UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA
UNAN - MANAGUA
RECINTO UNIVERSITARIO RUBEN DARIO
FACULTAD DE CIENCIAS E INGENIERIA
DEPARTAMENTO DE COMPUTACION

Trabajo de Seminario para optar al título de Licenciatura en Ciencias de la Computación

TEMA:
"Automatización del Proceso de Auditoría a las Tecnologías de la información y las Comunicaciones."

Tutor: Msc. Alejandro Antonio Ruíz Segovia

Presentado por:

1. **Br. Lorgia Maytel Flores Mendoza**
2. **Br. Alejandro Salvador López Tapia**
3. **Br. Frank Alexander Mena Mejía**

Managua, Septiembre del 2010

Subtema:

“Sistema de administración de planes anuales y trabajos de auditorías.”

Resumen

Este proyecto consiste en plantear un modelo de administración de planes anuales y trabajos de auditorías para automatizarlo con la realización de un sistema informático.

A partir del modelo planteado, se desarrollo SIAPTRA (Sistema de Administración de Planes Anuales y Trabajos de Auditorías), cuyo principal objetivo es llevar un control detallado de planes anuales y trabajos de auditoría, es decir, los planes de auditorías que se realizan anualmente, el personal que las realizan, sus fases y actividades..

Las herramientas principales utilizadas para la elaboración de SIAPTRA son: Diagrama de flujo de datos (DFD), el Lenguaje Unificado de Modelado (UML) para modelar los diferentes comportamientos y atributos del sistema, Visual Basic.Net 2005 para la codificación del software, y SQL Server 2000 para administrar la base de datos; todas estas herramientas se mencionan y describen a lo largo de esta tesis.

AGRADECIMIENTO

Agradecemos primeramente a Dios por lo vida y la oportunidad de culminar y presentar este trabajo monográfico, que representa unos de nuestros más importantes logros en nuestras vidas.

Extendemos nuestra gratitud a Msc. Alejandro Antonio Ruiz Segovia por su asesoramiento y supervisión durante el desarrollo de esta tesis, a profesores u compañeros estudiantes por sus opiniones que directa e indirectamente influyeron en el contenido de este documento.

A nuestros padres y familiares por el apoyo incondicional que contribuyo en el desarrollo intelectual de nuestras vidas y motivo a concretar el trabajo que hoy presentamos.

Dedicamos este trabajo a nuestros padres, por sus enseñanzas que nos inculcaron, el respeto a la inteligencia, sabiduría y el valor del conocimiento. Por el soporte económico y moral que nos brindaron a lo largo de nuestros estudios pues con mucho esfuerzo y amor nos los concedieron.

ÍNDICE

Contenido ..... Pág.

1. Introducción 1
2. Objetivos 2
3. Justificación 3
4. Marco teórico 4

4.1. Análisis y Diseño de un Sistema de Información 5
4.1.1. Roles del analista de sistemas 5
4.1.2. Sistemas de información 6
4.1.3. Diseño de salida eficaz 8
4.1.4. Diseño de entrada eficaz 10

4.2. Proceso de desarrollo de sistema 11
4.2.1. Ciclo de vida del desarrollo de sistemas 11
4.2.2. Modelo de construcción de prototipos 15
4.2.3. Modelo DRA 17
4.2.4. Modelos evolutivos de proceso del software 19
4.2.5. Modelo de ensamblaje de componentes 22
4.2.6. Modelo de desarrollo concurrente 23
4.2.7. Modelo de métodos formales 24
4.2.8. Programación Extrema 25
4.2.9. El Proceso Unificado Rational 27

4.3. Base de datos 29
4.3.1. Modelos de bases de datos 29
4.3.1.1. Bases de datos jerárquicas 29
4.3.1.2. Base de datos de red 30
4.3.1.3. Base de datos relacional 30
4.3.2. Normalización 31
4.3.2.1. Claves 31
4.3.2.2. Formas Normales 32
4.3.2.2.1. Primera Forma Normal 32
4.3.2.2.2. Segunda Forma Normal 33
4.3.2.2.3. Tercera Forma Normal 34
4.3.2.2.4. Tercera Forma Normal BOYCE-CODD 35
4.3.2.2.5. Dependencias multivaluadas y la cuarta Forma Normal 36
4.3.2.2.6. Quinta Forma Normal 38

4.4. Diagrama de Flujo de datos 41
4.5. Lenguaje unificado de modelado UML 44
4.5.1. Diagramas de UML 44
4.5.1.1. Diagrama de Clases 45

4.5.1.2. Diagrama de Casos de Uso 47
4.5.1.3. Diagrama de Actividades 49
4.5.1.4. Diagrama de Secuencia 53
4.5.1.5. Diagrama de Colaboración 54
4.5.1.6. Diagrama de Componentes 55
4.5.1.7. Diagrama de Objetos 56
4.5.1.8. Diagrama de Estado 57
4.5.1.6. Diagrama de Distribución y Despliegue 58

4.6. Conceptos de Estudios de Factibilidad 60
4.6.1. Factibilidad Técnica 60
4.6.2. Factibilidad Operacional 60
4.6.3. Factibilidad Económica 61

4.7. Auditoría 62
4.7.1. Plan de Auditoria 63
4.7.2. Tipos de Auditoria 65

4.8. Herramientas de Software a utilizar 67
4.9. Pruebas de Software 72
4.9.1. Pruebas del sistema 73
4.9.1.1. Pruebas de integración 73
4.9.1.2. Pruebas de entregas 74
4.9.1.3. Pruebas de rendimiento 76
4.9.2. Pruebas del componentes 77
4.9.2.1. Pruebas de interfaces 77

5. Diseño Metodológico 79

5.1. Tipo de Estudio 79
5.2. Análisis y diseño del sistemas 79
5.3. Modelo de Auditoría 83
5.4. Estudios de Factibilidad 87
5.5. Pruebas del software 90

6. Conclusiones 96
7. Recomendaciones 97
8. Bibliografía 98
9. Anexos 100
9.1. Diagrama de flujos de datos 101
9.2. Modelo Unificado UML 103
9.2.1. Diagrama de clases 103
9.2.2. Diagrama de caso de uso 104
9.2.3. Diagrama de actividades 114
9.2.4. Diagrama de secuencia 119
9.3. Normalización de la base de datos 122

9.4. Base de Datos..... 131
9.5. Diccionario de datos..... 132
9.6. Pantallas 137
9.7. Manual de Usuario 144

1. Introducción

En la actualidad, muchas organizaciones van creciendo o aumentando su producción, lo que conlleva a la necesidad de evaluarse o evaluar sus productos para mejorar su desempeño y ser aun más competitivos. A causa de esto, las organizaciones realizan auditorías de manera periódica, para comprobar con base a evidencias, que se cumplan los objetivos establecidos de dicha organización.

La auditoría es sin duda el principal mecanismo para evaluar la administración financiera y organizacional de las instituciones y empresas, así como también evaluar la eficiencia en el funcionamiento de algunos objetos o su validez, por lo cual, es indispensable llevar control de estas, de modo que se logre dar seguimiento al estado y avance de cada una de las actividades que la integran, pero no solamente es necesario efectuar a cabalidad las actividades de una auditoria, sino también realizarla en el periodo de tiempo preestablecido, por lo cual la planificación es ineludible para cumplir efectivamente cada una de las actividades de las auditorías en el tiempo planificado.

La planificación se realiza dividiendo la auditoria en fases y cada una de las fases, a su vez, se componen de actividades que se asignan a los auditores (personal) que se encargan de realizarlas en una cantidad de horas especificas. Todas las auditorías se agrupan en un plan que se realiza cada año (plan anual), en el cual se lleva control de cada una de las auditorias.

Para mejorar dicho control, es indispensable la automatización de este modelo de Planificación, adoptando el uso de herramientas, como son, la implementación de Sistemas Informáticos que almacenen, procesen y generen información relevante sobre los planes de auditoría, para de esta forma, llevar un mejor seguimiento del avance de las auditorías y sus datos específicos.

2. Objetivos

- **Objetivos General**

  Diseñar un modelo para la administración de planes anuales y trabajos de auditoría y automatizarlo mediante el desarrollo de un sistema informático.

- **Objetivos específicos**

1. Crear un modelo para la administración de planes anuales de auditorías.

2. Automatizar mediante un sistema de información, el modelo de sistema administrativo de auditorías planteado.

3. Disminuir el tiempo en que se emiten los informes de costo y estado de las auditorías.

4. Mejorar el control de los datos de las diferentes auditorías planificadas, en progreso y finalizadas, así como el de las empresas, el personal y sus respectivas actividades.

3. Justificación

En estos días, la globalización de la información exige la modernización de las instituciones, lo que las obliga a adoptar la utilización de nuevas herramientas como son la automatización de sus procesos administrativos y organizativos.

Uno de los procesos administrativos que una organización podría automatizar, es el control de sus planes de auditorías, pero inconvenientemente muchas de estas organizaciones utilizan aplicaciones de software que no son eficientes para administrar dichos planes, además, el hecho de obtener un software diseñado con este propósito, exclusivamente para una organización, tendría un alto precio. Por este motivo, es primordial diseñar e implementar un sistema de software que sea apropiado para administrar estos planes de auditoría y a la vez sea de bajo costo, ya que este software funcionaría de forma estándar, para cualquier institución.

Con este Sistema de Información se pretende disminuir el tiempo y costo de fabricación, además de agilizar los procesos de registrar las auditorías con sus datos, y emitir reportes gráficos del avance de las autorías.

4. Marco teórico

En las últimas décadas se ha seguido la tendencia de automatizar de manera progresiva procesos productivos de todo tipo y en todas las áreas. Esta tendencia ha sido y sigue siendo posible gracias al desarrollo de la tecnología y su creciente demanda. Los objetivos que se persiguen bajo el uso de automatizar los sistemas de información es mejorar la calidad y productividad de los procesos, entregas de productos en tiempo preciso, reducir costo, aumentar eficiencia y eficacia, entre otros.

Un sistema de información se define como un conjunto organizado de elementos (personas, información, tecnología y actividades) que interactúan entre sí para recolectar, controlar, procesar, almacenar y divulgar información de una organización de manera interna o externa que ayuden a la toma decisiones y progreso de la organización. Dentro de estos conceptos de sistema cabe mencionar la definición de Sistemas automatizados que son sistemas creados por el hombre y controlados por computadoras, son capaces de responder de manera automática sin la intervención de algún operador ante algún cambio que se produzca, dando lugar a las acciones adecuadas para cumplir la función para la que ha sido diseñado.

4.1. Análisis y Diseño de un Sistema de Información

El análisis y diseño de sistemas, tiene el propósito de analizar sistemáticamente la entrada o el flujo de datos, procesar o transformar datos, el almacenamiento de datos y la salida de información en el contexto de una organización en particular. Más aún, el análisis de sistemas se emplea para analizar, diseñar e implementar mejoras en el funcionamiento de las organizaciones, a través de sistemas de información computarizados [Kendall & Kendall, 2005].

La finalidad del análisis está en comprender los detalles de una situación y decir si es deseable o factible una mejora y la finalidad del diseño es especificar las características del producto terminado.

Para el análisis y diseño de sistemas informáticos se debe seguir un modelo, aplicar técnicas y herramientas ya establecidas.

4.1.1. Roles del analista de sistemas

4.1.1.1. El analista de sistema

Para el desarrollo de una aplicación el analista de sistema debe evaluar de manera sistemática el funcionamiento de un negocio mediante el examen de la entrada y el procesamiento de datos y su consiguiente producción de información, con el propósito de mejorar los procesos de una organización [Kendall & Kendall, 2005].

Un analista de sistema debe tener numerosas cualidades, es un solucionador de problemas, administrador de personal, responsable, ético, disciplinado, entre otras.

El analista desempeña diversos roles, en ocasiones varios de ellos al mismo tiempo. Los tres roles principales del analista de sistemas son el de consultor, experto en soporte técnico y agente de cambio [Kendall & Kendall, 2005].

4.1.1.2. El analista como consultor

El analista debe recopilar toda la información necesaria para el desarrollo del sistema, aplicado técnicas y métodos ya establecidos como entrevista, encuestas, cuestionario, etc.

4.1.1.3. El analista como experto en soporte técnico

El analista debe contar con muchos conocimientos en computación para la implementación del sistema en la organización.

4.1.1.4. El analista como agente de cambio

En su calidad de analista de sistemas desempeñando la función de agente de cambio, debe promover un cambio que involucre el uso de los sistemas de información. También es parte de su tarea enseñar a los usuarios el proceso del cambio, ya que las modificaciones a un sistema de información no sólo afectan a éste sino que provocan cambios en el resto de la organización [Kendall & Kendall, 2005].

4.1.2. Sistemas de Información

Un sistema de información se define como un conjunto organizado de elementos (personas, información, tecnología y actividades) que interactúan entre sí para recolectar, controlar, procesar, almacenar y divulgar información de una organización de manera interna o externa que ayuden a la toma decisiones y progreso de la organización. Dentro de estos conceptos de sistema cabe mencionar la definición de Sistemas automatizados que son sistemas creados por el hombre y controlados por computadoras, son capaces de responder de manera automática sin la intervención de algún operador ante algún cambio que se produzca, dando lugar a las acciones adecuadas para cumplir la función para la que ha sido diseñado.

4.1.2.1. Sistema de procesamiento de transacciones

Los sistemas de procesamiento de transacciones (TPS, *Transaction Processing Systems)* son sistemas de información computarizada creados para procesar

grandes cantidades de datos relacionadas con transacciones rutinarias de negocios, como las nóminas y los inventarios [Kendall & Kendall, 2005].

4.1.2.2. Sistema de automatización de la oficina y sistema de trabajo del conocimiento

Los sistemas de automatización de la oficina apoyan a los trabajadores de datos con el propósito de transformar y manipular datos, entre los más comunes tenemos el procesamiento de texto, las hojas de cálculo, la autoedición, la calendarización electrónica entre otros. Los sistemas de trabajo del conocimiento sirven de apoyo a los trabajadores profesionales, como los científicos, ingenieros y médicos, en sus esfuerzos de creación de nuevo conocimientos.

4.1.2.3. Sistema de información gerencial

Los sistemas de información gerencial dan apoyo a un espectro de tareas organizacionales mucho más amplio que los sistemas de procesamiento de transacciones, como el análisis y la toma de decisiones. Para acceder a la información, los usuarios de un sistema de información gerencial comparten una base de datos común. Ésta almacena datos y modelos que ayudan al usuario a interpretar y aplicar los datos. Los sistemas de información gerencial producen información que se emplea en la toma de decisiones. Un sistema de información gerencial también puede contribuir a unificar algunas de las funciones de información computarizadas de una empresa, a pesar de que no existe como una estructura individual en ninguna parte de ésta [Kendall & Kendall, 2005].

4.1.2.4. Sistema de Apoyo a la toma de decisiones

Los sistemas de apoyo a la toma de decisiones constituyen una clase de alto nivel de sistemas de información computarizada. Coinciden con los sistemas de información gerencial en que ambos dependen de una base de datos para abastecerse de datos. Sin embargo, difieren en que los sistemas de apoyo a la toma de decisiones ponen énfasis en el apoyo a la toma de decisiones en todas sus fases, aunque la decisión definitiva es responsabilidad exclusiva del encargado de tomarla. Los sistemas de

apoyo a la toma de decisiones se ajustan más al gusto de la persona o grupo que los utiliza que a los sistemas de información gerencial tradicionales. En ocasiones se hace referencia a ellos como sistemas que se enfocan en la inteligencia de negocios [Kendall & Kendall, 2005].

