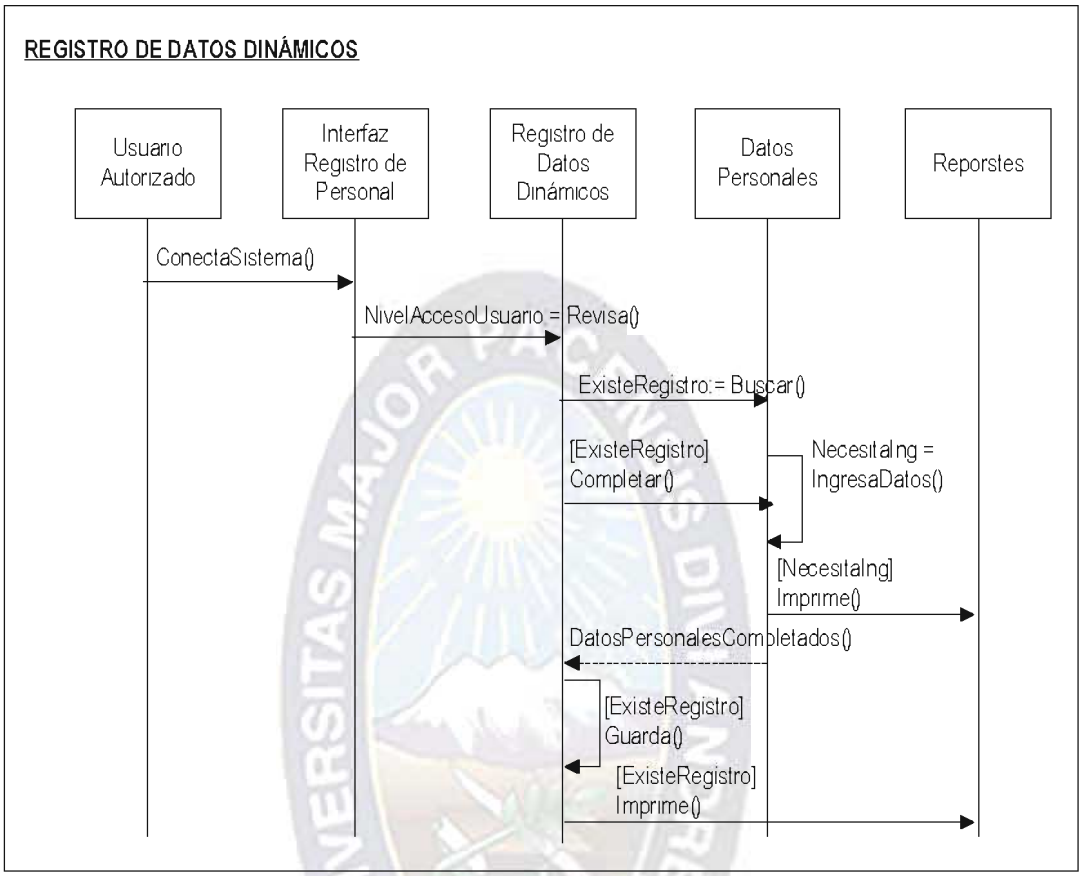


Figura 3.14 Diagrama de Secuencia del Registro de Datos Dinámicos
Fuente: [Elaboración Propia]

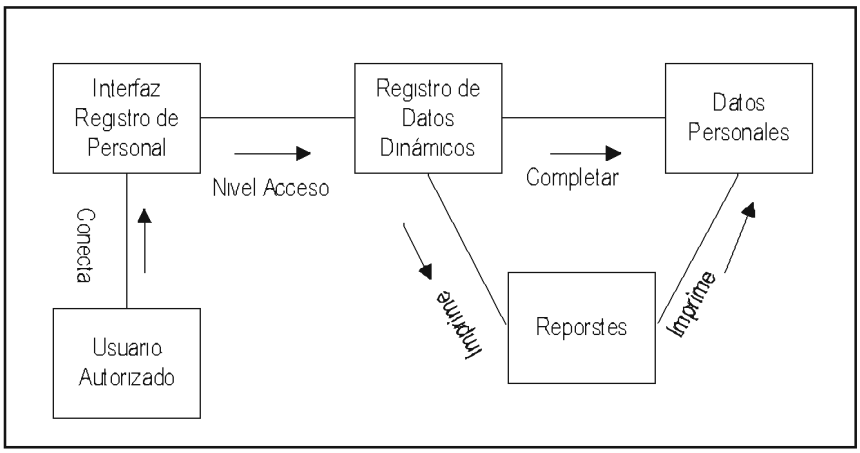


Figura 3.15. Diagrama de Colaboración del Registro de Datos Dinámicos
Fuente: [Elaboración Propia]

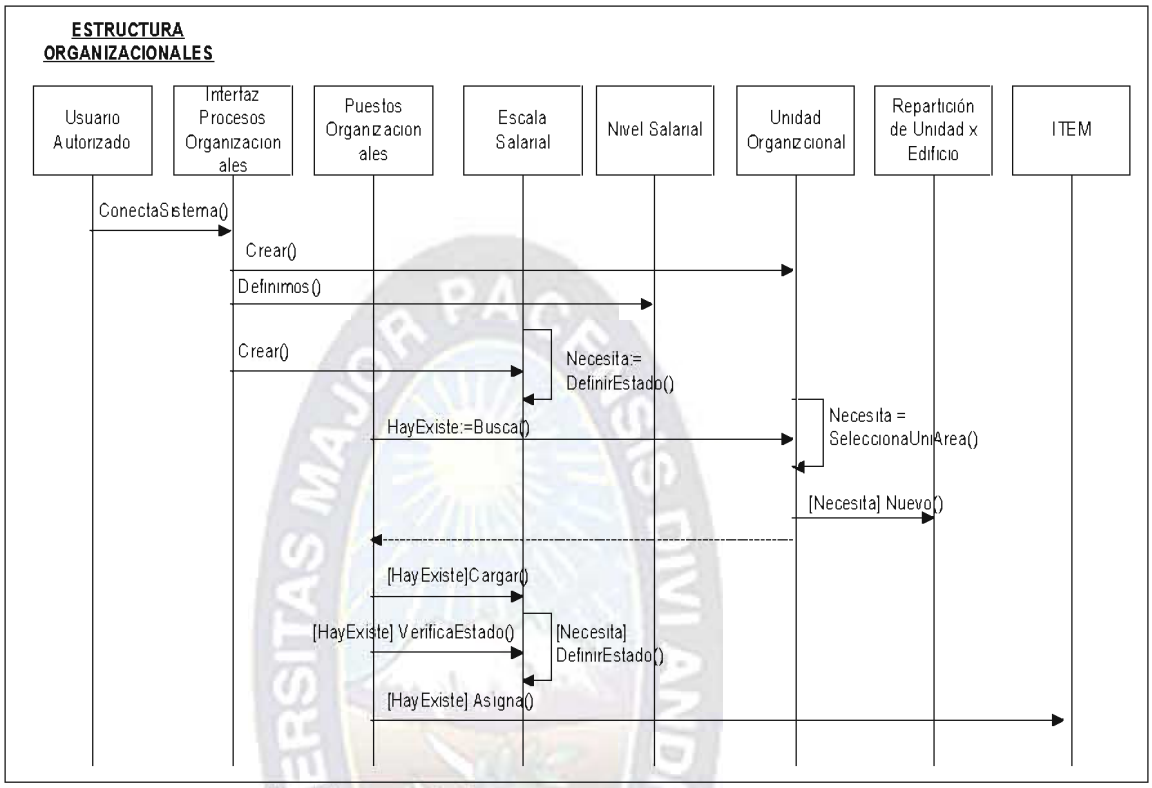


Figura 3.16. Diagrama de Secuencia de Estructuras Organizacionales
Fuente: [Elaboración Propia]

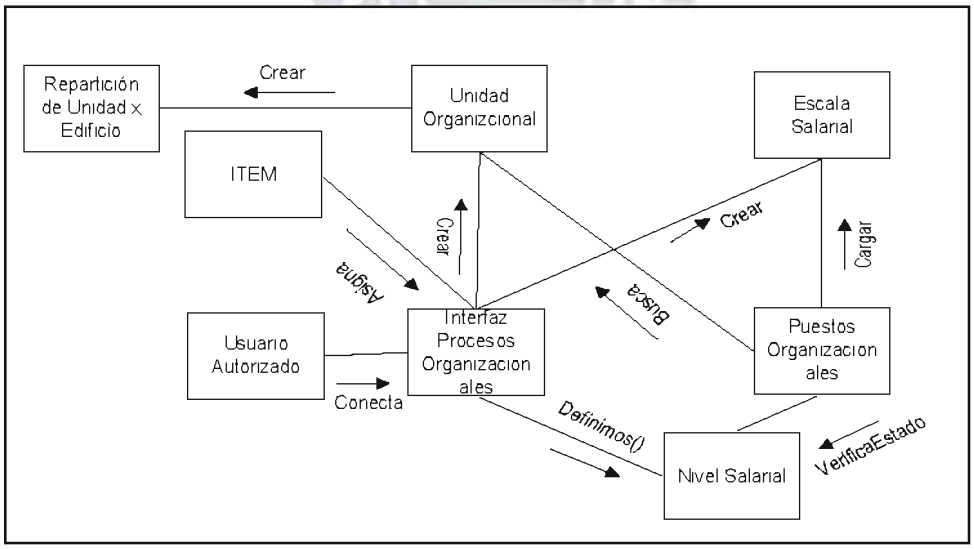


Figura 3.17. Diagrama de Colaboración de Estructuras Organizacionales
Fuente: [Elaboración Propia]