4.1.2.5. Sistemas expertos e inteligencia artificial

La inteligencia artificial se puede considerar como el campo general para los sistemas expertos. La motivación principal de la inteligencia artificial ha sido desarrollar máquinas que tengan un comportamiento inteligente. Dos de las líneas de investigación son la comprensión del lenguaje natural y el análisis de la capacidad para razonar un problema hasta su conclusión lógica. Los sistemas expertos utilizan las técnicas de razonamiento de la inteligencia artificial para solucionar los problemas que les plantean los usuarios de negocios. Un sistema experto captura y utiliza el conocimiento de un experto para solucionar un problema específico en una organización [Kendall & Kendall, 2005].

4.1.3. Diseño de salida eficaz

La salida de información de los sistemas informáticos es un aspecto muy importante a tener en cuenta ya que muchas veces los usuarios juzgan a los sistemas por los resultados que estos generan.

La salida de datos puede ser presentada de muchas formas pero lo más importante es presentarla donde se necesita.

Debido a que una salida útil es esencial para asegurar el uso y aceptación del sistema de información, son varios los objetivos que el analista debe tener en mente al diseñarla [Kendall & Kendall, 2005].

4.1.3.1. Diseño de salida para satisfacer un propósito específico

Toda la salida debe tener un propósito. Durante la fase de determinación de los requerimientos de información, el analista de sistemas averigua qué propósitos se deben satisfacer. A continuación diseña la salida con base en esos propósitos [Kendall & Kendall, 2005].

4.1.3.2. Diseño de salida para satisfacer al usuario

La salida de datos debe satisfacer a los usuarios. En sistemas donde lo manipulan diferentes tipos de usuarios, la salida debe satisfacer las necesidades de cada usuario.

4.1.3.3. Entrega de la cantidad adecuada de salida

Hay usuarios que reciben mayor información que otros, esto irá de acuerdo a los propósito del sistema. Se debe tener cuidado con la sobrecarga de información que se dé a los usuarios.

4.1.3.4. Asegúrese de que la salida este donde se necesite

A menudo la salida se produce en un lugar (por ejemplo, en el departamento de procesamiento de datos) y después se distribuye al usuario. El aumento de la salida en línea, desplegada en pantalla, que se puede acceder de manera individual, ha reducido en parte el problema de la distribución, pero la distribución apropiada continúa como un objetivo primordial para el analista de sistemas. Para que sea usada y que sirva de algo, la salida se debe presentar al usuario correcto. No importa qué tan bien diseñados estén los informes, si no llegan a los tomadores de decisiones que los requieren, no tienen valor [Kendall & Kendall, 2005].

4.1.3.5. Suministro de la salida a tiempo

Una de las quejas más comunes de los usuarios es que no reciben la información a tiempo para tomar las decisiones necesarias. Aunque el tiempo no lo es todo, representa una parte importante de la utilidad que tendrá la salida para los tomadores

de decisiones. Muchos informes son requeridos en forma diaria, algunos sólo mensualmente, otros anualmente y otros pocos sólo de manera ocasional. El uso de salida bien publicada y basada en la Web también puede solucionar en parte el problema de la distribución a tiempo de la salida. La entrega a tiempo de la salida puede ser crucial para las operaciones de negocios [Kendall & Kendall, 2005].

4.1.4. Diseño de entrada eficaz

La calidad y eficacia de la salida está en dependencia de las entradas al sistema, por lo tanto es necesario que las pantallas y formas estén bien diseñadas, organizadas y estructuradas de la mejor manera para que el usuario no cometa muchos errores.

Para el diseño de entradas se toman en cuenta el tipo de datos que se ingresan al sistema, que medios se utilizan, la forma en que se deben disponer o como codificar los datos, la validación. Las decisiones de diseño para el manejo de entrada, especifican la forma en que serán aceptados los datos para su procesamiento por computadora.

4.1.4.1. Los lineamientos para el buen diseño de formas.

- Fáciles de llenar.
- Satisfacer el propósito para el cual fueron diseñadas.
- Asegurar su llenado preciso.
- Atractivas.

4.1.4.2. Los lineamientos para el buen diseño de pantallas.

- Mantenga la pantalla sencilla.
- Mantenga la presentación consistente.
- Facilite los movimientos del usuario entre pantalla.
- Crear pantallas atractivas.

4.2. Proceso de desarrollo de sistemas

Un proceso de desarrollo de sistema es una estrategia de desarrollo que contiene una serie de proceso, métodos y herramientas, dicho modelo se selecciona según la naturaleza del proyecto que se va a ejecutar, las herramientas, métodos y aplicación a utilizar. Se puede definir además como una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real [Sommerville, 2005].

4.2.1. Ciclo de vida del desarrollo de sistemas **[Kendall & Kendall, 2005]**

El método de ciclo de vida conocido también como modelo lineal o de cascada, es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información. El método del ciclo de vida para el desarrollo de sistemas consta de 7 fases:

4.2.1.1. Identificación de problemas, oportunidades y objetivos

En esta primera etapa del ciclo de desarrollo de los sistemas, el analista se involucra en la identificación de los problemas, de las oportunidades y de los objetivos. Esta fase es crucial para el éxito del resto del proyecto, pues nadie estará dispuesto a desperdiciar su tiempo dedicándolo al problema equivocado.

La primera etapa requiere que el analista observe de forma objetiva lo que ocurre en una empresa. Luego, en conjunto con los otros miembros de la organización hará notar los problemas. Las oportunidades son aquellas situaciones que el analista considera que pueden perfeccionarse mediante el uso de los sistemas de información computarizados. Al aprovechar las oportunidades, la empresa puede lograr una ventaja competitiva o llegar a establecer un estándar industrial.

La identificación de objetivos también es un componente importante de la primera fase. En un comienzo, el analista deberá descubrir lo que la empresa intenta realizar, y luego estará en posibilidad de determinar si el uso de los sistemas de información apoyaría a la empresa para alcanzar sus metas, el encaminarla a problemas u oportunidades específicas.

4.2.1.2. Determinación de los requerimientos de información

La siguiente etapa que aborda el analista, es la determinación de los requerimientos de información a partir de los usuarios particularmente involucrados. Para identificar los requerimientos de información dentro de la empresa, pueden utilizarse diversos instrumentos, los cuales incluyen: el muestreo, el estudio de los datos y formas usadas por la organización, la entrevista, los cuestionarios, la observación de la conducta de quien toma las decisiones, así como de su ambiente y también el desarrollo de prototipos.

En esta etapa el analista hace todo lo posible por identificar qué información requiere el usuario para desempeñar sus tareas. Puede ver, cómo varios de los métodos para establecer las necesidades de información, lo obligan a relacionarse directamente con los usuarios. Esta etapa sirve para elaborar la imagen que el analista tiene de la organización y de sus objetivos. En ocasiones, se llegan a concluir sólo las primeras dos etapas del ciclo de desarrollo de los sistemas. El analista es el especialista que emprende esta clase de estudios.

4.2.1.3. Análisis de las necesidades del sistema

La siguiente etapa que ejecuta el analista de sistemas consiste en analizar las necesidades propias del sistema. Una vez más, existen herramientas y técnicas especiales que facilitan al analista la realización de las determinaciones requeridas. Estas incluyen uso de diagramas que usan técnicas estructuradas para representar en forma gráfica la entrada de datos de la empresa, los procesos y la salida de la información.

Durante esta fase el analista de sistemas analiza también las decisiones estructuradas que se hayan tomado. Las decisiones estructuradas son aquellas en las cuales se pueden determinar las condiciones, las alternativas de condición, las acciones y las reglas de acción.

Existen tres métodos principales para el análisis de decisiones estructuradas: español estructurado, tablas y árboles de decisión.

A esta altura del ciclo de desarrollo del sistema, el analista prepara una propuesta del sistema que resume todo lo que ha encontrado, presenta un análisis costo / beneficio de las alternativas y plantea las recomendaciones (si es que existen) de lo que deberá realizarse. Si la dirección acepta alguna de las recomendaciones, el analista procederá de acuerdo con ella.

4.2.1.4. Diseño del sistema

En esta etapa del ciclo de desarrollo de los sistemas, el analista de sistemas usa la información que recolectó con anterioridad y elabora el diseño lógico del sistema de información. El analista diseña procedimientos precisos de captura de datos, con el fin de que los datos que se introducen al sistema sean los correctos. El analista también diseña accesos efectivos al sistema de información, mediante el uso de las técnicas de diseño de formularios y de pantallas.

Una parte del diseño lógico del sistema de información es el diseño de la interfaz con el usuario. La interfaz conecta al usuario con el sistema, y evidentemente, es de suma importancia. Serían ejemplos de interfaces para el usuario: el uso del teclado para introducir preguntas o respuestas, el uso de menús en la pantalla, con las opciones que tiene el usuario, el uso de dispositivos como el ratón (mouse) y muchos otros.

La etapa del diseño también incluye el diseño de los archivos o la base de datos que almacenará aquellos datos requeridos por quien toma las decisiones en la

organización. Una base de datos bien organizada es fundamental para cualquier sistema de información. En esta etapa, el analista diseña la salida (en pantalla o impresa) hacia el usuario, de acuerdo con sus necesidades de información.

4.2.1.5. Desarrollo y documentación del software

En esta etapa del ciclo de desarrollo de los sistemas, el analista trabaja con los programadores para desarrollar todo el software original que sea necesario. Aquí es donde, el analista de sistemas transmite al programador los requerimientos de programación.

Durante esta fase, el analista también colabora con los usuarios para desarrollar la documentación indispensable del software, incluyendo los manuales de procedimientos. La documentación le dirá al usuario como operar el software, y así también, qué hacer en caso de presentarse algún problema.

4.2.1.6. Pruebas y mantenimiento del sistema

El sistema de información debe probarse antes de ser utilizarlo. El costo es menor si se detectan los problemas antes de la entrega del sistema. El programador realiza algunas pruebas por su cuenta, otras se llevan a cabo en colaboración con el analista de sistemas. En un principio, se hace una serie de pruebas, con datos, para identificar las posibles fallas del sistema.

El mantenimiento del sistema y de su documentación empieza justamente en esta etapa, después esta función se realizará de forma rutinaria a lo largo de toda la vida del sistema. Las actividades de mantenimiento integran una buena parte de la rutina del programador, que para las empresas llegan a implicar importantes sumas de dinero. Sin embargo, el costo del mantenimiento disminuye de manera importante cuando el analista aplica procedimientos sistemáticos en el desarrollo de los sistemas.

4.2.1.7. Implementación y evaluación de sistema

En esta última etapa del desarrollo del sistema, el analista ayuda a implantar el sistema de información. Esto incluye el adiestramiento que el usuario requerirá. Si bien, parte de esta capacitación la dan las casas comerciales, la supervisión del adiestramiento es una responsabilidad del analista de sistemas. Más aún, el analista necesita planear la suave transición que trae consigo un cambio de sistemas.

Aunque la evaluación del sistema se plantea como parte integrante de la última etapa del ciclo de desarrollo de los sistemas; realmente, la evaluación toma parte en cada una de las etapas. Uno de los criterios fundamentales que debe satisfacerse, es que el futuro usuario utilice el sistema desarrollado.

4.2.2. Modelo de construcción de prototipos

Un cliente, a menudo, define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, de la capacidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre máquina. En estas y en otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.

El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades

del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

El objetivo del modelo de prototipo es la entrega rápida de sistemas que permita de manera satisfactoria definir cada vez después de una entrega los requisitos que ayuden a definir la funcionalidad final del producto, de tal manera que ayude a aclarar los puntos clave donde aun no estaban bien definidos los requerimientos, aunque este no sea totalmente funcional. Algunas ocasiones este desarrollo de sistema puede ser problemática por que el usuario ve el diseño como un sistema final y sus exigencias pueden ser no positivas al no ver un producto factible y de calidad, y esto puede llevar al desarrollador a determinar un tiempo no prudente para una próxima entrega de prototipo o para entregar el producto final.

Ventajas:

- No modifica el flujo del ciclo de vida.
- Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
- También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.
- Reduce costos y aumenta la probabilidad de éxito.
- Exige disponer de las herramientas adecuadas.
- No presenta calidad ni robustez.
- Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.

Desventajas:

- A los usuarios les gusta el sistema real y a los desarrolladores les gusta construir algo de inmediato. Sin embargo, la construcción de prototipos se torna problemática por las siguientes razones:

- El cliente ve funcionando lo que para él, es la primera versión del prototipo que ha sido construido con “chicle y cable para embalaje”, y puede decepcionarse al indicarle que el sistema aun no ha sido construido.
- El desarrollador puede caer en la tentación de aumentar el prototipo para construir el sistema final sin tener en cuenta las obligaciones de calidad y de mantenimiento que tiene con el cliente.

4.2.3. Modelo DRA

El Desarrollo Rápido de Aplicaciones (DRA) (Rapid Application Development RAD) es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. DRA se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de periodos cortos de tiempo. Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las siguientes fases:

4.2.3.1. Modelado de gestión

El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la proceso?

4.2.3.2. Modelado de datos

El flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.

4.2.3.3. Modelado de proceso

Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.

4.2.3.4. Generación de aplicaciones

El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

4.2.3.5. Pruebas de entrega

Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

Obviamente las limitaciones de tiempos impuestas en un proyecto DRA demandan ámbito en escalas. Si una aplicación de gestión puede modularse se forma que permita completarse cada una de las funciones principales en menos de tres meses (utilizando el enfoque descrito anteriormente), es un candidato del DRA. Cada una de

las funciones puede ser afrontada por un equipo DRA diferente y ser integradas en un solo conjunto.

Al igual que todos los modelos de proceso, el enfoque DRA tiene inconvenientes:

- Para proyectos grandes aunque por escalas, el DRA requiere recursos humanos suficientes como para crear el número correcto de equipos DRA.
- DRA requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso, por ninguna de las partes constituyentes, los proyectos DRA fracasaran.

No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modulizar adecuadamente. La construcción de los componentes necesarios para DRA será problemático. Si está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistema, el enfoque DRA puede que no funcione. DRA no es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el nuevo software requiere un alto grado de interoperabilidad con programas de computadora ya existentes.

DRA enfatiza el desarrollo de componentes de programas reutilizables. La reutilización es la piedra angular de las tecnologías de objetos, y se encuentra en el modelo de proceso de ensamblaje.

4.2.4. Modelos evolutivos de proceso del software

Los modelos evolutivos son iterativos. Se caracterizan por la forma en que permiten a los ingenieros del software desarrollar versiones cada vez más completas del software. Las exigencias de los clientes y las fechas de entrega obligan algunas ocasiones a la publicación de sistemas funcionales pero no completos, por lo que a veces se requiere una publicación póstuma del software con una versión más completa y mejorada.

4.2.4.1. Modelo incremental

Este modelo combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos. El modelo incremental aplica secuencias lineales de la misma forma en que progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software. El flujo del proceso de cualquier incremento puede incorporar la construcción de un prototipo.

Surge como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.

Durante el desarrollo de cada incremento se puede utilizar el modelo de cascada o evolutivo, dependiendo del conocimiento que se tenga sobre los requisitos a implementar. Si se tiene un buen conocimiento, se puede optar por cascada, si es dudoso, evolutivo.

Una de las ventajas de este modelo es que los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde el primer incremento. Los clientes pueden aclarar los requisitos que no tengan claros conforme ven las entregas del sistema, de esta manera se disminuye el riesgo de fracaso de todo proyecto, ya que se puede distribuir en cada incremento, pero esto atrae como desventaja que cada incremento debe ser pequeño para limitar el riesgo y aumentar la funcionalidad.

4.2.4.2. Modelo en espiral

El modelo de desarrollo en espiral es actualmente uno de los más conocidos y fue propuesto por Boehm. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Es un modelo de proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con aspectos controlados y sistemáticos del modelo lineal secuencial. Durante las primeras iteraciones la versión incremental podría ser un modelo en papel o prototipo. Durante las últimas iteraciones se producen versiones cada vez más complejas de ingeniería del sistema.
El modelo en espiral se divide en un número de actividades estructurales, también llamadas regiones de tareas:

Comunicación con el cliente: Tareas requeridas para establecer comunicación con el cliente.