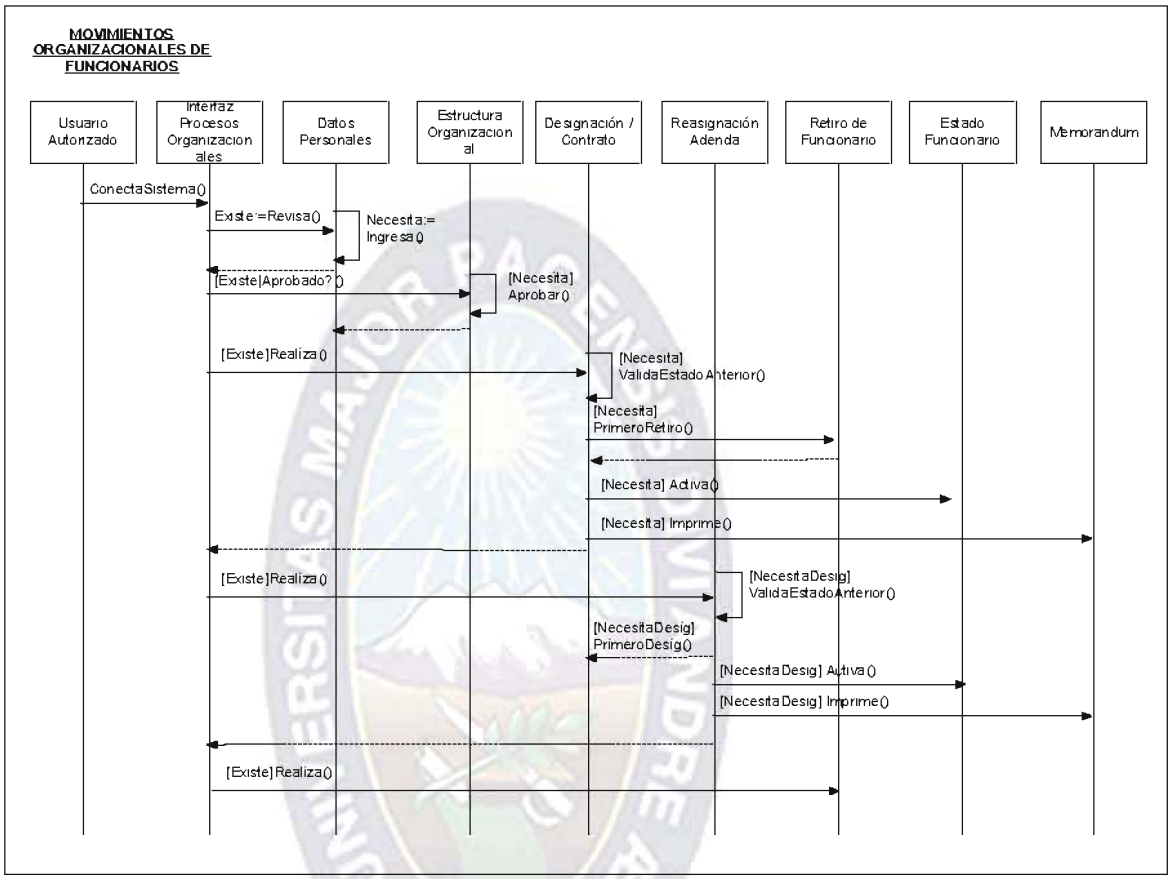


Figura 3.18. Diagrama de Secuencia de Movimientos Organizacionales de Funcionarios
Fuente: [Elaboración Propia]

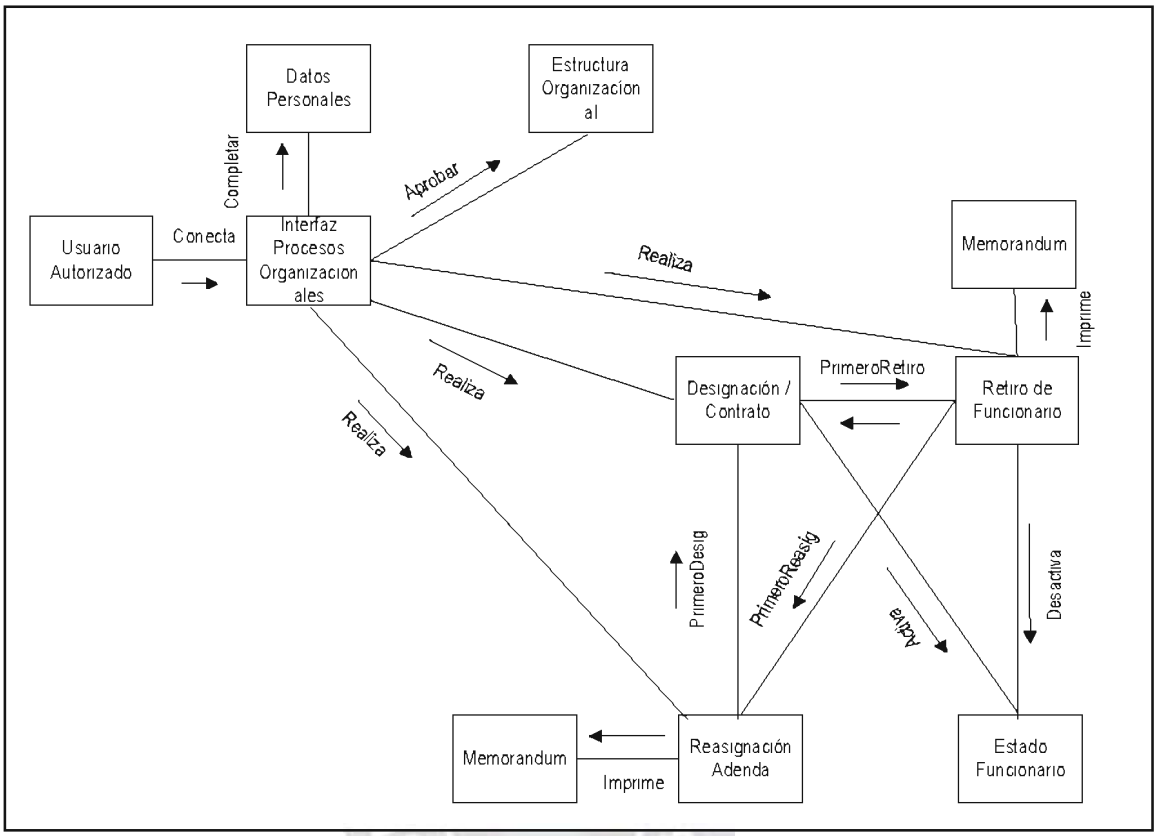


Figura 3.19. Diagrama de Colaboración de Movimientos Organizacionales de Funcionarios
Fuente: [Elaboración Propia]

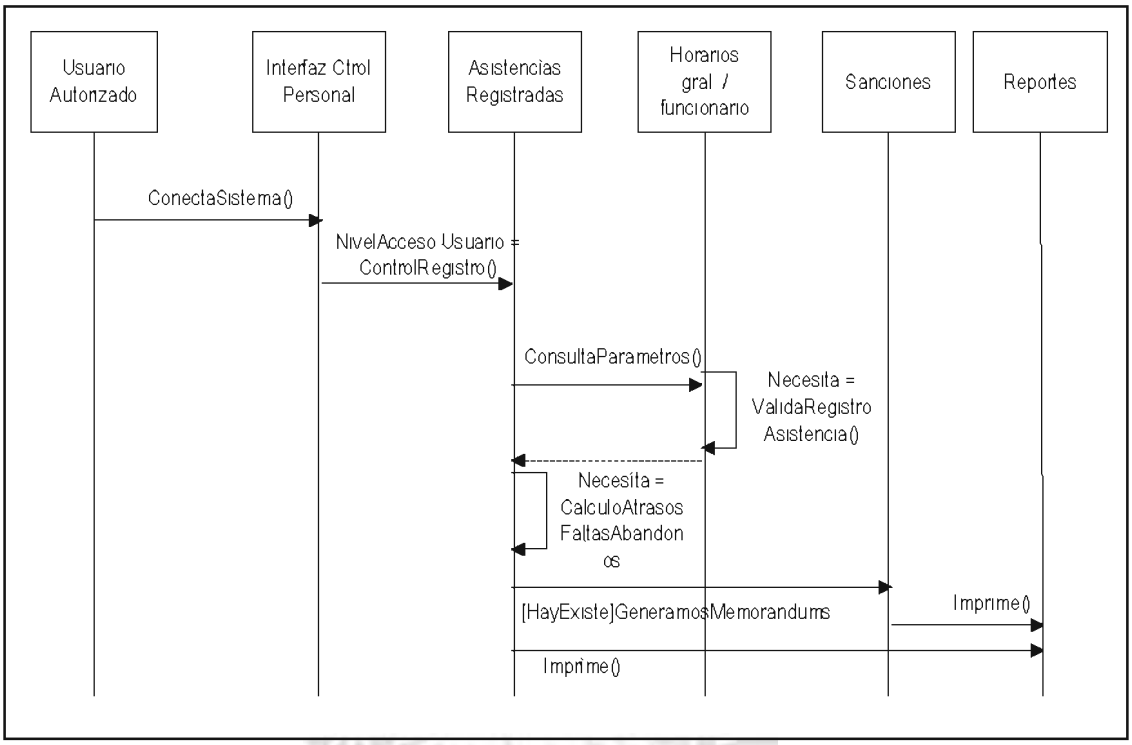


Figura 3.20. Diagrama de Secuencia de Control de Asistencia
Fuente: [Elaboración Propia]

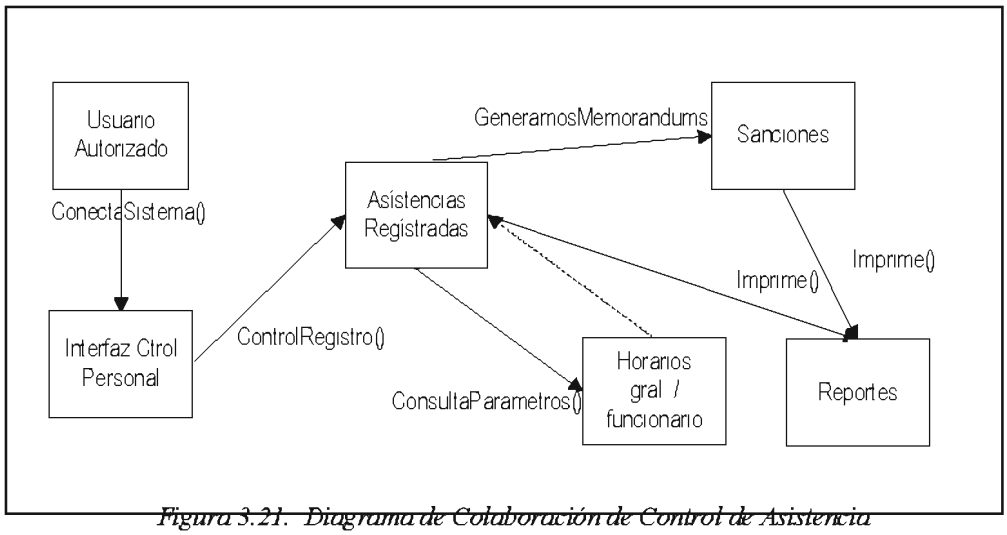


Figura 3.21. Diagrama de Colaboración de Control de Asistencia
Fuente: [Elaboración Propia]

3.6. FASE 6: CONSTRUCCIÓN POR FUNCIONALIDADES

3.6.1. DIAGRAMA DE COMPONENTES

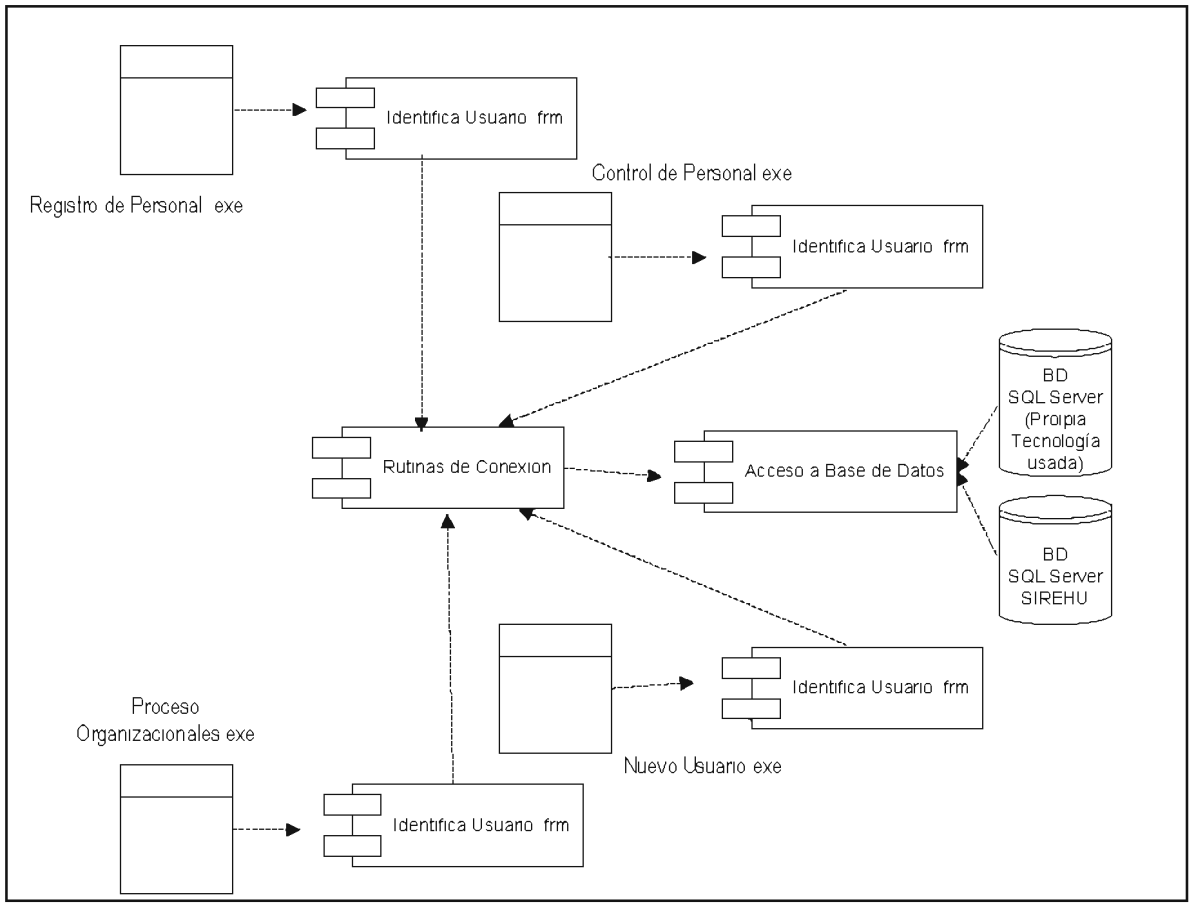


Figura 3.22. Diagrama de Componentes del SIREHU
Fuente: [Elaboración Propia]

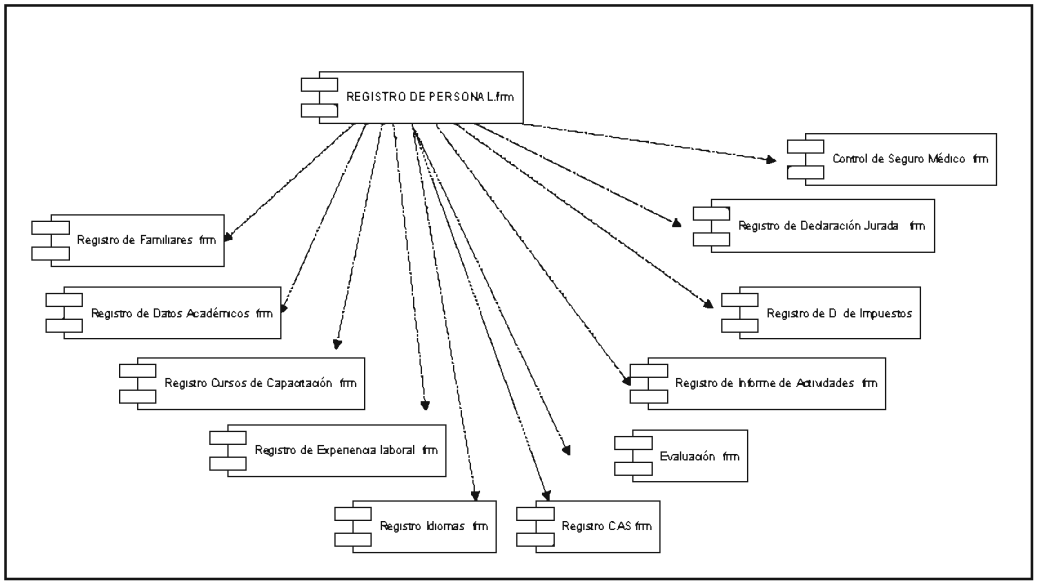


Figura 3.23. Diagrama de Componentes del Modulo de Registro de Personal
Fuente: [Elaboración Propia]

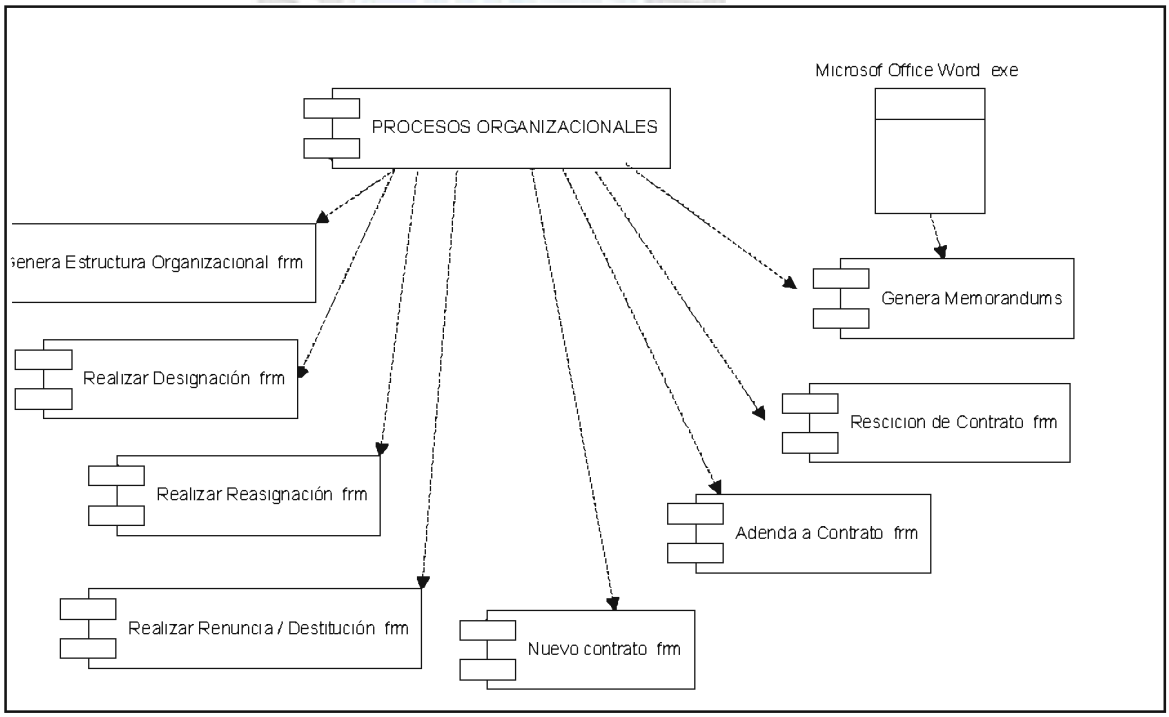


Figura 3.24. Diagrama de Componentes del Modulo de Procesos Organizacionales
Fuente: [Elaboración Propia]

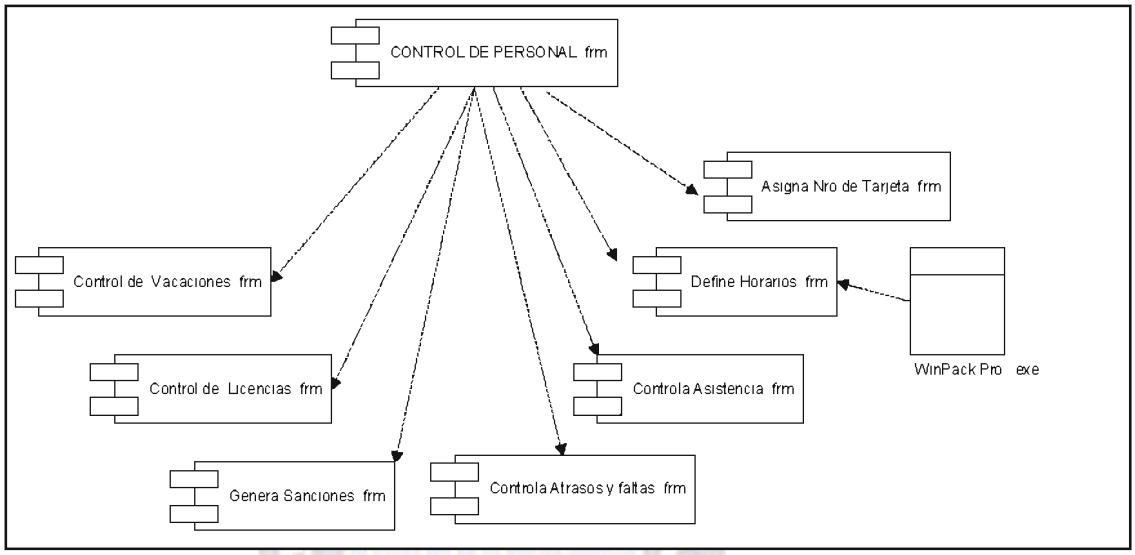


Figura 3.25. Diagrama de Componentes del Modulo de Control de Personal
Fuente: [Elaboración Propia]