Planificación: Tareas requeridas para definir los recursos, el tiempo y otras informaciones.

Análisis de riesgos: Tareas requeridas para evaluar riesgos técnicos y de gestión.

Ingeniería: tareas requeridas para construir una o más representaciones de la aplicación.

Construcción y adaptación: Las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.

Evaluación del cliente: Tareas requeridas para la evaluación del software por parte del cliente.

Cada región contiene tareas que se definen para lograr un nivel más alto de formalidad. El primer circuito de la espiral produce el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software.

El modelo en espiral puede ser adaptado para utilizarse a lo largo de toda la vida del software de computadora ya que su enfoque puede ser utilizado en un aspecto de mantenimiento.

4.2.5. Modelo de ensamblaje de componentes

Incorpora muchas de las características del Modelo Espiral. Es evolutivo por naturaleza y exige un enfoque interactivo para la creación del software. Sin embargo, el modelo ensamblador de componentes configura aplicaciones desde componentes separados del software (algunas veces llamados "clases").

Esto se debe gracias a que, si se diseñan y se implementan adecuadamente, las clases orientadas a objetos son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadoras.

En primer lugar se identifica las clases candidatas examinando los datos que se van a manejar por parte de la aplicación y el algoritmo que se va a crear para conseguir el tratamiento. Si estas clases han sido creadas por programas anteriores se almacenan en una biblioteca de clases o depósito. Se determina cuáles de ellas ya existen a fin de ser reutilizadas. En caso de que exista alguna que no esté diseñada, se aplican los métodos orientados a objetos. Este proceso se inicia en el estado de Análisis de Riesgos del Espiral y se inserta en el estado de Construcción de Ingeniería.

En resumen, este modelo se basa en ir construyendo con la construcción de cada sistema una biblioteca de Componentes (clases /objetos) , cuando se va a construir un nuevo sistema, se hace el proceso de definir los objetos del sistema, buscar en la librería de objetos, construir los que no existen, meterlos en la biblioteca, ensamblar los objetos, la metodología busca que sea evolutiva pasando por una fase de planificación, análisis de riesgos, ingeniería, construcción y adaptación, evaluación del cliente y repetir estas fases de tal forma que las primeras iteraciones desarrollan los conceptos, al avanzar se desarrollan los nuevos componentes, luego se busca mejorarlos y finalmente se les da mantenimiento [Pressman, 2002].

4.2.6. Modelo de desarrollo concurrente

El modelo de proceso concurrente se puede representar en forma de esquema como una serie de actividades técnicas importantes, tareas y estados asociados a ellas.

El modelo de proceso concurrente define una serie de acontecimientos que dispararán transiciones de estado a estado para cada una de las actividades. Durante las primeras etapas del diseño, no se contempla una inconsistencia del modelo de análisis. Esto genera la corrección del modelo de análisis de sucesos, que disparará la actividad de análisis del estado hecho al estado cambios en espera.

El modelo de proceso concurrente se utiliza como paradigma de desarrollo de aplicaciones cliente/servidor, que cuando se aplica, el modelo de proceso concurrente define actividades en dos dimensiones: una dimensión de sistemas y una dimensión de componentes. Los aspectos del nivel de sistemas se afrontan mediante tres actividades: diseño, ensamblaje y uso.

La dimensión de componentes se afronta con dos actividades: diseño y realización. La concurrencia se logra de dos formas: las actividades de sistema y de componentes ocurren simultáneamente y pueden modelarse con el enfoque orientado a objetos; y una aplicación cliente/servidor típica se implementa con muchos componentes, cada uno de los cuales se pueden diseñar y realizar concurrentemente. En realidad, el modelo de proceso concurrente es aplicable a todo tipo de desarrollo de software y proporciona una imagen exacta del estado actual de un proyecto [Pressman, 2002].

4.2.7. Modelo de métodos formales

El modelo de métodos formales comprende un conjunto de actividades que conducen a la especificación matemática del software de computadora. Los métodos formales permiten que un ingeniero de software especifique, desarrolle y verifique un sistema basado en computadora aplicando una notación rigurosa y matemática.

Algunas organizaciones de desarrollo del software actualmente aplican una variación de este enfoque, llamado ingeniería del software de sala limpia. Cuando se utilizan métodos formales durante el desarrollo, proporcionan un mecanismo para eliminar muchos de los problemas que son difíciles de superar con paradigmas de la ingeniería del software. La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen más fácilmente, no mediante revisión, sino mediante la aplicación del análisis matemático. Cuando se utilizan métodos formales durante el diseño, sirven como base para la verificación de programas y por consiguiente permiten que el ingeniero del software descubra y corrija errores que no se pudieron detectar de otra manera.

Los modelos de métodos formales ofrecen la promesa de un software libre de defectos. Sin embargo, se ha hablado de una gran preocupación sobre su aplicabilidad en un entorno de gestión. Algunas de las dificultades que presenta este método es:

- Es bastante caro y lleva mucho tiempo
- Pocos desarrolladores de software tienen los conocimientos necesarios
- Es difícil utilizar los modelos como mecanismo de comunicación con los clientes que no tienen muchos conocimientos técnicos [Pressman, 2002].

4.2.8. Programación extrema

La programación extrema es una técnica muy conocida, se desarrolla de manera iterativa y con la participación activa del cliente.

En la programación extrema, todos los requerimientos se expresan como escenarios (llamados historias de usuario), los cuales se implementan directamente como una serie de tareas. Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Todas las pruebas se deben ejecutar satisfactoriamente cuando el código nuevo se integre al sistema. Existe un pequeño espacio de tiempo entre las entregas del sistema [Pressman, 2002].

La Figura 1 ilustra el proceso de la XP para producir un incremento del sistema que se está desarrollando.

La programación extrema sigue los siguientes pasos:

1. Está basada en las historias de clientes. Una vez que se toma en cuenta la opinión del cliente se realizan pequeñas entregas.
2. El cliente es el agente principal en las pruebas y desarrollo, por lo que se requiere un tiempo prudencial de él en las participaciones al desarrollarse el sistema.
3. Se apunta a trabajar en pareja, tomando en cuenta la opinión de las personas en vez de los de proceso.
4. El desarrollo se va logrando por medio de pequeñas entregas a las que se le han realizado las pruebas correspondientes.
5. El mantenimiento de la simplicidad se lleva a cabo a través de la refactorización constante para mejorar la calidad del código y la utilización de diseños sencillos que no pre-ven cambios futuros en el sistema.

Figura 1. El ciclo de entrega en la programación extrema.

En XP, las pruebas de aceptación, como el desarrollo, son incrementales. El desarrollo previamente probado y el uso de bancos de pruebas automatizados son las principales virtudes del enfoque de la XP. En vez de escribir el código del programa y luego las pruebas de ese código, el desarrollo previamente probado significa que las pruebas se escriben antes que el código.

Fundamentalmente, las pruebas se escriben como un componente ejecutable antes de que se implemente la tarea. Una vez que se ha implementado el software, se pueden ejecutar las pruebas inmediatamente. Este componente de pruebas debe ser una aplicación independiente, debe simular el envío de la entrada a probar y debe verificar que el resultado cumple la especificación de salida.

Con el desarrollo previamente probado, siempre hay un conjunto de pruebas que se pueden ejecutar fácilmente y de forma rápida. Esto significa que siempre que se añada cualquier funcionalidad al sistema, se pueden ejecutar las pruebas y detectar inmediatamente los problemas que el código nuevo haya introducido.

Contar con el cliente para el apoyo al desarrollo de las pruebas de aceptación es a veces un problema serio en el proceso de pruebas de la XP. Las personas que adoptan el papel de cliente tienen muy poco tiempo disponible y es posible que no puedan trabajar a tiempo complete con el equipo de desarrollo. El cliente puede pensar que proporcionar los requerimientos es contribución suficiente y puede ser reaccionar a participar en el proceso de pruebas [Pressman, 2002].

4.2.9. El Proceso Unificado Rational

Racional Unified Process (RUP), es un modelo de proceso que está basado en el trabajo de UML y el asociado Proceso Unificado de Desarrollo de Software [Sommerville, 2005]. Rational es un modelo que comprende cuatro fases de desarrollo, las cuales son:

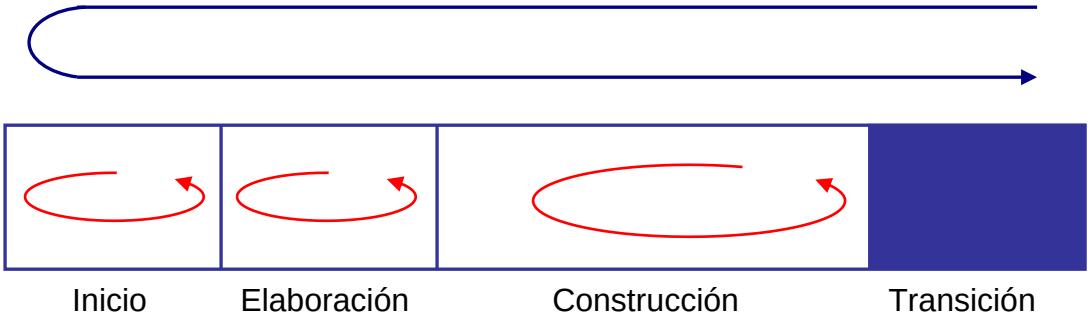


Figura 2. Fases de desarrollo de Rational

- Inicio: Se establece un caso de negocio para el sistema, se identifican los usuarios que interactuarán con el sistema, ya sea personas u otros sistemas. Con esta información se valora la aportación que hace el sistema al negocio; si la aportación es de poca importancia, se puede desechar la implementación del Proyecto.

- Elaboración: la importancia de esta fase, es comprender el dominio del problema, establecer un marco de trabajo arquitectónico para la elaboración del sistema e identificar los riesgos existentes del proyecto. En esta fase se definen los requerimientos del sistema mediante la utilización de casos de uso en UML.

- Construcción: Se diseña el sistema, se elaboran las líneas de código y se prueba. En la conclusión de esta fase, el sistema de software debe ser

operativo, y la documentación del sistema, lista para ser entregada a los usuarios.

- Transición: es la última fase, en la que el sistema, una vez concluido se instala en el lugar donde será utilizado por los usuarios. Al final de esta fase, el sistema debe estar operando correctamente en el entorno para el que fue diseñado y la documentación de software, en manos de los usuarios.

Las ventajas que ofrece rational son el desarrollo de software de forma iterativa, gestión de requerimientos, utilización de arquitecturas basadas en componentes, modelaje de software visualmente, verifica la calidad del software y controla los cambios del software. Como desventaja, RUP no es apropiado para todos los tipos de desarrollo de sistema ya que representa un de proceso de desarrollo genérico.

4.3. Base de datos

Una base de datos en una colección de datos almacenados de tal forma que sean accesibles para su manipulación. Contiene la información de manera precisa y consistente, fácil de obtener y de compartir.

Los objetivos de efectividad de la base de datos incluyen [Kendall & Kendall, 2005]:

- Asegurarse de que la base de datos pueda ser compartida entre los usuarios de una diversidad de aplicaciones.
- Mantener datos que sean precisos y consistentes.
- Asegurarse de que todos los datos requeridos para las aplicaciones actuales y futuras estén fácilmente disponibles.
- Permitir que la base de datos evolucione y que las necesidades de los usuarios crezcan.
- Permitir que los usuarios construyan su vista personal de los datos sin preocuparse de la forma en que estén físicamente guardados los datos.

4.3.1. Modelos de bases de datos

Un modelo de dato define como una descripción o contener de datos en donde se almacena información, así como de los métodos para la recuperación de almacenamiento de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos*; por lo general se refieren a algoritmos y conceptos matemáticos.

4.3.1.1. Bases de datos jerárquicas

Almacena su información en una estructura jerárquica, se organiza de la misma forma que un árbol, solo que de forma al revés. El nodo padre contiene uno o más hijos, el nodo que no tiene padre es considerado el nodo raíz, o el inicio de la estructura jerárquico, y aquellos que no tienen hijos son llamados hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Los principales inconvenientes que tiene es: Un segmento hijo no puede tener más de un padre, no se permiten más de una relación entre dos segmentos, para acceder a cualquier segmento es necesario comenzar por el segmento raíz , el árbol se debe de recorrer en el orden designado.

4.3.1.2. Base de datos de red

Su estructura es similar a la jerárquica, pero a diferencia de ella, un nodo hijo puede poseer más de un padre, es decir cualquier componente puede relacionarse con cualquier otro.

Es una gran mejora con respecto al modelo jerárquico, ya que ofrece una solución eficiente al problema de redundancia de datos; pero, aun así, resulta ser un poco difícil para ser usado por usuarios finales.

4.3.1.3. Base de datos relacional

Es actualmente el modelo de base de datos más en la actualidad para modelar problemas reales y administrar datos dinámicamente. Se fundamenta en el uso de relaciones, que se consideran en forma lógica como conjunto de datos. Hace uso de tablas que contiene registros representados por filas y campos representados en columnas.

En la aplicación de este modelo el almacenamiento de datos no tiene ninguna relevancia, es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Durante el proceso de este modelo se aplica el proceso de normalización a las tablas a utilizar de tal forma que los campos queden de tal forma que logren una relación en la que mínese al máximo la redundancia y aumente la integridad de los datos.

4.3.2. Normalización

El proceso de normalización de bases de datos consiste en aplicar, elaborar y mejorar por medio de una serie de reglas las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

1- Evitar la redundancia de los datos.
2- Evitar problemas de actualización de los datos en las tablas.
3- Proteger la integridad de los datos.

En el modelo relacional es frecuente llamar tabla a una relación, aunque para que una tabla sea considerada como una relación tiene que cumplir con algunas restricciones:

1.Cada columna debe tener su nombre único.

2. No puede haber dos filas iguales.

3. No se permiten los duplicados. Todos los datos en una columna deben ser del mismo tipo.

El proceso de normalización se compone de una serie de seis etapas llamadas formas normales [González Alvarado, 1996], las que tienen como objetivo encontrar una serie de relaciones que tendrían un comportamiento exento de problemas, el proceso pretende llevar a que las relaciones presentes en la base de datos se encuentren todas en quinta forma normal (5FN).

4.3.2.1. Claves

Las tablas están compuestas de un conjunto de atributos que la integran y que son únicos. Una clave es un atributo que identifica de manera única a una fila. La clave primaria determina a los demás atributos de una tabla. La clave puede tomar

cualquier valor, aunque generalmente tienden a ser valores alfanuméricos. En una tabla puede haber más de una clave o llave, en tal caso a esto se le denomina clave compuesta. Cuando se quiere escoger dentro de los atributos a uno de ellos para ser la clave primaria se les denomina claves candidatas. Además es la posible clave primaria.

Una clave foránea es aquella columna que existiendo como dependiente en una tabla, es a su vez clave primaria en otra tabla.

4.3.2.2. Formas normales

4.3.2.2.1. Primera forma normal (1FN)

La primera forma normal (1FN) se refiere a la representación de una relación, en la cual los atributos son diferentes y los valores de cada uno de esos atributos son diferentes y los valores de cada uno son componentes atómicos [González Alvarado, 1996].

Una tabla esta en primera forma normal si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos.
- La tabla contiene una clave primaria.
- La tabla no contiene atributos nulos.
- Si no posee ciclos repetitivos.

Una columna no puede tener múltiples valores. Los datos son atómicos. (Si a cada valor de X le pertenece un valor de Y, entonces a cada valor de Y le pertenece un valor de X).

Esta forma normal elimina los valores repetidos dentro de una BD.