3.6.2. DIAGRAMA DE DESPLIEGUE

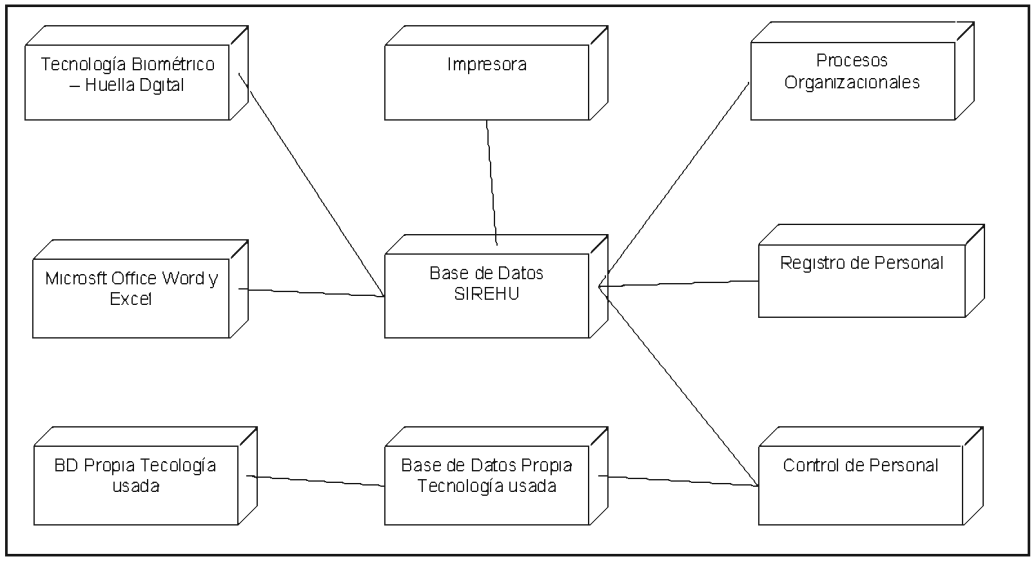


Figura 3.26. Diagrama de Despliegue del Sistema
Fuente: [Elaboración Propia]

3.6.3. INTERFAZ GRÁFICA DE USUARIO

Figura 3.27. Identificación de usuario y pantalla principal del Sistema
Fuente: [Elaboración Propia]

Modulo de Registro de Personal

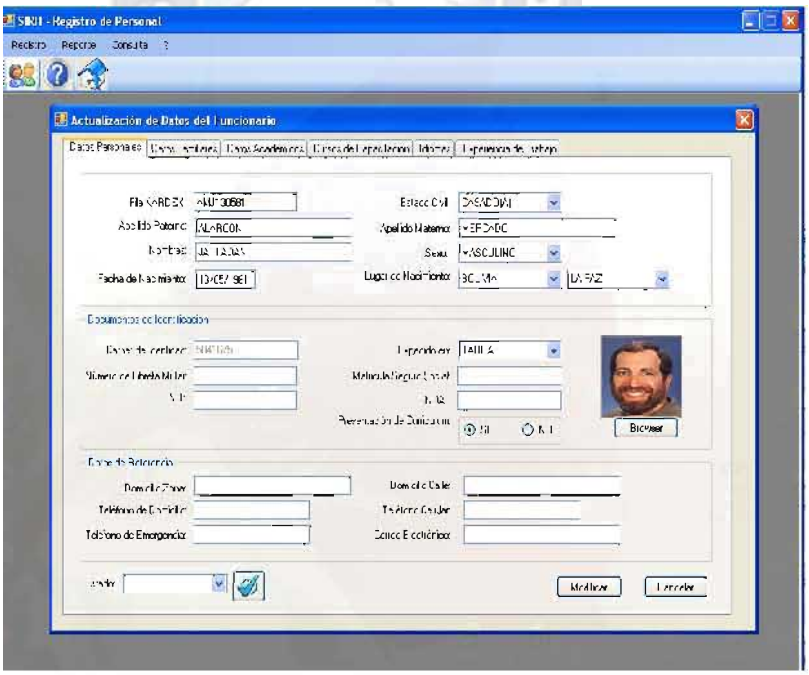


Figura 3.28. Pantalla de Registro de Personal: Actualización de datos
Fuente: [Elaboración Propia]

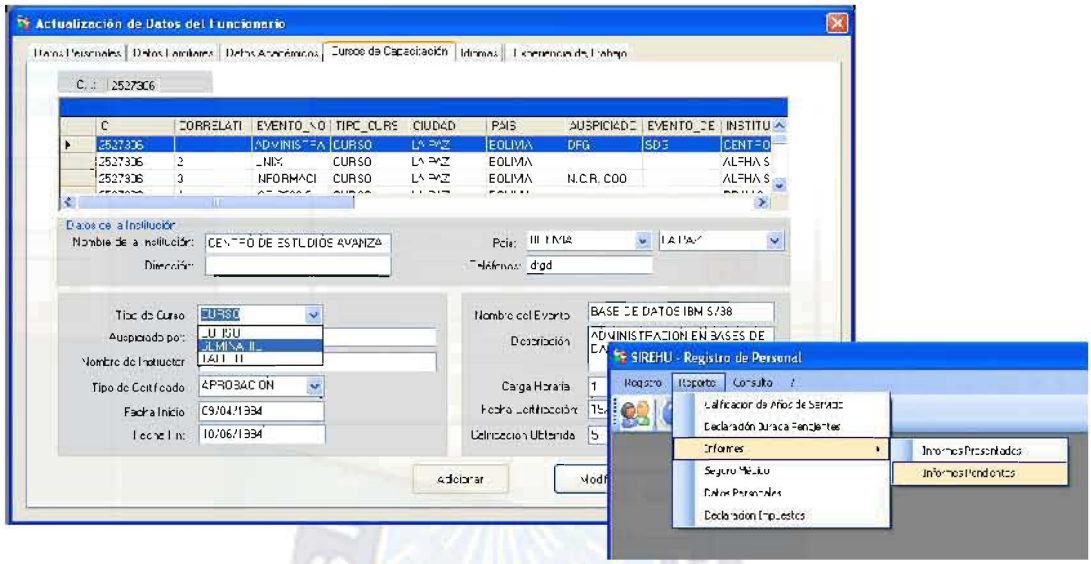


Figura 3.29. Pantalla de Registro de Personal: Cursos Capacitación
Fuente: [Elaboración Propia]

Modulo Proceso Organizacionales

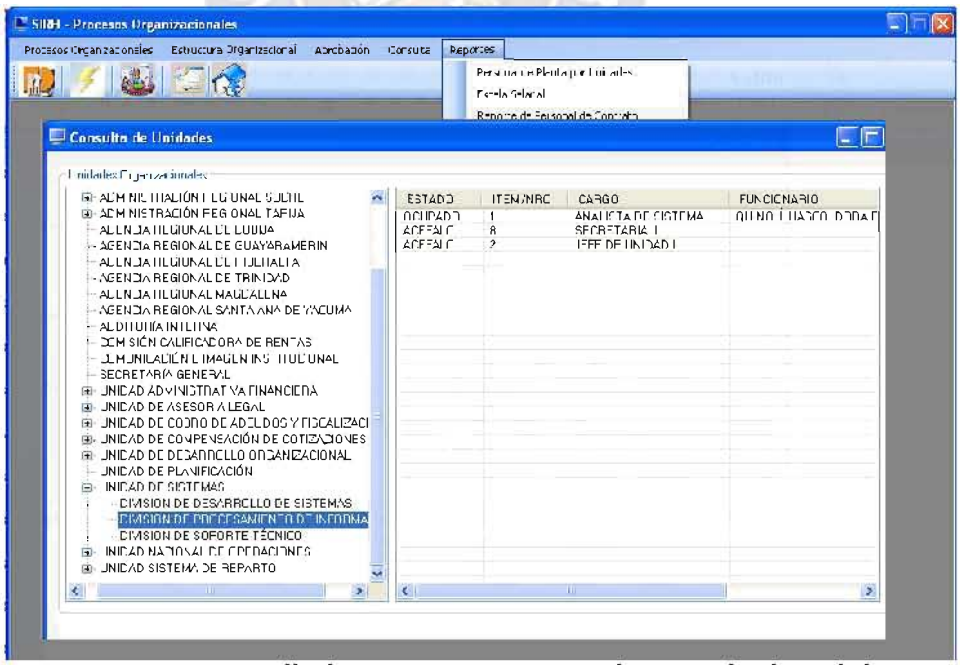


Figura 3.30. Pantalla de Procesos Organizacionales: Consulta de unidades
Fuente: [Elaboración Propia]

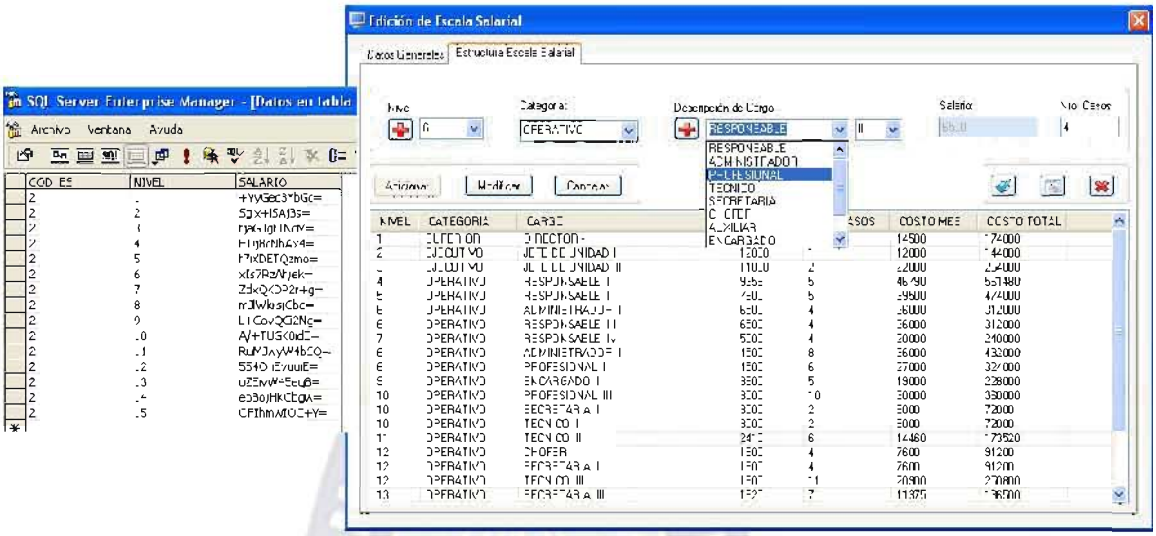


Figura 3.31. Pantalla de Procesos Organizacionales: Edición Escalas Salariales
Fuente: [Elaboración Propia]

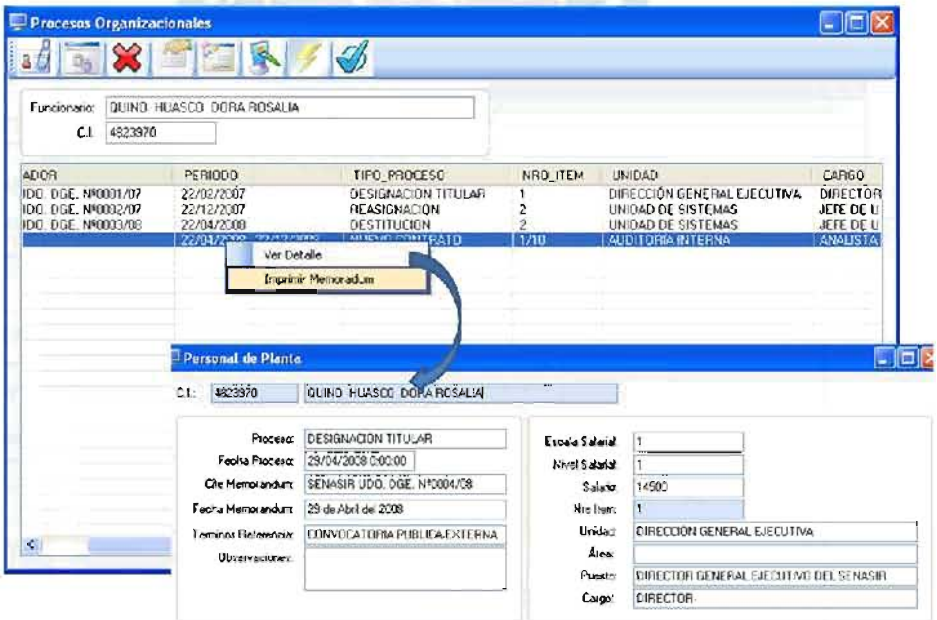


Figura 3.32. Pantalla de Procesos Organizacionales: Movilidad de Funcionarios
Fuente: [Elaboración Propia]

Modulo Control de Personal

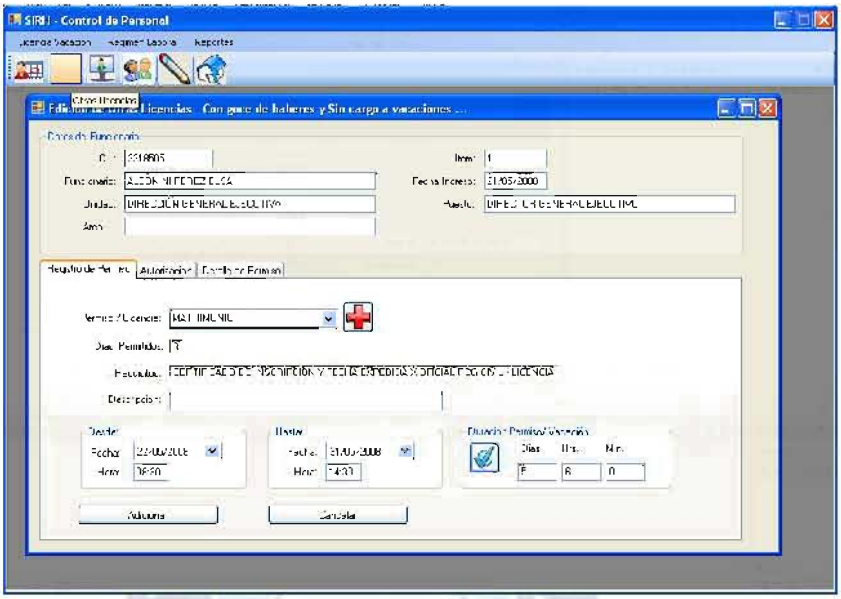


Figura 3.33. Pantalla de Control de Personal: Asignación de Licencias/ permisos
Fuente: [Elaboración Propia]

CAPITULO IV
PRUEBAS Y
CALIDAD DEL SOFTWARE

4.1. INTRODUCCIÓN

Las métricas de calidad aportan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. A veces, cuando se intentan obtener medidas precisas de la calidad del software se acaba frustrado por la naturaleza subjetiva de esta actividad. Para resolver este problema, se buscan medidas cuantitativas de la calidad del software, para poder llevar a cabo un análisis objetivo.

Pero no es posible medir la calidad del software de forma exacta, ya que cada medida es parcialmente imperfecta. Las medidas de calidad siempre son indirectas, ya que no se mide directamente la calidad sino algunas de sus manifestaciones. El factor que lo complica es la relación precisa entre la variable que es medida y la calidad del software

Para las pruebas del sistema de información SIREHU, se considera las Pruebas de Caja Blanca y la Caja Negra, y la norma ISO 9126 para la Calidad del Software, además del uso de métricas en los factores identificados de la ISO 9126.

4.2. EVALUACION / PRUEBAS DE SOFTWARE

4.2.1. PRUEBAS DE CAJA BLANCA O ESTRUCTURALES

El objetivo de aplicar este método es de diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, como se especifico en el capítulo del Marco Teórico. Los pasos a realizar son:

- Representar el programa en un grafo de flujo
- Calcular la complejidad ciclomática
- Determinar el conjunto básico de caminos independientes
- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

Representar el programa en un grafo de flujo.-

```
5.  AccesoBD.IdeUSUARIO = txtUsuario.Text;
6.  AccesoBD.IdeClave = txtPassword.Text;
7.  if ((txtUsuario.Text == "") || (txtPassword.Text == ""))
8.  MessageBox.Show("Por favor complete los datos solicitados...");
9.  return;
10. CodigoDesencriptado = Convert.ToString(AccesoBD.Encrypt(txtPassword.Text, true));
11.
12. string Usu = "SELECT COUNT(*) FROM S_USUARIO WHERE LOGIN = '" +
    txtUsuario.Text + "' AND PASWORD = '" + CodigoDesencriptado + "' AND ESTADO
    = 'A'";
13. SqlDataAdapter daUsu = new SqlDataAdapter(Usu, cn);
14. DataTable dtUsu = new DataTable();
15. daUsu.Fill(dtUsu);
16. foreach (DataRow drUsu in dtUsu.Rows)
17. {
18.  if (Convert.ToInt32(drUsu[0]) > 0)
19.    {  SIRH.frmMenu frm = new frmMenu(txtUsuario.Text, txtPassword.Text);
20.       frm.Show();
21.       this.Hide();
22.    }
23.    else{MessageBox.Show("Usuario o Password incorrectos...Por favor Verifique sus
    datos ...");
24.        Limpia();
25.        return;
26.       }
27.  }
```

Figura 4.1. Fragmento de código para el control de acceso de usuario
Fuente: [Elaboración Propia]

A partir del fragmento de código presentado, se construye el grafo de acceso a los módulos del sistema de información.

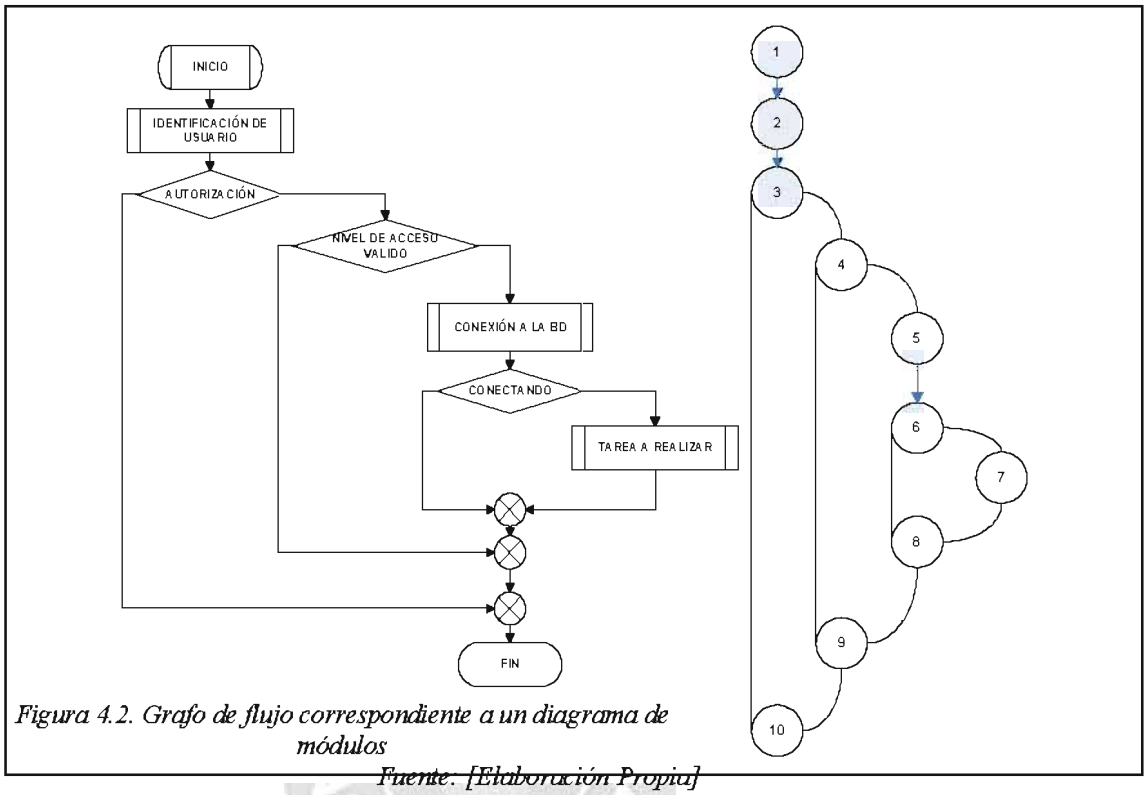


Figura 4.2. Grafo de flujo correspondiente a un diagrama de módulos

Fuente: [Elaboración Propia]

Calcular la complejidad ciclomática.- Ahora se calcula la complejidad ciclomática. Existen varias formas a partir de un grafo de flujo:

- El número de regiones del grafo coincide con la complejidad ciclomática, V(G).
- La complejidad ciclomática, V(G), de un grafo de flujo G se define como V(G) = Aristas – Nodos + 2
- La complejidad ciclomática, V(G), de un grafo de flujo G se define como V(G) = Nodos Predicado + 1

Entonces,

V(G) = Número de regiones = 3

V(G) = Aristas – Nodos + 2 = 11-10 + 2 = 3

V(G) = Nodos Predicado + 1 = 2 +1 = 3

Determinar el conjunto básico de caminos independientes.- El valor V(G) nos indica que son tres los casos de prueba que deben de ejecutarse y diseñar para garantizar que cubren todas las sentencias del programa. Tres caminos independientes generados serian:

Camino1: 1 – 2 – 3 - 10

Camino 2: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10

Camino 3: 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 - 10

Estos cuatro caminos constituyen el camino básico para el grafo de flujo correspondiente.