Por ejemplo:

TOUR

Numero - viaje	Sitio
06-96	(Irazú, Volcán)
04-96	(Valle de la muerte, Desierto)
05-96	(Ngorongoro, Volcán)

a). Ejemplo de relación que no se encuentra en 1FN

TOUR

Numero – Viaje	Nombre- sitio	Tipo-Sitio
06-96	Irazú	Volcán
04-96	Valle de la muerte	Desierto
05-96	Ngorongoro	Volcán

b). Ejemplo de una relación en 1FN

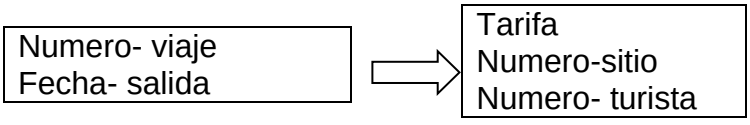
Figura 3. Primera forma normal

4.3.2.2.2. Segunda forma normal (2FN)

Antes de introducir el concepto de segunda forma normal se requiere conocer lo que significa dependencia parcial. En efecto, sea un esquema de relación **R** y A un atributo no llave. Se dice que el atributo A depende parcialmente de X si se verifica $Y \rightarrow A$, en donde Y es un subconjunto propio de la llave.

Así, una relación **R** se dice que se encuentra en segunda forma normal (2FN), si se encuentra en 1FN y si ningún atributo no llave depende parcialmente de la llave primaria.

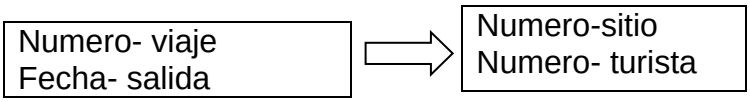
Itinerario
Llave primaria



a). Ejemplo de una relación que no está en 2FN

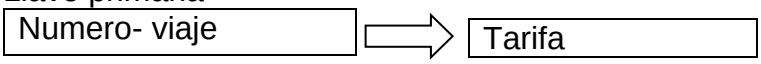
Itinerario

Llave primaria



Costo itinerario

Llave primaria



b). Ejemplo de relaciones en 2FN

Figura 4. Segunda forma normal

4.3.2.2.3. Tercera forma normal (3FN)

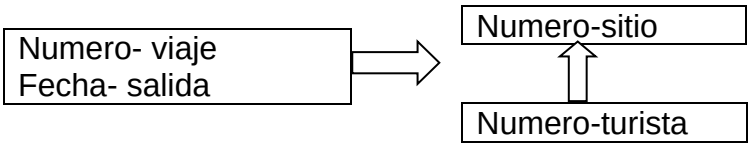
La tabla se encuentra en 3FN si esta 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente, de la clave primaria.

Sea **R(X, Y, Z)** un esquema de relación en X, Y, Z son subconjuntos de atributos. Se dice que Z es transitivamente dependiente de X si existe Y tal que se dan las siguientes condiciones:

- Se verifica X→Y
- No se verifica Y → X
- Se verifica Y→ Z

Itinerario

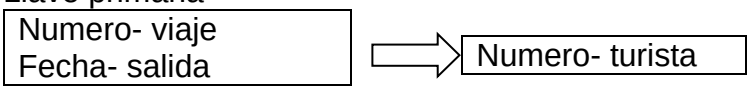
Llave primaria



a). Ejemplo de relación que no está en 3FN

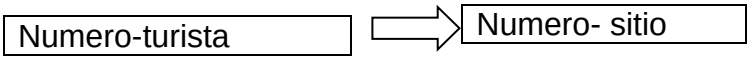
Itinerario2

Llave primaria



Sitio-visitado

Llave primaria



b). Ejemplo de relación en 3FN

Figura 5. Tercera Forma Normal 3FN

4.3.2.2.4. Tercera Forma Normal BOYCE – CODD (3FNBC)

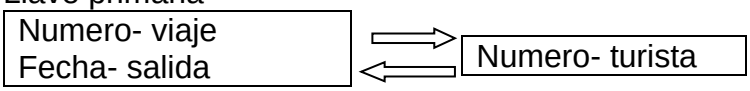
Una relación se encuentra en tercera forma normal Boyce - Codd (3FNBC) si todos los atributos son determinados solo por llaves.

$X \rightarrow A$, $A \notin X$.

Se verifica en **R** entonces X contiene una llave **R.**

Itinerario2

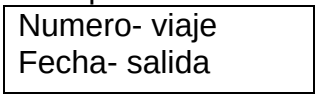
Llave primaria



a). Ejemplo no está en 3FNBC

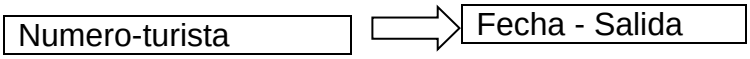
Itinerario3

Llave primaria



Salida – turista

Llave primaria



b). Ejemplo de relaciones esta 3FNBC

Figura 6. Tercera Forma Normal 3FNBC

4.3.2.2.5. Dependencias multivaluadas y la cuarta forma normal (4FN)

Existe dependencia funcional multivalorada o de múltiples valores si, dados tres atributos de una tabla, si para cada valor del primer atributo existen múltiples valores en el segundo atributo y no hay ninguna relación entre el tercer atributo y el primero, a no ser a través del segundo atributo.

Una tabla está en Cuarta Forma Normal o 4FN si está en 3FNBC y las únicas dependencias funcionales multivaloradas que existen son las dependencias funcionales de la clave con los atributos que no forman parte de la misma. Estas dependencias multievaluadas de la clave con los atributos que no forman parte de la misma son dependencias triviales.

Supongamos que los atributos de la tabla transporte son conductor, tipo de vehículo y tipo de carga, formando los tres campos la clave primaria. A cada conductor se le puede asignar un vehículo u otro y cada vehículo puede transportar varios tipos de carga ver Figura 6.a.

Con estas condiciones, los conductores son independientes de la carga; el tipo de vehículos depende del conductor y el tipo de vehículo depende de la carga. En este caso hay dependencias funcionales multivaloradas, ya que algunos atributos que forman la clave dependen de otro atributo que también la forman.

Para conseguir que esta tabla esté en 4FN se necesita crear dos nuevas tablas (ver Figura 6.b) en lugar de la tabla actual, manteniéndose en cada una de ellas una

dependencia múltiple. La primera tabla tendrá los atributos conductor y tipo de vehículo y la segunda, tipo de vehículo y tipo de carga. De este modo la tabla en 4FN debido a que la clave primaria de ambas tablas son todos los campos que la forman. Resultado:

Transporte

Conductor	**Tipo Vehículo**	**Tipo Carga**
Juan	Furgoneta	Perecederos
Marcos	Furgoneta	Perecederos
Juan	Furgoneta	Muebles
Marcos	Furgoneta	Muebles
Juan	Camión	Mudanza
Marcos	Camión	Mudanza

a). Ejemplo no está en cuarta forma normal

Vehículo-carga

Tipo Vehículo	**Tipo Carga**
Furgoneta	Perecederos
Furgoneta	Muebles
Camión	Mudanza

Conductor - vehículo

Conductor	**Tipo Vehículo**
Juan	Furgoneta
Marcos	Furgoneta
Juan	Furgoneta
Marcos	Furgoneta
Juan	Camión
Marcos	Camión

b). Ejemplo esta en 4FN

Figura 7. Cuarta forma normal 4FN

4.3.2.2.6. Quinta forma normal 5FN

Se dice que hay dependencia de JOIN, de unión o de producto si una tabla tiene dependencia de unión con varias de sus proyecciones y se puede obtener la tabla por medio de la unión de dichas proyecciones.

Proyección

Creación de una tabla cuyos elementos forman un subconjunto de una tabla dada. Se incluyen todas las filas y algunas columnas.

Unión

Formar, a partir de dos tablas, una nueva con todos los campos de una de ellas y los registros de ambas, excepto los repetidos. Ambas tablas han de tener el mismo grado y las mismas columnas.

Una tabla esta en Quinta Forma Normal (5FN) o Forma Normal de Proyección-Unión si está en 4FN y las únicas dependencias que existen son las dependencias de unión de una tabla con sus proyecciones relacionándose entre las distintas proyecciones mediante la clave primaria o cualquier clave alternativa. La 5FN se emplea cuando en una misma tabla tenemos mucha información redundante, con pocos atributos o cuando una tabla posee una gran cantidad de atributos y se hace por ello inmanejable.

Para conseguir que una tabla 4FN con gran cantidad de atributos esté en 5FN, se parte la tabla original en tantas tablas como se desee, teniendo cada una de ellas en común con las demás los campos que forman la clave primaria en la tabla original.

Si se tiene una tabla de préstamo de libros de una biblioteca, con los atributos título, fecha de préstamo y número de socios que ha tomado prestado el libro, (ver Figura 7.a existen multitud de registros que se crean diariamente en esa tabla, pero para cada libro o para cada socio habrá pocos registros, con lo que una consulta para esa tabla como: ¿Cuáles son los libros leídos por un determinado socio?, puede tener

una velocidad de respuesta elevada. Si esta tabla se parte en las tablas título-fecha, título-socio y socio-fecha, cualquier consulta similar a la anterior tendrá un tiempo de respuesta tolerable, y cuando sea necesario, se podrán realizar consultas que impliquen los datos de las tres tablas.

Biblioteca

Título	**Fecha**	**Socio**
T1	FT	S1
T2	FU	S2
T3	FV	S1
T4	FG	S4
T1	FH	S3
T2	FT	S4
T3	FV	S3

a). Ejemplo no está en 5FN

Título-Fecha

Título	**Fecha**
T1	FT
T2	FU
T3	FV
T4	FG
T1	FH
T2	FT
T3	FV

Título-Socio

Título	**Socio**
T1	S1
T2	S2
T3	S1
T4	S4
T1	S3
T2	S4
T3	S3

Fecha-Socio

Fecha	Socio
FT	S1
FU	S2
FV	S1
FG	S4
FH	S3
FT	S4
FV	S3

b). Ejemplo esta en 5FN

Figura 8. Ejemplo de 5FN

4.4. Diagramas de Flujo

Los diagramas de flujo de datos permiten al analista de sistema elaborar representaciones gráficas que modelan el flujo de información de una organización, lo que permite verificar las entradas, procesos, salidas y almacenes de datos.

Los DFD muestran cómo será construido el sistema y proporcionan documentación firme a este.

Elementos de un diagrama de flujo de datos:

- **Entidades** ya sea una persona, una maquina u otro negocio que envía o recibe datos del sistema, pueden ser graficados como rectángulos.

- **Procesos** puede ser graficados como rectángulos con esquinas redondeadas y par de líneas que lo atraviesan de forma horizontal, y representa transformación de datos, es decir en donde se reciben datos para ser trabajados y luego entregados.

- **Almacén de datos** puede ser graficados como rectángulos con una línea incompleta y otra que la corta de forma vertical, y representan archivos lógicos donde se guardan o extraen datos.

- **Flujos de datos** son flechas punteadas que indica que se envían o reciben datos de una entidad, proceso o almacén.

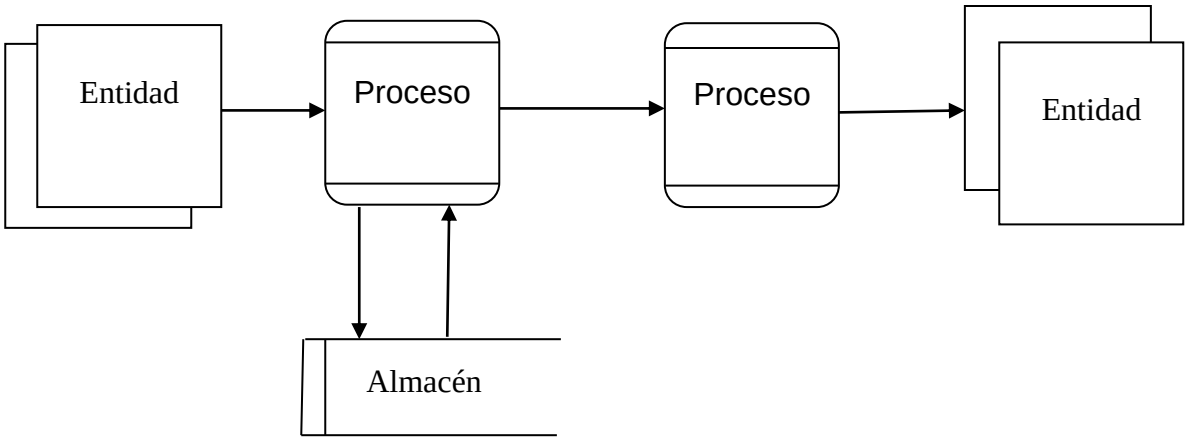


Figura 9. Diagrama de flujo de datos

Para un mejor entendimiento el diagrama de flujo de datos se divide en tres niveles:

- Diagrama de flujo de nivel de contexto
- Diagrama de flujo de nivel cero
- Diagrama de flujo de nivel 1

4.4.1. Diagrama de flujo de nivel contexto

Este diagrama es el de más alto nivel y solamente se realizará un proceso, enumerado con el número cero representará el sistema completo a elaborar por lo que se que le debe indicar también el nombre completo del sistema, en este diagrama se verifican las principales entradas y salidas de datos y las entidades que interactúan con dicho sistema.

4.4.2. Diagrama de flujo de nivel cero

Este diagrama es más detallado, este debe mostrar los principales procesos de la organización relacionados con las entidades y almacenes datos que se verán involucrados. Cabe destacar cada proceso debe estar enumerado iniciando por el numero uno y así sucesivamente.

4.4.3. Diagrama de flujo de nivel 1:

Este diagrama brindará una información más precisa debido a que se realizara un diagrama a partir de cada proceso del diagrama de nivel de cero. Así se tendrá una idea clara de cada proceso que se realiza en la organización.

4.5. Lenguaje Unificado de Modelado UML

El Lenguaje unificado de modelado (UML) permite especificar, visualizar, construir y crear un esquema previo del sistema que se va a desarrollar, captando de manera estática y dinámica el funcionamiento de todo el software.

El objetivo de UML es modelar por medio de diagramas el comportamiento del sistema y que finalmente realizara un usuario externo, permite una abstracción visual del sistema y sus componentes.

UML es apto para cualquier tipo de lenguaje orientado a objetos, pues permite la adaptación de modelar de forma sencilla el funcionamiento previo del sistema. Antes de iniciar a modelar un sistema se debe definir el universo sobre el que se va a modelar, así como especificar los objetos y la función que va a desarrollar este dentro del sistema.

UML utiliza diferentes diagramas para modelar el sistema, los diagramas son la representación de las todas las actividades y estructura del sistema. Su utilización es independiente del lenguaje de programación y de las características de los proyectos ya que ha sido diseñado para modelar.

4.5.1. Diagramas de UML

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema [Schmuller, 1999].

Los tipos de diagramas que se utilizan para construir modelos son:

- Diagrama de Clases
- Diagrama de Casos de Uso
- Diagrama de Actividades
- Diagrama de Secuencias
- Diagrama de Componentes
- Diagrama de objetos
- Diagrama de estado
- Diagrama de colaboración
- Diagrama de despliegue

Es importante señalar que en un modelo UML no es necesario que aparezcan todos los diagramas, debido a que va en dependencia de lo que se modela. Así tenemos que para una aplicación sencilla se puede realizar entre tres y seis diagramas y para uno complejo se recomienda realizar todos los diagramas.

4.5.1.1. Diagrama de Clases

Es uno de los diagramas base en el modelado, permite la definición de todas las clases a utilizar dentro del sistema, con sus respectivas relaciones, operaciones a realizar, definición de los atributos, así como la definición clara de los objetos.

Los diagramas de clase facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar. Permiten al analista hablarles a los clientes en su propia terminología, lo hace que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos [Schmuller, 1999].

Para entender mejor como realizar el diagrama de clases definamos que es una clase. Una clase es una plantilla de objetos que posen atributos y métodos.

Los atributos son como propiedades o características y los métodos son las acciones de los objetos.