Por tanto, la aplicación de este criterio de cobertura asegura que la sentencia del fragmento de código evaluado se ejecuta al menos una vez y que las condiciones fueron probadas tanto para un valor verdadero como falso.

4.3. EVALUACION / CALIDAD DE SOFTWARE

4.3.1. PUNTO FUNCION

A través de Punto Función se establecerá el tamaño y complejidad del sistemas de información basada en la cantidad de funcionalidad requerida y entregada a los usuarios. Primero identificamos aspectos importantes, como el número de entradas, salidas, entre otros.

- ✓ Número de entradas de usuario: 64
- ✓ Numero de salidas de usuario: 35
- ✓ Numero de peticiones de usuario: 30

- ✓ Numero de archivos: 55
- ✓ Interfaces externas: 3

Bajo los datos establecidos se completa la siguiente tabla, considerando un factor de ponderación medio.

Parámetros de Medición	Cuenta	Fac. Ponderación	Totales
Nro de entradas	64	4	256
Nro de salidas	35	5	175
Nro de consulta	30	4	120
Nro de archivos	55	10	550
Interfaces externas	3	7	21
			1122

Tabla 4.1. Factores de Ponderación
Fuente: [Elaboración Propia]

Se presenta la información paramétrica de los valores de ajuste de complejidad.

Código	Descripción	Valor
SI	Sin Importancia	0
I	Incidental	1
MO	Moderado	2
ME	Medio	3
SI	Significativo	4
E	Esencial	5

Tabla 4.2. Valores de Ajuste de complejidad
Fuente:[Elaboración Propia]

Definimos el Ajuste de la complejidad.

Escala	SI: Sin Importancia	I: Incidental	MO: Moderado	ME: Medio	S:Significativo	E: Esencial
	0	1	2	3	4	5
¿Requiere el sistema copias de seguridad y de recuperación fiable?						X
¿Se requiere comunicación de datos?					X	
¿Existen funciones de procesos distribuidos?				X		
¿Es critico el rendimiento?	X					
¿Será ejecutado el sistema en S.O. existente?		X				
¿Requiere el sistema copias de seguridad y de recuperación fiable?				X		
¿Requiere el sistema de entradas interactivas en línea?				X		
¿Requiere el sistema entrada de datos interactiva sobre múltiples ventanas?					X	
¿Se actualizan los archivos maestros de manera interactiva?						x
¿Son complejas las entradas, las salidas, los archivos o las peticiones?			X			
¿Es complejo el procesamiento interno?				X		
¿Se ha diseñado el código para ser reutilizable?						X
¿Están incluidas en el diseño la conversión y la instalación?			X			
¿Se ha diseñado el sistema para soportar múltiples instalaciones?				X		
¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?						X
Σ (Fj)						**48**

Tabla 4.3. Definición de ajuste
Fuente: [Elaboración Propia]

Entonces, considerando un grado de complejidad mínimo, se tiene.

PF = CuentaTotal * (Grado de Confiabilidad + tasa de error * Σ Fi)

= 1122 * (0.65 + 0.01 * 48)

=1267,86

Ahora considerando el máximo valor de complejidad tenemos;

PF = CuentaTotal * (Grado de Confiabilidad + tasa de error * Σ Fi)

= 1122 * (0.65 + 0.01 * 70)

=1514,7

Entonces, si **Σ** Fj es considerada como el 100%, la relación obtenida entre los puntos será:

PF / PF máximo = 1267,86 / 1514,7 = **0,84**

Por tanto la funcionalidad del sistema es del 84 % tomando en cuenta el punto de función máximo.

4.3.2. ISO 9126

Características	Pregunta central
Funcionalidad	¿Las funciones y propiedades satisfacen las necesidades explícitas e implícitas; esto es, el qué ...?
Confiabilidad	¿Pueden mantener el nivel de rendimiento, bajo ciertas condiciones y por cierto tiempo?
Usabilidad	¿El software es fácil de usar y de aprender?
Eficiencia	¿Es rápido y minimalista en cuanto el uso de recursos?
Mantenibilidad	¿Es fácil de modificar y verificar?
Portabilidad	¿Es fácil de transferir de un ambiente a otro?

Tabla 4.4. Preguntas centrales de para métricas de calidad de la ISO 9126
Fuente: [Elaboración Propia]

FUNCIONALIDAD.- Las funciones y propiedades satisfacen las necesidades éxplicits e implícitas.

- ✓ **Adecuación,** el sistema presenta un conjunto de funciones apropiadas para las tareas especificas, en cada uno de los módulos que conforman el sistema.
- ✓ **Exactitud,** hace y cumple con cada uno de los requisitos definidos en la fase de ingeniería de requerimientos
- ✓ **Interoperabilidad,** ningún modulo interactúa con otros sistemas.
- ✓ **Conformidad,** el sistema, esencialmente el modulo de control de personal, esta diseñado de tal forma que se encuentra bajo el reglamento interno de control de personal del SENASIR.
- ✓ **Seguridad de acceso,** presenta el sistema un diseño seguro, evitando que usuarios no autorizados ingresen al sistema.

CONFIABILIDAD

Nivel de madurez, las fallas o errores presentados por el sistema no se presentan frecuentemente, en la fase inicial del sistema se realizo la corrección y el ajuste necesario para corregir los errores que presento el sistema.

Para determinar la confiabilidad del sistema se especifica el instante en el que comienza a funcionar, determinado por $to = 0$. A partir de este momento se observa el trabajo del sistema hasta que se introduzca una falla en el instante T que se va aproximando a una variable aleatoria continua, que nos determina la confiabilidad en términos probabilístico. Entonces se tiene las siguientes probabilidades:

$$P(T <= t) = F(t) \quad \text{...(1) Probabilidad de fallas}$$

$$P(T > t) = 1 - F(t) \quad \text{...(2) Probabilidad de trabajo sin fallas}$$

Se calcula estas probabilidades mediante la distribución exponencial.

Donde,

Punto función, resultado obtenido en el cálculo realizado en un punto anterior a este.

Fueron realizadas 8 ejecuciones por mes en un periodo de ***t = 9 meses.***

Con un margen de error **λ = 1/10.**

Entonces se tiene,

$$F(t) = \text{Punto función} * e^{(-\lambda * t)} \quad \text{...ecuación de confiabilidad}$$

$$= 0{,}84\, e^{(-\lambda * t)}$$

$$= 0{,}84\, e^{(-(1/10)*9)}$$

$$= \mathbf{0{,}19}$$

Por tanto, el sistema presenta una probabilidad de fallo de un 19%. En lo posible el sistema es capaz de recuperar los datos.

USABILIDAD.- El sistema es de fácil uso, los procesos complicados están modularizados para una mejor comprensión.

- ✓ **Comprensibilidad**, las pantallas del sistema presentan gráficos para una mejor comprensión.
- ✓ **Facilidad de aprender,** el sistema es fácil aprender a usarlo.
- ✓ **Operabilidad,** presenta facilidad en controlar los módulos, y fácil de operarlo.

EFICIENCIA.-

- ✓ **Comportamiento respecto al tiempo,** se resuelve los problemas de usuario en un tiempo aceptable.

Tiempo medio: Tav= Sum(Tu) / N, donde **Tu = Trc – Tsn**

Tsn= Tiempo de envio de la petición por el usuario

Trc= Tiempo en el que usuario recibe la versión revisada

N= numero de versiones revisadas

Entonces;

Tu = (12 – 10) + (21 -19) + (4 – 3) + (11 -10) + (25 – 18) + (4 -3)

Tu = 14

Tav = 14 / 6 = **2,33**

Por tanto como 0 < 2.33 y Tav es una cantidad pequeña se considera un resultado bueno, dado que las peticiones del usuario son atendidas de manera rápida.

- ✓ **Comportamiento respecto a recursos,** los recursos utilizados por el sistema son mínimos.

MANTENIMIENTO

Se calcula el Índice de Madurez del Software (IMS), estableciendo los cambios que ocurrieron con cada versión del producto.

- ***MT:*** Nro. de módulos en la versión actual 4
- ***Fc:*** Nro. de módulos en la versión actual que se han cambiado 1
- ***Fa:*** Nro. de módulos en la versión actual que se han añadido 0
- ***Fe:*** Nro. de módulos en la versión actual que se han eliminado 0

El IMS está dada por,

$$\mathbf{IMS = [MT - (Fc + Fa + Fe)] / MT}$$
$$= [4 - (1 + 0 + 0)] / 4$$
$$= \mathbf{0.75}$$

Considerando la siguiente escala de valores, los resultados obtenidos tienen una estabilidad alta al final de la evolución en las versiones logradas.

Rango	Ponderación
75% y 100%	Optima
50% y 75%	Buena
25% y 50%	Suficiente
0% y 25%	Deficiente

Tabla 4.5 Ponderaciones de IMS
Fuente: [Elaboración Propia]

Por tanto, el sistema presenta el ***75%*** en cuanto a mantenimiento, es decir presenta una ponderación ***óptima*** para la realización de modificaciones y actualizaciiones necesarios.

MOVILIDAD

- ✓ Adaptabilidad, el sistema es fácil de adaptarse en otro entorno
- ✓ Instalabilidad, puede ser instalado en ambientes específicos.
- ✓ Reusabilidad, el sistema está estructurado por módulos, los cuales pueden ser acoplados a otros sistemas, sin tener mayores problemas.

Entonces, se tiene el siguiente cálculo.

Transición del producto = 1/ 3 (Portabilidad + Reusabilidad + Interoperabilidad)

= 1/3 (6,5 + 7,5 + 7)

= 7

Finalmente, los resultados alcanzados por el sistema en cuanto a calidad, según la ISO 9126, son presentados en la siguiente tabla.