La figura 10 muestra un ejemplo de la notación de UML que captura los atributos y acciones de una clase. El rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre de la clase, el área de en medio sus propiedades y el área inferior sus métodos.

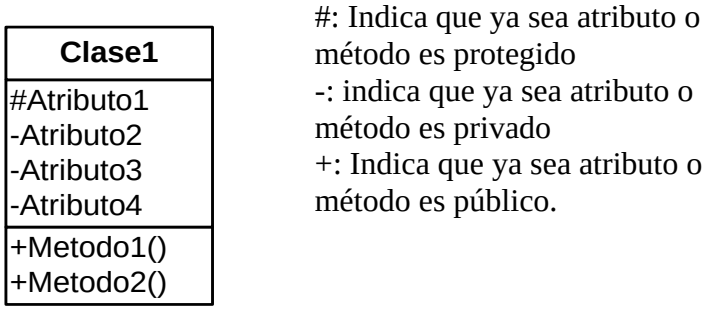


Figura 10. El símbolo UML de una clase

Es importante señalar que un diagrama de clases pueden existir muchas clases que pudieran relacionarse. Por ejemplo una clase podría relacionarse con una otra y el tipo de relación ira de acuerdo con lo que se quiere modelar. Por ejemplo una relación puede ser de uno a uno, uno a muchos o muchos a muchos, todo estará en dependencia de lo que se modela. Verificar siguiente figura. 11.

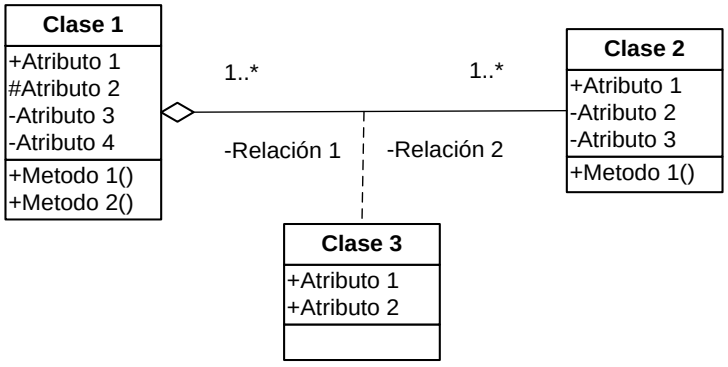


Figura 11. Diagrama de clases con relación de muchos a muchos.

Hay ocacioneses en que apartir de la realción de dos clase nazca una tercera clase (clase asociación) como en la figura 11 de nombre “Clase3” esto debido al tipo de relación (muchos a muchos), mediante esa clase podemos conocer las relaciones entre los objetos de las clase1 y clase2. Pero si la relación fuera de uno a uno el diagrama cambiaría. Verificar siguiente figura 12.

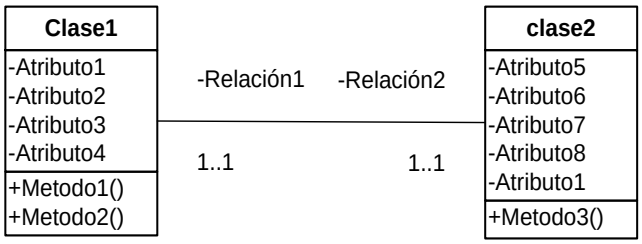


Figura 12. Diagrama de clases con relación de uno a uno.

4.5.1.2. Diagrama de Casos de Uso

Los diagramas de Casos de Uso modelan las distintas operaciones que se esperan de un sistema y como se relaciona con su entorno ya sean usuarios u otras aplicaciones involucradas. Los diagramas de casos de uso permiten especificar y visualizar como trabajara un sistema.

Los elementos para realzar un diagrama son:

4.5.1.2.1. Los Actores: Representado por un figura de una persona aunque no necesariamente sea un persona al que se represente ya que puede ser otra aplicación. Los actores representan algo o alguien que debe interactuar con el sistema o que lo usa de alguna forma. Ver Figura 13.

Figura 13. Actor de un caso de uso

4.5.1.2.2. Casos de Uso: Representado por una elipse y es una operación que realiza el sistema iniciada por los actores o por otro caso de uso, es un requerimiento solucionado por el sistema.

4.5.1.2.3. Relaciones:

Comunicación: (Comunicates) Es la relación de un actor con un caso de uso y se representa por una línea.

Usa:(Uses) Relación entre dos casos de usos, denota cuando un caso de uso hace uso de otro, es decir le permite usar los pasos de un caso de uso dentro de otro.

Extiende: (extends) Relación entre dos casos de usos. Le permite crear un caso de uso mediante la adición de pasos de uno existente

Cabe destacar que para representar el sistema a modelar se hace mediante un rectángulo con nombre indicado en el cual estarán todos los casos de usos de dicho sistema. Los actores estarán fuera del sistema interactuando con los casos de usos.

Es importante indicar que los casos de uso se pueden clasificar y agrupar dependiendo a su nivel de importancia. Por ejemplo hay caso de usos que son esenciales y otros que son extendidos. Por supuesto los primeros tienen mayor relevancia, todo estará en dependencia de lo que se modela.

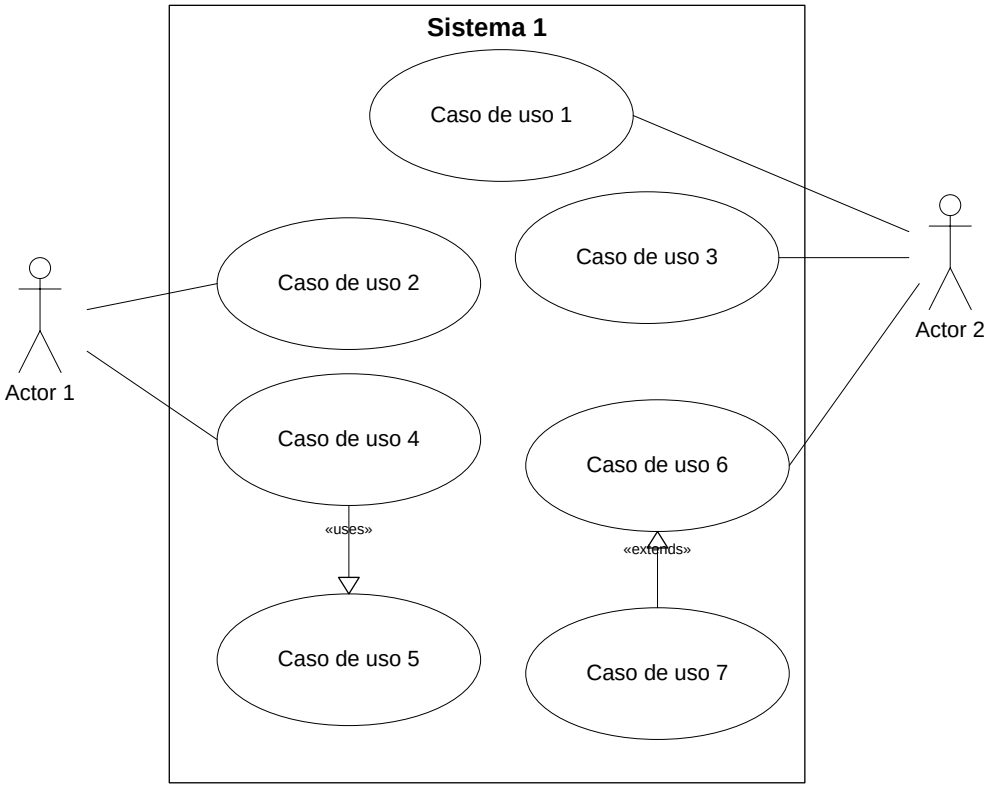


Figura 14. Simbología de un diagrama de casos de uso

4.5.1.3. Diagrama de Actividades

Permite una visualización clara de todos los pasos a seguir dentro del sistema, es decir, todas las actividades que el sistema va a permitir realizar, más específicamente las acciones y uso.

Un diagrama de actividades ha sido diseñado para mostrar una visión simplificada que ocurre durante una operación o proceso [Schmuller, 1999].

Elementos del diagrama de actividades:

Las actividades: se representan por un rectángulo con esquinas redondeadas.

Transición de las actividades: luego de una actividad pueden seguir una o más actividades y la transición de una a otra es representada por una flecha.

Punto inicial: un diagrama de actividad debe contar con un punto inicial y es representado por un circulo relleno pequeño.

Punto final: un diagrama de actividad debe contar con un punto final y es representado por un circulo relleno pequeño con borde blanco.

Decisiones: Casi siempre una secuencia de actividades llegará a un punto donde se realizará alguna decisión. Se puede mostrar un punto de decisión de dos formas una es mostrar la rutas posibles que parten directamente de una actividad y otra es llevar la transición hacia un rombo [Schmuller, 1999]. Ver figura 15.

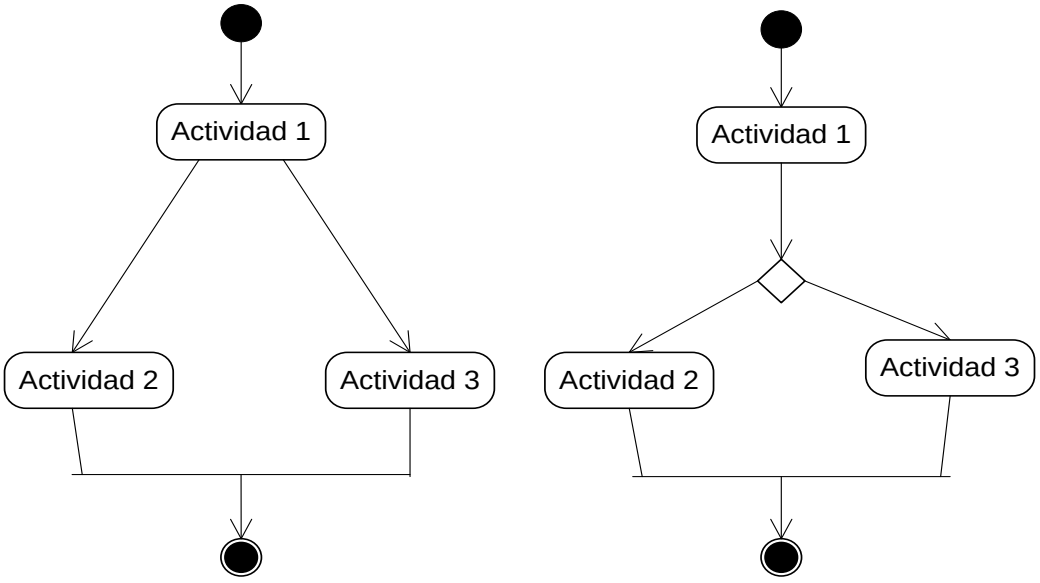


Figura 15. Dos formas de tomar una decisión. Tomado de: [Schmuller, 1999]

Rutas Concurrentes: Conforme modele actividades tendrá la oportunidad de separar una transición en dos rutas que se ejecuten al mismo tiempo y luego se reúnan. Para representar esta división, utilizara una línea gruesa perpendicular a la transición y la rutas partirán de ellas y para representar la incorporación ambas rutas apuntaran a otra línea gruesa [Schmuller, 1999]. Ver figura 16.

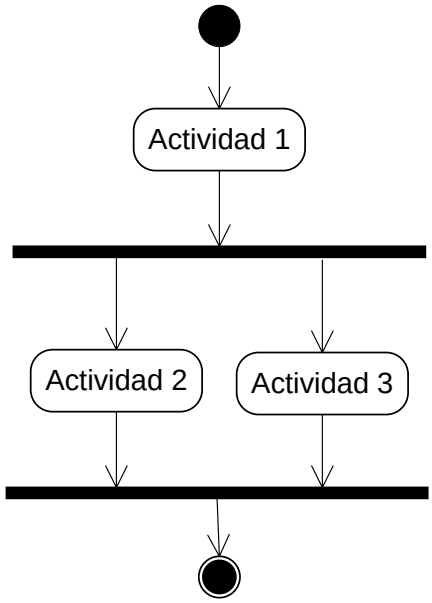


Figura 16. Ejemplo de rutas concurrentes.

Indicaciones: Durante una secuencia de actividades, es posible enviar una indicación. El símbolo para enviar una indicación es un pentágono convexo, y el que lo recibe es un pentágono cóncavo [Schmuller, 1999]. Ver siguiente figura 17.

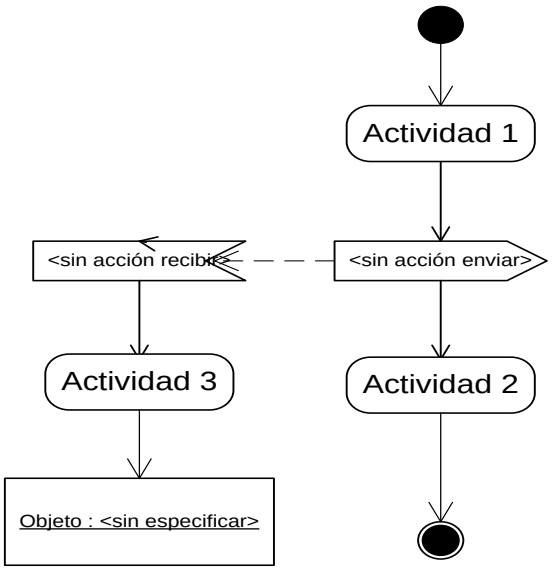


Figura 17. Envío y recepción de una indicación. Tomado de: [Schmuller, 1999]

Marcos de responsabilidad: El diagrama de actividades agrega la dimensión de visualizar responsabilidades, es decir dividir el diagrama en rectángulos en el que se muestre un nombre para cada rectángulo que serán los responsables de realizar las actividades. Ver figura 18.

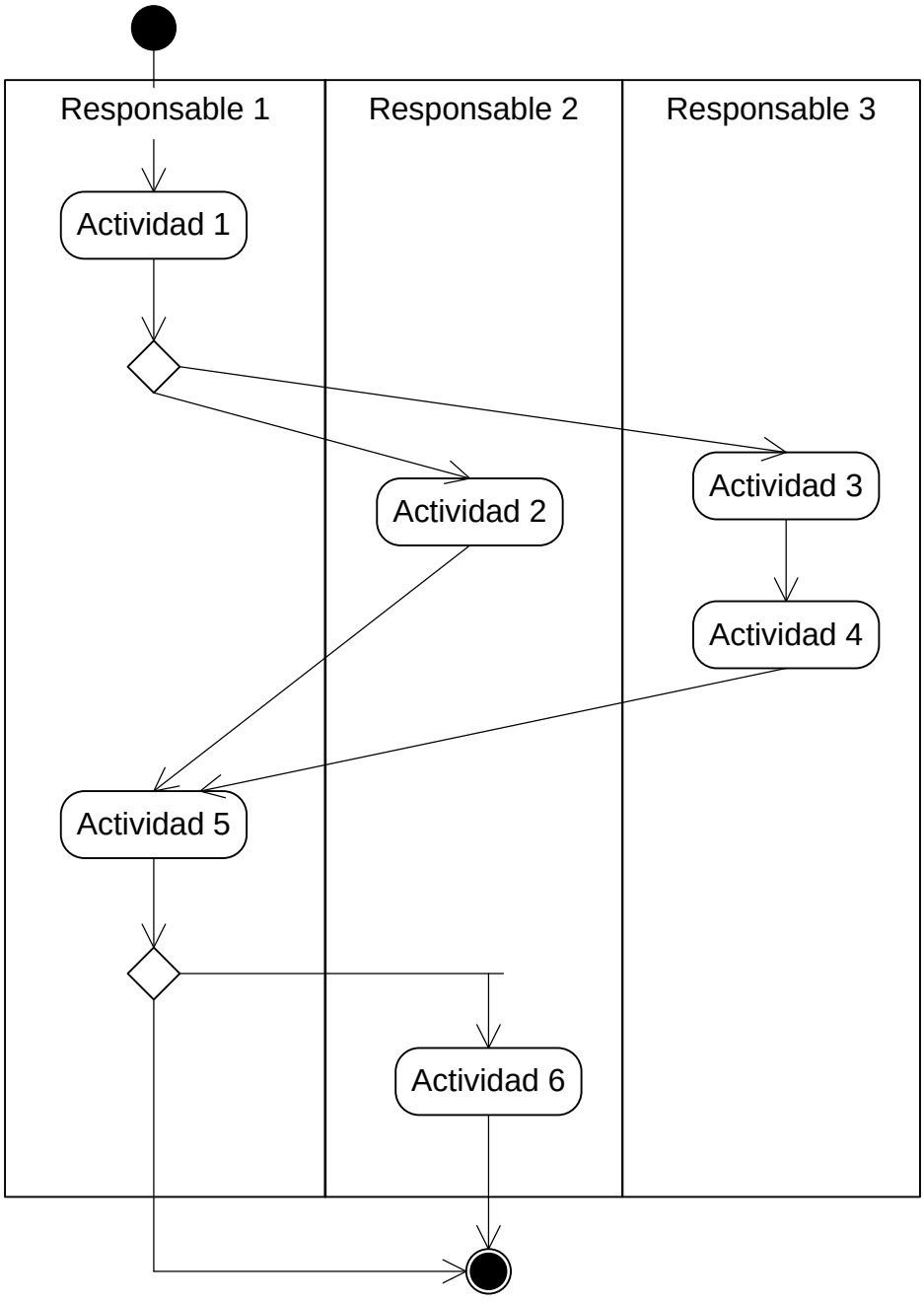


Figura 18. Marcos de responsabilidad.

4.5.1.4. Diagrama de Secuencia

Describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Un rol clasificador, o simplemente "un rol", es la descripción de un objeto, que desempeña un determinado papel dentro de una interacción, distinto de los otros objetos de la misma clase.

Los diagramas de secuencia son importantes ya que pueden dar detalles a los casos de uso.

El diagrama de secuencia costa de objetos que se representan del modo usual. Rectángulos con nombre (subrayado), mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical [Schmuller, 1999].

Como realizar un diagrama de secuencia: (ver figura 19.)

- Los objetos se colocan cerca de la parte superior del diagrama de izquierda a derecha de manera que se ordene de forma más simple.
- De cada objeto sale una línea discontinua conocida como línea de vida del objeto.
- En cada línea de vida se encuentra un pequeño rectángulo conocido como activación, el cual representa una operación que realiza el objeto y el tamaño del rectángulo representa la duración de la operación.
- Existen mensajes que van de la línea de vida de un objeto a otro. También un objeto puede enviarse mensajes a sí mismo. Existe tres tipos de mensajes simples (que es la transición de control de un objeto a otro), sincrónico (un objeto espera respuesta al mensaje enviado para continuar su ejecución), asincrónico (no se espera respuesta y continua su ejecución) [Schmuller, 1999]. ver figura 20.

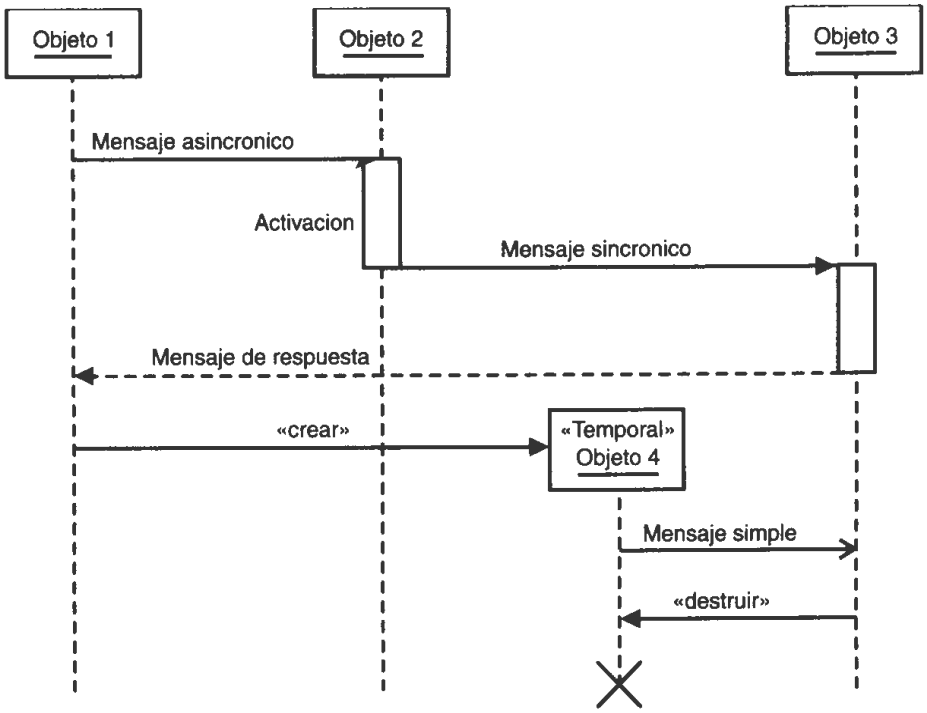
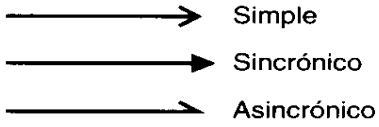


Figura 19. Ejemplo de cómo diseñar un diagrama de secuencia. Tomado de: [Schmuller, 1999]

Figura 20. Símbolos para los mensajes de un diagrama de secuencia. Tomado de: [Schmuller, 1999]



4.5.1.5. Diagrama de colaboración

Un diagrama de colaboraciones es una extensión de un diagrama de objetos. Además de las relaciones entre objetos, el diagrama de colaboraciones muestra los mensajes que se envían los objetos entre sí. Por lo general, evitará la multiplicidad dado que podría ser fuente de confusión [Schmuller, 1999].

El diagrama de secuencia y colaboraciones son semánticamente equivalentes, es decir que representan la misma información [Schmuller, 1999].

La diferencia entre ambos es que el diagrama de secuencia se organiza de acuerdo al tiempo en cambio el de colaboraciones destaca el contexto y organización general de los objetos.

Como elaborar un diagrama de colaboración: (ver figura 21)

- Los objetos son representan por rectángulos que contiene su nombre y clase a la que pertenece.
- Los objetos se unen por una línea continua y los mensajes que se envían se representan por una flecha y una etiqueta que llevara el nombre del mensaje que finalizara con un par de paréntesis donde se pondrán parámetros en caso de haberlo.

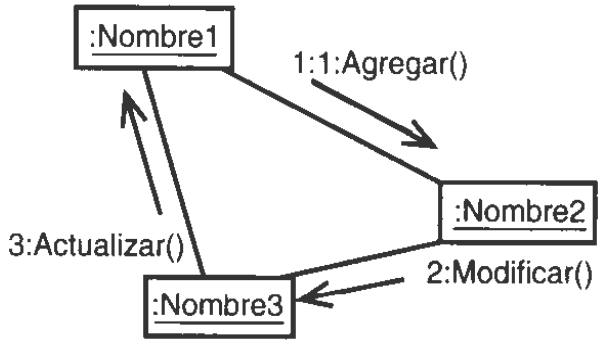


Figura 21. Simbología del diagrama de colaboraciones. Tomado de: [Schmuller, 1999]

4.5.1.6. Diagrama de Componentes

Describen los elementos del sistema con sus respectivas relaciones. Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables.

Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistema o entre diferentes partes de un sistema. El símbolo

principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos a su lado izquierdo [Schmuller, 1999]. Ver siguiente figura (figura 22).

Figura 22. El símbolo que representa a un componente. Tomado de: [Schmuller, 1999]

4.5.1.7. Diagrama de Objetos

Es una entidad que encapsula comportamiento y estado de una unidad, hace uso de las propiedades del encapsulamiento. Los diagramas de objetos son una derivación de los diagramas de clases, y tienen como función principal proteger los datos, disminuir el acoplamiento, favorece la modularidad y el mantenimiento.

Un objeto es una instancia de una clase (una entidad que tiene valores específicos de los atributos y métodos de los objetos).

Los atributos son propiedades o características de los objetos y los métodos sus acciones, eventos o comportamientos de estos.

El diagrama de objetos muestra las instancias de los objetos sobre las clases cabe destacar que se debe decidir una situación que se quiere representar del sistema a modelar. Ver figura 23.

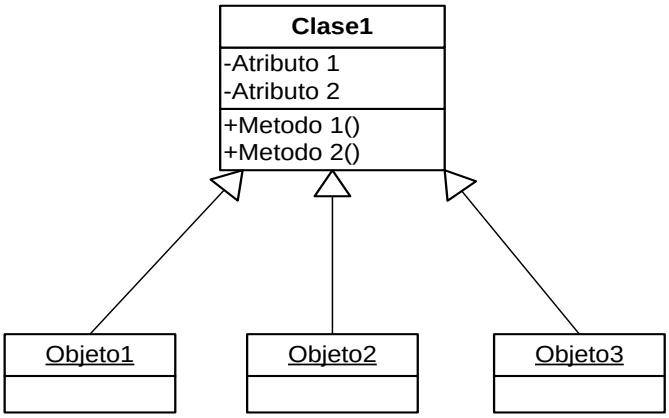


Figura 23. El símbolo de un diagrama de objeto. Tomado de: [Schmuller, 1999]

4.5.1.8. Diagrama de Estado

Muestra un conjunto de estado por la cual pasa un objeto durante su vida en una aplicación dentro del sistema, estos están representados por autómatas finitos. Cada objeto se mueve a través del estado de una forma constante según la función definida por el sistema para el objeto.

La presentación del diagrama de estados es parecida al diagrama de actividades la diferencia está en lo que representa cada uno.

Elementos de un diagrama de estados: (ver figura 24)

- El diagrama de estados debe tener un estado inicial y final, estos son representados por círculo relleno y un círculo relleno con borde blanco respectivamente.
- Los estados son rectángulos con borde redondeados y representan el periodo por donde pasa un objeto el cual espera una operación o evento para pasar a otro estado.
- Los eventos son flechas punteadas en las que se puede indicar el nombre y valor que recibirá, representan la transición de un estado a otro.

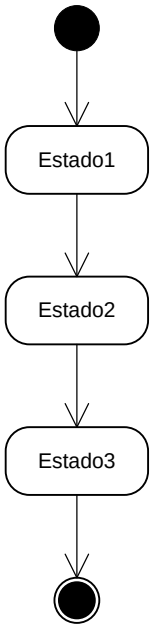


Figura 24. Símbolo que representa estados de un objeto.

4.5.1.9. Diagrama de Distribución o despliegue

El diagrama de distribución muestra la forma en que luce un sistema físicamente cuando sea conjugado, este es representado por nodos [Schmuller, 1999].

Un nodo se representa por un cubo y se le debe asignar un nombre que indique el recurso a representar.

Una línea asocia a dos nodos y simboliza una conexión entre ambos. Los tipos de nodos son procesador (que puede ejecutar un componente) y dispositivo (que no lo puede hacer). Los dispositivos por lo general interactúan con el mundo. Los diagramas de distribución son muy útiles para modelar redes [Schmuller, 1999].

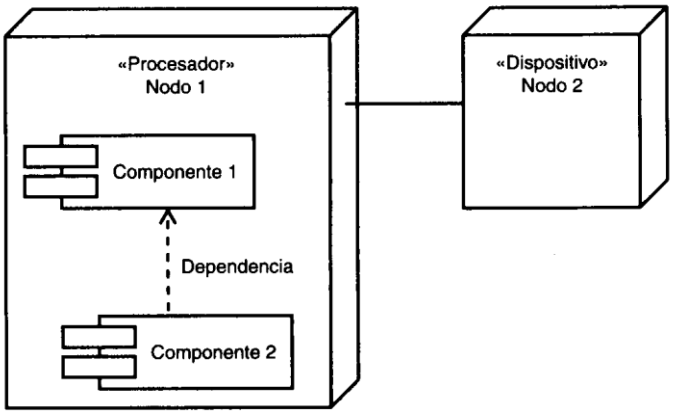


Figura 25. Símbolo de un diagrama de distribución. Tomado de: [Schmuller, 1999].

4.6. Conceptos de Estudios de Factibilidad

El estudio de factibilidad comúnmente se realiza durante la fase de diseño de sistema y se presenta con el fin de determinar si llevar a cabo un proyecto es viable considerando tres aspectos: económico, operacional y técnico. [Hynes Richard, 2003]

4.6.1. Factibilidad técnica

El análisis de factibilidad técnica evalúa si pueden implementarse los requerimientos con la tecnología que se posee actualmente, esto es, si el equipo y software están disponibles, y si tienen las capacidades técnicas requeridas para el diseño que se esté considerando, es decir, se requiere listar todas las herramientas a utilizar y sus características para que el sistema se pueda desarrollar y ejecutar adecuadamente.

4.6.2. Factibilidad operacional

Este tipo de factibilidad examina si el sistema puede ser utilizado sin alterar el organigrama actual de la empresa o institución que lo va a utilizar. Para realizar un estudio de factibilidad operacional se deben considerar algunos aspectos, como son:

- Un nuevo sistema puede ser demasiado complejo para los usuarios de la organización o los operadores del sistema. Si lo es, los usuarios pueden ignorar el sistema o bien usarlo en tal forma que cause errores o fallas en el sistema.

- Un sistema puede hacer que los usuarios se resistan a él como consecuencia de una técnica de trabajo, miedo a ser desplazados, intereses en el sistema antiguo u otras razones.

- Un nuevo sistema puede introducir cambios demasiado rápido para permitir al personal adaptarse a él y aceptarlo.

4.6.3. Factibilidad económica

Los estudios de factibilidad económica se basan en calcular los costos de implementación del proyecto, lo que incluye los gastos en que incurren los desarrolladores y analistas, precios de hardware y software con que se desarrollara el proyecto, entre otros.

La idea principal de este estudio de factibilidad es realizar un presupuesto del proyecto en general, para mostrarlo a los clientes y así, ellos aprueben o rechacen la idea del proyecto, según les convenga.

4.7. Auditoría

Los sistemas informáticos se han desarrollado conjuntamente con el mundo industrial, el crecimiento económico y las nuevas tendencias de tecnologías; esto ha conllevado a que se creen nuevas formas de controlar todo este movimiento en las empresas con el fin de detectar y prevenir el fraude, con el tiempo las pruebas se centraron en cerciorar la condición financiera y las ganancias de las empresas, a este proceso se le llamó auditoría.

La palabra auditoría proviene del latín auditorius que significa oír, escuchar, atender. Actualmente auditoría es el proceso formal y sistemático para comprobar la eficiencia y eficacia de las operaciones realizadas en las organizaciones según sus objetivos establecidos.

La contraloría general de la república define a la auditoria como un examen objetivo, sistemático y profesional de las operaciones u actividades o de ambas a la vez, practicado con posterioridad a su ejecución, con la finalidad de verificarlas, evaluarlas y elaborar el correspondiente informe que debe contener comentarios, conclusiones y recomendaciones.

Una auditoría es realizada en un área de una empresa, especialmente aquellas con mayor vulnerabilidad, se lleva a cabo en un período determinado. En una auditoría se ven involucrado tanto el auditado como el auditor, la persona que realiza la auditoría es llamado auditor es el encargado de realizar las adecuadas observaciones, pruebas, procesos y un conjunto de actividades para verificar la integridad de la información del área en la que se está ejecutando la auditoría, según reglas establecidas por la empresa, mientras que el auditado es aquella persona a la que se le aplican este conjunto de reglas para verificar la validez de los resultados expuesto por él de tal forma que cumpla con los estatutos impuesto por la organización.

Las empresas realizan este proceso de forma independiente, cada una utiliza diferentes métodos, técnicas y procedimientos para realizar una auditoría ya sea de forma interna o externa determinada por un tiempo.