Factor de Medida	% Obtenido
Punto Función	84%
Confiabilidad	81%
Usabilidad	85%
Mantenimiento	75%
Portabilidad	70 %

Tabla 4.6. Resultados alcanzados por el sistema
Fuente: [Elaboración Propia]

CAPITULO V
CONCLUSION Y RECOMENDACIONES

5.1. CONCLUSIONES

El Sistema de Información de Recursos Humanos (SIREHU), compuesto por los módulos; registro de personal, procesos organizacionales, control de personal y administración de usuarios, desarrollado para el Servicio Nacional del Sistema de Reparto (SENASIR), cumple con los objetivos planteados en el perfil del presente proyecto de grado. Cabe mencionar los puntos más relevantes respecto al sistema informático.

- ✓ Aporta al logro de una organización conformada por personal calificado, a través del seguimiento y control adecuado de la evaluación del personal, y además del registro actualizado, disponible y completo del 'file' del personal de planta y/o contrato de la institución a nivel nacional.
- ✓ Automatiza los procesos de control de personal, permitiendo a través de la estructura del sistema utilizar la tecnología de "Control Biométrico – Huella Digital", permitiendo de esta manera automatizar procesos como asistencias, atrasos, faltas, abandonos, permisos y/o licencias, refrigerios, generando sus correspondientes sanciones y/o llamadas de atención, considerando la definición de horarios y tolerancias.
- ✓ Automatiza del cálculo de vacaciones, según permisos y/o licencias emitidas, Revolucionando de esta forma el manejo de la información de manera ágil,

funcional, disminuyendo tiempos y aumentando la productividad del talento humano.

- ✓ Controla y hace el seguimiento adecuado de los movimientos laborales del funcionario, tanto vertical como horizontal, desde el momento de su ingreso hasta su retiro, llevando un seguimiento adecuado de los movimientos organizacionales; como designación, reasignación, retiro, renuncia, nuevos contratos, adendas y rescisiones. Permitiendo además el manejo de estructuras organizacionales (unidades, puestos, cargos, escalas salariales, ítems), generando de esta forma, información respecto a la línea de tiempo laboral del funcionario, estado de las unidades organizacionales; ocupadas o acéfalos y la conformación de personal en cada una de estas reparticiones.
- ✓ Controla los procesos de gestión de recursos humanos, validando en cada uno de los módulos del sistema, el cumplimiento de los lineamientos legales[4] que atribuyen a la administración de recursos humanos.
- ✓ Garantiza la seguridad lógica de la información almacenada en el sistema de información, permitiendo solo el acceso a usuarios autorizados, según niveles de perfil.
- ✓ Brinda reportes dinámicos, generación automática de Memorándums (en una interfaz Microsoft Word), exportación de datos a Microsoft Excel, según se requieran en cada módulo, para la toma de decisiones futuras, a nivel superior, operativo y ejecutivo. Ofreciendo disponibilidad de información al momento requerido.

Por tanto, SIREHU garantiza una eficiente y eficaz administración de recursos humanos. Coadyuvando al alcance de los objetivos del SENASIR, además ofrece una solución integral que automatiza procesos en la gestión de Recursos Humanos

5.2. RECOMENDACIONES

Es importante considerar las siguientes recomendaciones.

- ✓ El módulo de administración de usuarios, asigna una clave maestra por defecto, que solo el administrador conoce, es recomendable que al finalizar la instalación se realice la asignación de los perfiles de usuario correspondientes para los módulos.
- ✓ Si se cambia el uso de la tecnología de Control Biométrico para el control de asistencia del personal, por algún otro dispositivo, se debe considerar la estructura de la Base de Datos del sistema para su correcto funcionamiento.
- ✓ El módulo de Procesos Organizacionales puede ser completado con algunos otros requerimientos en cuanto a planillas por ejemplo.
- ✓ En el módulo de Registro de Personal, el proceso de registro de impuestos, declaraciones juradas, seguro medico, etc, pueden ser complementados y realizarse interfaces con otros sistemas referentes a estos procesos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. Chiavenato, A., (1993). *Administración de Recursos Humanos*. Ed. Segunda.

[2]. Misión y Visión del SENASIR. Recuperado el 5 de mayo de 2008, de http://www.senasir.org

[3]. Sabino C., (1993). Errores que más frecuentemente cometen los tesistas. *Como hacer una tesis*. Caracas: Panamericana

[4]. Sabino C., (1997). Planteamiento del problema. *Los caminos de la ciencia*. Bogotá: Panamericana.

[5]. Cursos de Marco Lógico – Banco Interamericano de Desarrollo. Recuperado el 12 de junio de 20008, de http:// www.bid.com.

[6]. Coad, P. y De Luca, J. (2004). *FDD - Feature-Driven Development*. Recuperado el 20 de junio 2008, de www.nebulon.com/fdd, de www.featuredrivendevelopment.com.

[7]. Sabino C., (2000). Análisis y síntesis de los resultados. *Procesos de la investigación*. Caraca: Panamericana.

[8]. Lewis G. (1994). DataPro. *What is Software Engineering*. (págs. 1-10)

[9]. Cota A. (1994). Soluciones Avanzadas. *Ingeniería de Software*. (pags. 5-13)

[10]. Jacobson. (1992). A Use Case Driven Aproach. En Adison Wesley. *Object-Oriented Software Engineering*. (págs. 465-493). U.S.A.: ACM Press.

[11]. Larman Craig (1999). UML y Patrones. (pags. 15 - 16) Mexico: 2da. Edición.

[12]. Coad P., Lefebvre E. y De Luca J. *Java Modeling In Color With UML. Enterprise Components and Process*. Prentice Hall.

[13]. Don Batory (2003). Proceedings of the 25th International Conference on Software Engineering. *A tutorial on Feature Oriented Programming and Product Lines*.

[14]. Palmer J. y Felsing C. (2002). *A practical Guide to Feature- Driven Development*.

[15]. Kenneth E., Kendall Julie E. (1997). *Análisis y Diseño de Sistemas*. USA: 3ra. Edición.

[16]. Roger S. Pressman (2002). *Ingeniería de Sistemas*. (págs. 173). España: 5ta Edición.

[17]. Jacobson, L. (1992) *Object – Oriented Software Engineering*, Addison – Wesley.

[18]. Roger S. Pressman (2002). *Diseño Orientado a Objetos*. (págs. 393). España: 5ta Edición.

[19]. Stevens, P. *Utilización de UML en ingeniería del software con objetos y componentes*.

[20]. Larman Craig (1999). *UML y Patrones*. (págs. 169) Mexico: 2da. Edición.

[21]. Roger S. Pressman (2002). *Diseño Orientado a Objetos* (págs. 393). España: 5ta Edición.

[22]. Davila A. *Tesis en calidad de Software*. Recuperado e 12 de septiembre de 2001, de http:www.desarrollossoftware.com

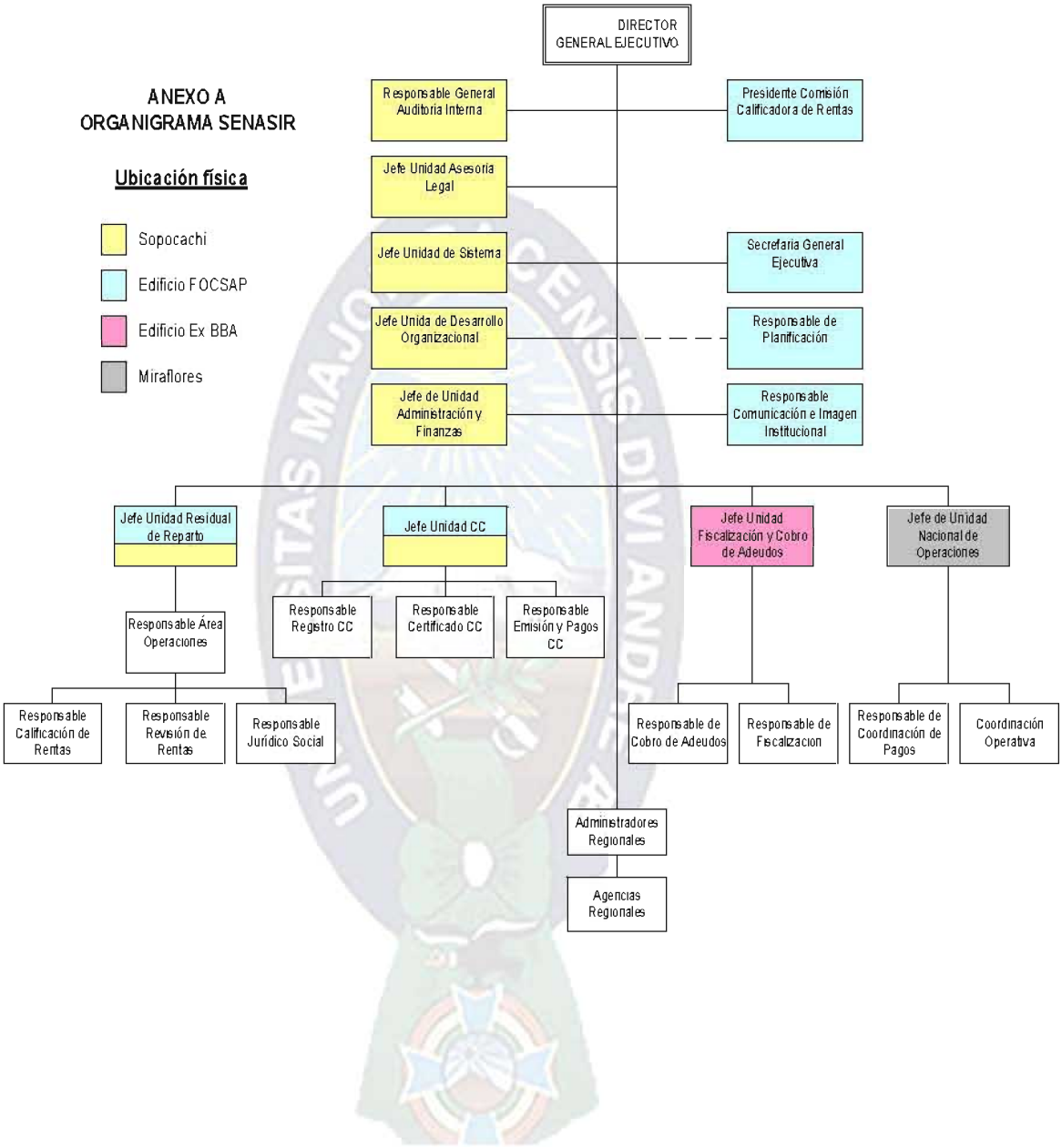
[23]. Pressman Roger. *Ingeniería de Software*. México 5ta. Edición