4.7.1. Plan de Auditoría

Al planear se deben definir objetivos, así como el alcance y la metodología para alcanzar esos objetivos. Los objetivos son los que la auditoría trata de alcanzar, identifican los puntos y las partidas específicas a revisar y los aspectos y hallazgos que los auditores esperan desarrollar. Los procedimientos de auditoría son los pasos y exámenes específicos que se llevan a cabo para atender los objetivos.

La planeación se inicia con la comprensión de las operaciones de la entidad a ser examinada. Estas actividades implican reunir información que nos permita llevar a cabo una evaluación del control interno y determinar el grado del riesgo de auditoría.

La labor de auditoría debe ser controlada por medio de la utilización de programas de trabajo. Estos programas son planes que se hacen por adelantado del trabajo que ha de efectuarse durante la ejecución y están basados en objetivos aprobados y en la información disponible sobre las actividades, operaciones y procedimientos de la entidad.

El proceso de planeación deberá ser supervisado por los niveles adecuados del grupo de auditores. Asimismo es un proceso continuo durante toda la auditoría, por lo tanto, los auditores conforme se vayan allegando de más información deben evaluar si necesitan hacer ajustes a los objetivos, alcance y metodología. Los programas de auditoría son parte de la planeación.

Concluida la planeación comienza la etapa de ejecución a través de la cual el auditor realiza los programas y ejecuta su plan de revisión aplicando las pruebas sustantivas y de control necesarias con el fin de determinar los hallazgos de auditoría que habrán

de presentarse en su informe reuniendo la evidencia suficiente y competente que pruebe los hechos encontrados.

Algunos factores a considerar son:

• Permitir que el equipo asignado pueda hacer uso apropiado del potencial humano disponible.

• Identificar las áreas más importantes y los errores potenciales, determinar el nivel de riesgo y programar la obtención de la evidencia necesaria para determinar observaciones y recomendaciones.

• Determinar la forma de obtener los datos necesarios e informar acerca de la información financiera y presupuestal de la entidad.

• La naturaleza y alcance de los procedimientos, puede variar según el tipo de auditoría, el volumen de las operaciones, la experiencia del auditor, el conocimiento de las operaciones y el nivel de riesgo.

• El éxito de una adecuada planeación radica en que esta sea efectuada por miembros experimentados del equipo de auditores, que posean especialización que requiera las circunstancias de la comisión.

• Determinar lo que debe hacerse durante la auditoría, por quién y cuándo. La planeación es vista como una secuencia de pasos que conducen a la ejecución de procedimientos de auditoría; sin embargo, este proceso debe proseguir en forma continua durante el curso de la misma y dirigido a la obtención del objetivo del examen.

La planeación tiene una importancia fundamental, porque si se realiza de manera adecuada se asegura el cumplimiento de los siguientes objetivos:

- Cumplir con el alcance y el tiempo establecido.
- Aplicar la normativa correspondiente.
- Utilizar los procedimientos necesarios.
- Obtener resultados con eficiencia y eficacia.
- Informar oportuna y objetivamente.

Planear implica la aplicación de técnicas y procedimientos (tales como: investigar, analizar, etc.) a efecto de recabar la información, que permita tomar en cuenta todos los aspectos que proporcionen un panorama completo de la entidad a auditar, de su control interno, recomendaciones de auditorías anteriores, situaciones especiales, estructura, volumen de operaciones, etc.

Además se deben considerar, los recursos humanos y materiales disponibles para cumplir con los objetivos que se pretenden establecer. La planeación debe permitir la realización de cambios y adecuaciones; debe ser flexible, con el fin de afrontar situaciones imprevistas, permitiendo replantear objetivos sobre la marcha. Una planeación eficiente, presentará menos adecuaciones por situaciones imprevistas.

4.7.2. Tipos de Auditoría

La Auditoría constituye una herramienta de control y supervisión que contribuye a la creación de una cultura de la disciplina de la organización y permite descubrir fallas en las estructuras o vulnerabilidades existentes en la organización.

Existen diferentes tipos de auditorías aplicables en diferentes áreas, entre ellas tenemos:

Auditoría Financiera: Se encarga de revisar la veracidad de las cuentas y registros financieros de una organización.

Auditoría Operacional: Evalúa normas y procedimientos que rigen una empresa e investiga que estas se cumplen.

Auditoría Fiscal: Se ocupa de observar que se cumpla con las obligaciones en materia tributaria a los contribuyentes.

Auditoría Interna: Es la auditoría que se ejecuta por el personal que labora dentro de la misma organización.

Auditoría Externa: Esta auditoría se realiza por una entidad ajena a la empresa, para que los resultados no sean alterados por conveniencia o fraude.

Auditoría administrativa: Determina si la estructura organizativa, planes y objetivos de la organización operan eficazmente.

Auditoría Informática: Es el proceso de procesar, evaluar y revisar los sistemas informáticos para verificar el cumplimiento correcto según las reglas impuestas por la organización.

Auditoría de resultados de programas: Esta auditoría evalúa la eficacia y congruencia alcanzadas en el logro de los objetivos y las metas establecidas, en relación con el avance del ejercicio presupuestal.

Auditoría de legalidad: Este tipo de auditoría tiene como finalidad revisar si la dependencia o entidad, en el desarrollo de sus actividades, ha observado el cumplimiento de disposiciones legales que sean aplicables (leyes, reglamentos, decretos, circulares, etc.).

Auditoría integral: Es un examen que proporciona una evaluación objetiva y constructiva acerca del grado en que los recursos humanos, financieros y materiales son manejados con debidas economías, eficacia y eficiencia.

4.8. Herramientas de Software a utilizar

4.8.1. Visual Basic .Net

Visual Basic es un lenguaje poderoso y escalable, con compatibilidad total con el lenguaje común del .NET Framework, el objetivo de Microsoft con VB.NET es facilitar la programación para internet de la misma manera que las primeras versiones de Visual Basic facilitaron la programación de aplicaciones para Windows [Hynes, 2003].

La plataforma .NET es una capa de software que se coloca entre el Sistema Operativo (SO) y el programador y que abstrae los detalles internos del SO. Las características fundamentales de esta plataforma son las siguientes:

Portabilidad: Debido a la abstracción del programador respecto al SO, una aplicación .NET puede ser ejecutada en cualquier SO de cualquier máquina que disponga de una versión de la plataforma.

Multilenguaje: Cualquier lenguaje de programación puede adaptarse a la plataforma .NET y ejecutarse en ella.

Interoperabilidad: La interoperabilidad entre diferentes trozos de código escritos en diferentes lenguajes es total.

La programación en VB. NET pone como realce el uso de clase, dado que ahora es un lenguaje meramente orientado a objetos.

4.8.1.1. Framework

Es un esquema, un esqueleto, un patrón para el desarrollo y/o la implementación de una aplicación.

A continuación se resumen las ventajas más importantes que proporciona .Net Framework:

• Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

• Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL).

• Compilación just-in-time: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma.

• Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura.

• Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

Visual Basic .NET moderniza y simplifica el lenguaje de programación Visual Basic, con algunas novedades sintácticas, herencia simple, tratamiento de hebras y manejo de excepciones.

La plataforma .NET está compuesta por dos pilares fundamentales:

– Common Language Runtime (CLR): Este es el entorno de ejecución que traducirá el código intermedio CIL a Código máquina y por tanto permitirá ejecutar cualquier aplicación de la Plataforma. Algunas implantaciones del CLR tienen incorporado lo que se denomina JIT (Just in time) de forma que sólo se traduce a código máquina las partes Necesarias y se recuerdan por si vuelven a ser llamadas (e.g. funciones). Así se consigue un mayor rendimiento de ejecución.

4.8.1.2. ADO.NET

ADO por las siglas en ingles ActiveX Data Objets, creadas por Microsoft, es un intermediario entre un programa y una de base de datos, el uso de ADO permite la comunicación entre ambos, es decir ADO posee un conjunto de componentes que permiten manejar Bases de datos.

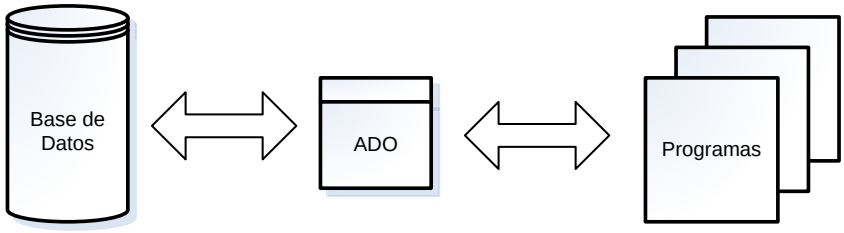


Figura 26. Ado como intermediario.

ADO sustituyó a DAO (Data Access Object) y a RDO (Remote Data Object), tiene la misma funcionalidad de estos pero más fácil de entender y programar. Con el uso de componentes de ADO se puede acceder, agregar, actualizar, eliminar registros entre otras cosas de una base de datos.

Entre los principales componentes de ADO están:

- Connection: permite realizar una conexión con una base de datos.
- Recordset: Permite navegar entre los registro de la base de datos.
- Comand: Permite enviar sentencias SQL que se ejecutarán sobre las base de datos.

ADO.NET

Es la evolución de ADO, y se usa en los entornos de programación de la plataforma .NET, de Microsoft, para manejar bases de datos tanto en Windows como en la Web mediante ASP.NET.

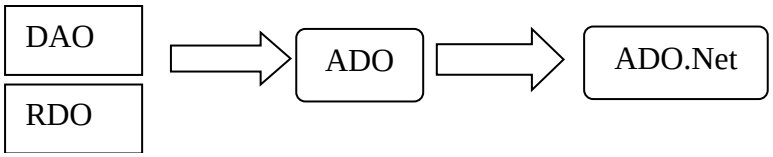


Figura 27. Evolución de proveedores para acceso a datos

ADO.NET es mucho más poderoso. Es la estrategia que ofrece Microsoft para cubrir todas las necesidades que ADO no ofrecía, posee una serie de objetos que son los mismos que aparecen en ADO e introduce nuevos objetos. Lo que permite tener numerosas ventajas. [Blanco, 2003]
Principales componentes de ADO.NET:

- Connection: Permite establecer la conexión con la base de datos.
- Comand: Permite ejecutar sentencias SQL contra el almacén de datos.
- DataReader: Permita la navegación entre los registros de las base de datos.
- DataSet: Permite crear una copia la base de datos y tener acceso a ella sin tener que estar conectado lo que mejora el rendimiento en la red.
- DataAdapter: Permite llenar al Dataset con información de la base de datos.

Entre las principales ventajas que posee están:

- Navegación mejorada. Acceso a datos desconectados
- Mejor rendimiento
- Clase de objetos más estructurados

4.8.2. SQL SERVER 2000

Microsoft SQL Server 2000 puede proporcionar los servicios de base de datos necesarios para sistemas extremadamente grandes. SQL Server 2000 dispone de protección total para los entornos de usuarios, con medidas de seguridad que evitan problemas como tener varios usuarios intentando actualizar los mismos datos al mismo tiempo. SQL Server 2000 asigna también de manera muy eficaz los recursos

disponibles, como memoria, ancho de banda de la red y E/S del disco, entre los distintos usuarios.

Los sitios de Internet extremadamente grandes pueden dividir sus datos entre varios servidores, extendiendo la carga de procesamiento entre varios equipos y permitiendo que el sitio sirva a miles de usuarios simultáneos.

Las aplicaciones de SQL Server 2000 se pueden ejecutar en el mismo equipo que SQL Server 2000. La aplicación se conecta a SQL Server 2000 utilizando los componentes de comunicaciones entre procesos (IPC) de Windows, tales como memoria compartida, en lugar de una red. Esto permite utilizar SQL Server 2000 en sistemas pequeños en los que una aplicación debe almacenar los datos localmente.

4.9. Pruebas de Software

Las pruebas a un sistema pueden iniciar a partir de las unidades de programas tales como funciones u objetos que van integrando un sistema en sí, el cual después se unifica en subsistemas y sistemas creando un producto al cual se le puede seguir realizando pruebas antes de una entrega final, luego el producto es entregado al cliente quien puede evaluarlo y verifica si es totalmente funcional.

Las pruebas se realizan a sistemas grandes y complejos, así como a sistemas que no poseen muchos requerimientos, pero que tienen como objetivo entregar un producto totalmente funcional. Las dos actividades fundamentales de pruebas son la prueba de componentes (probar las partes del sistema) y la prueba del sistema (probar el sistema como un todo) [Pressman, 2002].

El objetivo de la etapa de la prueba de componentes es descubrir defectos probando componentes de programas individuales (funciones, objetos o componentes reutilizables). Durante las pruebas del sistema, estos componentes se integran para formar subsistemas o el sistema completo. En esta etapa, la prueba del sistema debería centrarse en establecer que el sistema satisface sus requerimientos funcionales y no funcionales, y no se comporta de forma inesperada. Inevitablemente, los defectos en los componentes que no se han detectado durante las primeras etapas de las pruebas se descubren durante las pruebas del sistema.

El proceso de pruebas del software tiene dos objetivos distintos:

1- Para demostrar al desarrollador y al cliente que el software satisface sus requerimientos.
2- Para descubrir defectos en el software en que el comportamiento de este es incorrecto, no deseable o no cumple su especificación [Pressman, 2002].

4.9.1. Pruebas del sistema

Las pruebas del sistema están basadas en probar un sistema integrado de funciones u objetos. En cada integración de componentes del sistema se va realiza una entrega al cliente quien evalúa la funcionalidad del mismo, por eso se dice que esto es un proceso iterativo.

Para la mayoría de los sistemas complejos, existen dos fases distintas de pruebas del sistema: Pruebas de integración, Pruebas de entregas y Pruebas del sistema.

4.9.1.1. Pruebas de integración

Consiste en la manipulación directa del código, cuando es descubierto un problema y el punto exacto al momento de la depuración este es reparado. El objetivo principal es encontrar defectos en el sistema.

La integración del sistema implica identificar grupos de componentes que proporcionan alguna funcionalidad del sistema e integrar estos añadiendo código para hacer que funcionen conjuntamente. La principal dificultad que surge durante las pruebas de integración es la localización de los errores. Existen interacciones complejas entre los componentes del sistema, y cuando se descubre una salida anómala, puede resultar difícil identificar donde ha ocurrido el error.

Cuando se planifica la integración, tiene que decidirse el orden de integración de los componentes. En un proceso como XP, el cliente se implica en el proceso de desarrollo y decide que funcionalidad debería incluirse en cada incremento del sistema. Por lo tanto, la integración del sistema está dirigida por las prioridades del cliente. En otras aproximaciones al desarrollo, cuando se integran componentes comerciales y componentes especialmente desarrollados, el cliente puede no estar implicado y el equipo de integración decide sobre las prioridades de la integración.

Para probar una nueva característica, pueden tener que integrarse varios componentes diferentes. Las pruebas pueden revelar errores en las interacciones

entre estos componentes individuales y otras partes del sistema. La reparación de errores puede ser difícil debido a que un grupo de componentes que implementan la característica del sistema pueden tener que cambiarse. Además, la integración y prueba de un nuevo componente puede cambiar el patrón de las interacciones de componentes ya probados. Se pueden manifestar errores que no habían aparecido en las pruebas de la configuración más simple.

Estos problemas significan que, cuando se integra un nuevo incremento, es importante volver a ejecutar las pruebas para incrementos previos, así como las nuevas pruebas requeridas para verificar la nueva funcionalidad del sistema.

4.9.1.2. Pruebas de entregas

Son el proceso de probar una entrega del sistema que será distribuida a los clientes. El principal objetivo de este proceso es incrementar la confianza del suministrador en que el sistema satisface sus requerimientos. Si es así, este puede entregarse como un producto o ser entregado al cliente. Para demostrar que el sistema satisface sus requerimientos, tiene que mostrarse que este entrega la funcionalidad especificada, rendimiento y confiabilidad, y que no falla durante su uso normal.