ANEXOS

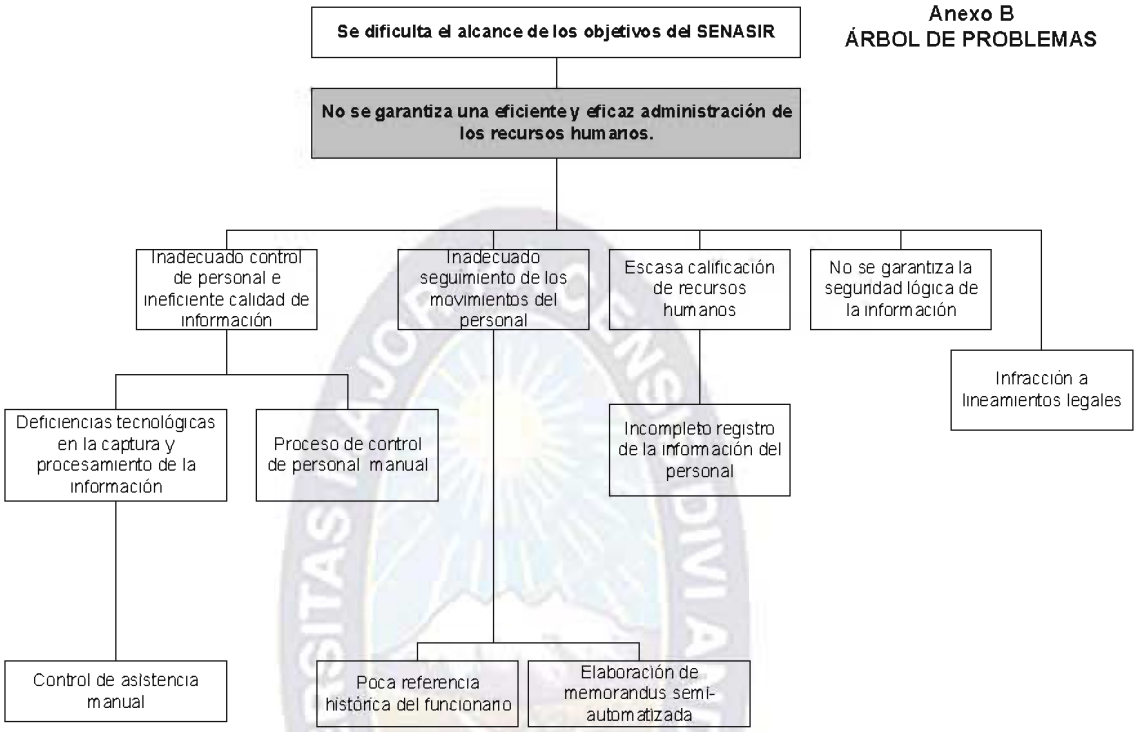
ANEXO A
ORGANIGRAMA SENASIR

Ubicación física

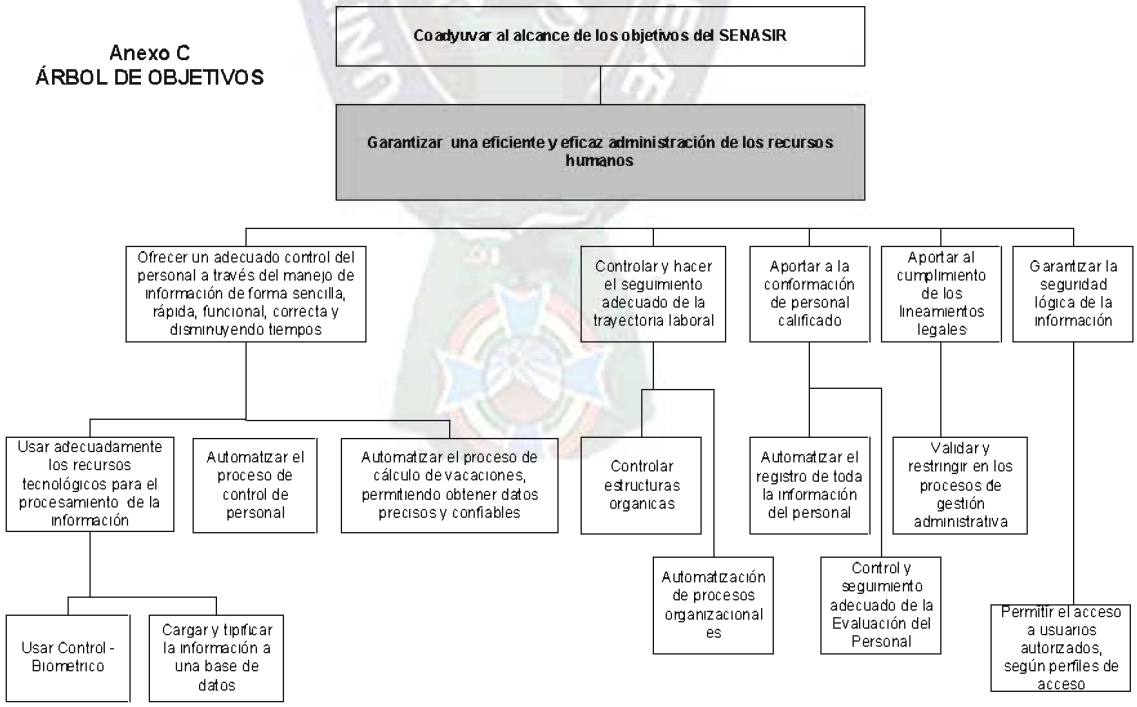
- Sopocachi
- Edificio FOCSAP
- Edificio Ex BBA
- Miraflores



Anexo B
ÁRBOL DE PROBLEMAS



Anexo C
ÁRBOL DE OBJETIVOS



ANEXO D

MATRIZ DE MARCO LÓGICO

RESUMEN NARRATIVO	INDICADORES OBJETIVAMENTE VERIFICABLES	MEDIOS OBJETIVAMENTE VERIFICABLES	SUPUESTOS
Fin: Coadyuvar al alcance de los objetivos del SENASIR	Incremento en un 90% en la eficiencia y eficacia de procesos administrativos y de control de recursos humanos.	La información resultante del sistema básicamente es confiable, legal, integra y disponible para toma de decisiones futuras.	Las verificaciones y evaluaciones de la información resultante del SIRH, son aceptadas de manera satisfactoria, a través de inspecciones visuales, encuestas, muestreo, etc.
Propósito: El desarrollo de SIREHU garantiza una eficiente y eficaz administración de recursos humanos.	Para Abril de 2009 Implementación del SIREHU, en las dependencias de la Unidad de Desarrollo Organizacional y Unidad de Sistemas del SENASIR.	Informes y evaluaciones del sistema.	Personal calificado para la manipulación del sistema y la disponibilidad de recursos tecnológicos necesarios.
Componentes: Adecuado control de personal mediante una solución integral que automatiza procesos, a través del Módulo de Control de Personal, revolucionando el manejo de la información de una manera sencilla, ágil y funcional, disminuyendo tiempos y aumentando la productividad del talento humano	Las tareas atribuidas a este Módulo de Control de Personal, son realizadas en tiempos menores y de manera eficiente.	Automatización de procesos de control de personal, cargado y tipificación de información a la Base de Datos.	Contar con los medios necesarios para el desarrollo e implementación del sistema. Contar con la carta de aprobación y certificación del SIREHU. Contar con la información y recursos necesarios para el diseño e implementación.

Proceso automatizado del cálculo de vacaciones, a través del Módulo Control de Personal, permitiendo obtener datos precisos y confiables.	El Módulo de Control de Personal, realiza el cómputo de vacaciones de manera satisfactoria, procesos referentes a control de asistencia, permitiendo tener toda la información requerida de manera oportuna y brindando información relevante para la institución.	Aplicación Windows (intranet) y automatización del proceso de cálculo de vacaciones.	
Organización conformada por personal calificado, a través del registro completo del personal de planta y/o contrato.	El sistema ofrece información completa, oportuna y actualizada del personal.	Implementación automatizada del Módulo Registro del Personal	
Control de los procesos de gestión de recursos humanos, validando el cumplimiento de los lineamientos legales que se atribuyen a la administración de recursos humanos.	El sistema coadyuva el cumplimiento de las leyes, normas y reglamentos que se atribuyen a la administración de recursos humanos en el SENASIR.	Implementación de validaciones y restricciones del sistema en cuanto a la ejecución de procesos administrativos y de control de personal	
Garantizada la seguridad lógica del SIREHU.	Se controlan adecuadamente el acceso al SIREHU y el tratamiento de datos de parte los usuarios y los administradores son registrados satisfactoriamente.	Aplicación de perfiles de usuario, a través de niveles de acceso.	
Reportes dinámicos, datos históricos que se requieren en cada módulo, para la toma de decisiones futuras, a nivel superior, operativo y ejecutivo. Ofreciendo disponibilidad de Información al momento requerido.	Las tomas de decisiones son más certeras, gracias a la información que provee el SIREHU.	Implementación de reportes en Crystal Report.	

Actividades			
Ingeniería de requerimientos	A través de entrevistas, cuestionarios y observaciones. Del 21 de octubre al 6 de noviembre del 2008	Recopilación de los documentos e información necesaria.	Información accesible y disponible.
Análisis del sistema	Análisis de toda la información recopilada, estructuración y organización de datos. Del 7 al 17 de noviembre.	Documentación sobre el análisis del sistema.	
Diseño del Sistema	Elaboración del diseño de la Base de Datos. Del 18 de noviembre al 23 de marzo de 2009.	Documentación del diseño del SIRH	
Implementación del módulo de Registro de Personal.	Programación del primer módulo, al 13 de enero de 2009.	Certificación del primer módulo concluido.	
Implementación del módulo de Proceso Organizacionales.	Programación del segundo módulo, al 16 de febrero de 2009.	Certificación del segundo módulo concluido.	
Implementación del módulo de Control de Personal	Programación del último módulo, al 23 de marzo del 2009.	Certificación del cumplimiento del sistema de información desarrollado.	
Implementación del módulo de Administración de usuarios.	Programación d este módulo, al 26 de marzo del 2009.		
Pruebas de funcionalidad e instalación del sistema.	Pruebas y correcciones funcionales del sistema, para su instalación, al 3 de abril de 2009.	Informe sobre pruebas e instalación.	