Las pruebas de entregas son normalmente un proceso de pruebas de caja negra en las que las pruebas se derivan a partir de la especificación del sistema. El sistema se trata como una caja negra cuyo comportamiento solo puede ser determinado estudiando sus entradas y sus salidas relacionadas. Otro nombre para esto es pruebas funcionales, debido a que al probador solo le interesa la funcionalidad y no la implementación del software.

En la figura 28 se muestra un ejemplo de la forma en que trabajan las pruebas en ella se introducen entrada de datos de prueba al sistema y examina los salidas. Si las salidas no son las esperadas, entonces la prueba ha detectado un problema con el software.

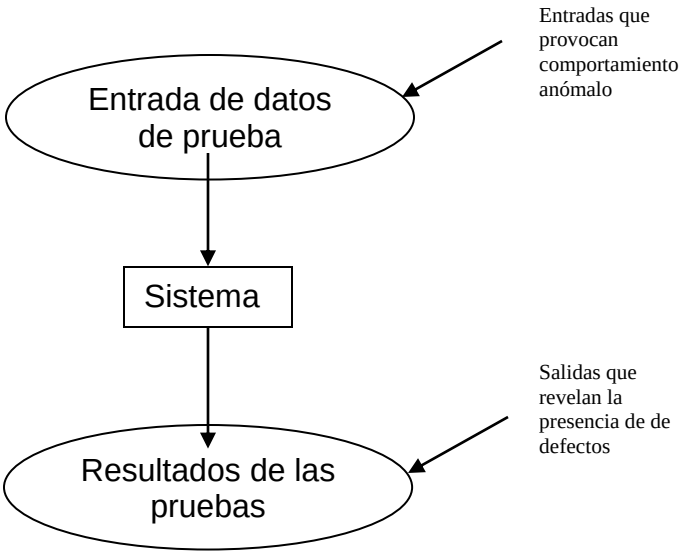


Figura 28. Pruebas de cajas negras

El objetivo es seleccionar datos de entradas que tiene una alta probabilidad de generar fallas a la hora de ejecución en el sistema.

Algunos de los posibles casos pueden ser:

1- Elegir entradas que generen todos los errores posibles.
2- Repetir la misma entrada o series de entradas varias veces.
3- Forzar a que se generen las salidas inválidas.
4- Forzar los resultados de los cálculos para que sean demasiado grandes o demasiado pequeños.

Para cada una de estas pruebas, deberán diseñarse un conjunto de pruebas que incluyan entradas validas e inválidas y que generen salidas validas e inválidas. También deberían organizarse pruebas basadas en escenarios para que los escenarios más probables sean probados primero, y los escenarios inusuales o excepcionales sean probados más tarde, de forma que el esfuerzo se centre en aquellas partes del sistema que reciben un mayor uso.

4.9.1.3 Pruebas de rendimiento

Cuando todas las partes de un sistema se han integrado, es posible probar las propiedades emergentes del sistema tales como rendimiento y fiabilidad. Las pruebas de rendimiento tienen que diseñarse para asegurar que el sistema pueda procesar su carga esperada. Esto normalmente implica planificar una serie de pruebas en las que la carga se va incrementando regularmente hasta que el rendimiento del sistema se hace inaceptable.

El objetivo de estas pruebas es probar la funcionalidad total del sistema y comprobar si cumple con los requerimientos que el cliente necesita, así como descubrir fallas o defectos del mismo.

Para comprobar el rendimiento del sistema lo recomendable es que se pruebe con datos reales y que se cerciore la funcionalidad tal como si estuviese operando casi totalmente, de esta manera se ponen en juego un sin números de actividades que debería resolver correctamente. Las pruebas de rendimiento implican estresar el sistema realizando demandas que están fuera de los límites del diseño del software.

El objetivo de las pruebas de estrés es poner al máximo la funcionalidad del sistema, de tal manera que se cargue con la cantidad de trabajo al cual debería normalmente trabajar, si el sistema falla, las pruebas han cumplido su objetivo.

Las pruebas de estrés realiza un sin número de actividades en el sistema, estas actividades tiene como objetivo cargar el sistema acercándose a la máxima carga del diseño del sistema hasta que el este falle. Este tipo de pruebas tienen dos funciones:

1- Prueba el comportamiento de fallo de ejecución del sistema: Al efectuarse una carga al sistema, se espera que el fallo no provoque anomalías o pérdidas de datos en el sistema, el objetivo de estas es encontrar fallas ligeras o pequeñas y no que el sistema colapse.

2- Sobrecargan el sistema y pueden provocar que se manifiesten defectos que normalmente no serán descubiertos. Aunque se puede argumentar que estos defectos es improbable que causen fallos de funcionamiento en un uso normal, puede haber combinaciones inusuales de circunstancias normales que las pruebas de estrés pueden reproducir [Pressman, 2002].

4.9.2. Pruebas de componentes

Las pruebas de componentes son el proceso de probar los componentes individuales en el sistema. Este es un proceso de pruebas de defectos, por lo que su objetivo es encontrar defectos en estos componentes.

Existen diferentes tipos de componentes que pueden probarse en esta etapa:

1. Funciones individuales o métodos dentro de un objeto.
2. Clases de objetos que tienen varios atributos y métodos.
3. Componentes compuestos formados por diferentes objetos o funciones.

4.9.2.1. Pruebas de interfaces

Muchos componentes en un sistema no son simples funciones u objetos, sino que son componentes compuestos formados por varios objetos que interaccionan. Se accede a las funcionalidades de estos componentes a través de sus interfaces definidas. Entonces las pruebas de estos componentes se ocupan principalmente de probar que la interfaz del componente que se comporte de acuerdo con su especificación. Los casos de prueba no se aplican a componentes individuales, sino a la interfaz creada por los componentes combinados.

Las interfaces de componentes son varios entre ellos parámetros que pasan de un componente a otro, componente que se comparte por medio de memoria compartida, paso de mensajes de un componente a otro; el cual debe realizar una respuesta.

Las pruebas para encontrar defectos en las interfaces son difíciles debido a que algunos defectos de las interfaces solo se pueden manifestar en condiciones inusuales los más comunes se pueden dar por un mal uso de la interfaz, no hay una compresión clara de la interfaz y errores temporales dados generalmente en sistemas que funcionan en tiempo real.

Para no caer en tales errores se recomienda tener presente una lista de los componentes y de las llamadas externas de los mismos, examinar el código a probar, probar valores nulos, diseñar pruebas que hagan fallar al componente, aplicar otras pruebas como la de estrés, entre otras.

5. Diseño Metodológico

5.1. Tipo de Estudio

El tipo de estudio que se realizó Explicativo, por que se busca las causa y que se necesitan para realizar el trabajo y la importancia que este tendrá y como efecto el resultado del mismo al desarrollar el sistema, el sistema ayudara a la gerencia a tomar decisiones sobre el desarrollo de una auditoria especifica, se aplican los conocimientos adquiridos acerca del análisis, diseño e implementación de un sistema, para que ayude a interpretar la dirección de las actividades que se debe realizar para elaborar el sistema.

5.2. Análisis y diseño del sistema

El Sistema de administración de planes anuales y trabajos de auditoría es un sistema para el control y creación de planes de auditorías, se presenta de manera automatizada, un modelo que define de forma estándar y que reúne de manera global lo necesario para que una unidad de auditoría interna o firmas de auditoría externa pueda utilizarlo perfectamente para llevar un control detallado de sus auditorías.

El sistema define la creación como primera instancia de un plan anual, el que contendrá cada una de las auditorías a realizarse dentro de este período, así como las fases y actividades involucradas. El organigrama de la empresa es un foco de gran importancia, porque debido a ella se definen los permisos que tienen los usuarios dentro del sistema.

El Sistema informático de Administración de Planes Anuales y Trabajos de Auditorias (SIAPTRA) permite:

1. Registrar planes anuales, incluyen año y descripción.
2. Crear un registro por cada personal de trabajo que incluye login, contraseña, nombres, apellidos, sexo, dirección, teléfono, cargo y división asignada.

3. Organizar por divisiones de auditorías al personal de trabajo.
4. Registrar detalles de las unidades que serán auditadas, tales como nombre, teléfono, áreas y representantes.

5. Registrar los planes de auditorías. Esto contiene nombre, objetivo, estado, unidad auditada, fechas planificadas, personal encargado, fases y sus actividades.

6. Calcular costos totales y generar hoja de tiempo por quincena para las auditorías.

7. Brindar informes del avance de las auditorías, de acuerdo a la cantidad de actividades realizadas en una fecha específica de ejecución, y a la vez todo el personal involucrado en cada una de las actividades dentro de la fase de las auditorías.

5.2.1. Identificación de problemas, oportunidades y objetivos.

En esta fase surge la necesidad de plantear un modelo que resume de manera estándar los procesos de control y manejo de planes anuales y trabajos de auditorías que pueda adaptarse a firmas o unidades internas de auditorías. Este modelo está planteado en el punto 5.3.

A partir de dicho modelo se desarrolló el Sistema informático de Administración de Planes Anuales y Trabajos de Auditorías (SIAPTRA).

5.2.2. Determinación de los requerimientos de información

El objetivo principal en esta etapa fue conocer los requerimientos del sistema a desarrollar, la forma como registran la información, los datos que toman en cuenta, el proceso de ejecución de actividades y la asignación del personal. Se verificaron las normativas de planes de auditorías, publicados en la Contraloría General de la República (CGR) de Nicaragua [9], se realizaron investigaciones de auditorías en Internet y se tomó como punto de enfoque información brindada por nuestro tutor Msc. Alejandro Antonio Ruiz Segovia, auditor informático del Instituto Nicaragüense de Seguridad Social (INSS), para plantear el modelo que sirvió de base para el desarrollo del sistema.

5.2.3. Análisis de las necesidades del sistema

Tomando como base la etapa anterior, se realizaron estudios de factibilidad del sistema (5.4), se elaboró un diagrama de flujo de datos (Anexo 9.1), se utilizó la herramienta UML (Anexos 9.2), para la modelación del sistema y normalización de los elementos de datos (Anexos 9.3), a partir de los cuales se desarrolló el diccionario de datos (Anexos 9.5) que contiene todos los elementos que utiliza el sistema.

5.2.4. Diseño del sistema

En esta etapa se diseñó la interfaz de usuario que se utilizará, lo que incluyo entradas que permitieran una captura correcta datos y salidas de acuerdo a las necesidades del usuario; y también se creó la base de datos, haciendo uso del proceso de normalización y salidas de datos**.** (Anexos 9.3)

5.2.5. Desarrollo y documentación del software

El software fue desarrollado en el lenguaje de Visual Basic .Net 2005 con el gestor de base de datos SQL server 2000, ambas aplicaciones ofrecen un sin número de ventajas para elaboraciones de software, se elaboró el manual de usuario para el uso correcto del sistema.

5.2.6. Pruebas y mantenimiento del sistema

Las pruebas del sistema se fueron haciendo a lo largo de su desarrollo, con el objetivo de reducir fallas y ganar tiempo durante la creación del mismo. Las pruebas estuvieron sujetas bajo datos reales y datos no validos para comprobar que esta cumpliera con los requisitos establecidos y las debidas validaciones que debe cumplir.

El sistema no está diseñado para una empresa en particular, así que el mantenimiento del mismo es libre por parte de los diseñadores del software y de las empresas que lo desean adquirir.

5.2.7. Implantación y evaluación de sistema

El sistema se podrá implementar en unidades de auditorías donde se realizan a diarios el tipo de trabajo para el cual está diseñado el sistema, la evaluación del mismo estará en dependencia de la puesta en marcha del sistema en una entidad que se dedica a realizar auditorías de tipo externo o interno.

5.3. Modelo de Auditoría

Los trabajos de auditoría deben planearse adecuadamente para poder alcanzar totalmente los objetivos planteados, para que se puedan obtener eficiencia y eficacia y la debida prontitud.

Para cada auditoría programada, se elaborará un plan de trabajo específico, que incluyan los procedimientos a aplicarse, su alcance, el momento de su aplicación y deberá definirse de acuerdo con la estructura de la organización o práctica, el personal responsable de ejecutarlo.

El plan de auditoría estará basado en la comprensión de las actividades de las entidades.

Un plan de anual de auditoría definirá específicamente todas las auditorías a realizarse en un periodo determinado. Estará definido por:

1- Un plan anual será realizado por el Director General.

2- El plan contendrá información detallada de la auditoría a realizarse como: objetivos, fechas de ejecución, área o empresa donde se ejecutará, estado, horas planificadas, tipo y personal asignado.
3- Una vez registrado un plan no puede ser eliminado, solo puede ser modificado.

Una auditoría puede pertenecer a un o más planes anuales. Una auditoría estará basada en:

1- Una auditoría estará comprendida por fases, cada fase tendrá: fechas reales y planificadas y total de horas.

2- Una auditoría deberá tener un estado, los estados de una auditoría son: no iniciada, iniciada, suspendida y finalizada.

   2.1. Una auditoría esta no iniciada cuando está registrada en un plan, pero aun no está en ejecución.
   2.2. Una auditoría esta iniciada cuando está registrada en plan y se está ejecutando.
   2.3. Una auditoría está suspendida cuando ha iniciado y se detuvo su ejecución.
   2.4. Una auditoría está finalizada cuando todas sus actividades fueron ejecutadas.

3- Una auditoría puede ser realiza únicamente para un área o una entidad.
4- Una auditoría no puede ser eliminada una vez ya almacenada, solo se puede modificar los datos.

Una fase está contenida dentro de una auditoría, las fases contienen detalles de las actividades a ejecutarse, una fase estará basa en:

1- Una fase contendrá fechas reales y planificadas, total de horas de ejecución y las actividades a ejecutarse en cada una de ellas.
2- La cantidad de fases definidas para una auditoría estará en dependencia de las normas que la empresa utilice.

Una actividad o procedimientos pertenecer a una fase en particular, cada actividad deberá tener:

1- Una actividad posee nombre, fechas de inicios y reales, fecha de ejecución, persona encargada en realizar la auditoría.

2- Una actividad debe de realizarse en el periodo que comprende la fase, una actividad no puede pasar se de la fecha de culminación o de inicio de las fechas que corresponde a la fase a la que pertenece.

El personal será contratado por el director general en dependencia de la necesidad que se tenga en una división. El pago por personal será de salario por hora.

El personal de la auditoría cumple diferentes funciones de acuerdo al cargo que desempeñan (Anexos 9.2.2.1):

1- El director general: Es la persona encargada de registrar los planes anuales, las auditorías, personal de trabajo y salario por hora.
2- El director específico: Es la persona encargada de verificar y registrar personal de trabajo de auditoría por divisiones.
3- Supervisor de auditoría: Es la persona encargada de verificar el plan anual, planes de auditorías y avances de las auditorías.
4- Encargado de auditoría: Es la persona encargado de registrar la planeación de auditoría y avance de las sus actividades. Puede realizar una o más actividades simultáneamente en la ejecución de una auditoría.
5- Asistente: Es la persona encargada de verificar la planeación de la auditoría y registra el avance de las actividades. Igual que el encargado puede realizar una o más actividades simultáneamente.
6- Especialista:

La unidad auditada estará en dependencia del tipo de empresa que adquiera el software, es decir una unidad auditada en una firma o unidad de auditoría externa se refiere a la entidad donde se ejecutará la auditoría, mientras si es una firma o unidad de auditoría interna, es el área o lugar dentro de la entidad donde se ejecutará la auditoría.

La unidad auditada contendrá datos referentes al lugar donde se ejecutará la auditoría, los datos de nombre, teléfono, dirección y correos se guardaran como referencia del lugar de ejecución.

La división de auditoría estará organizada de acuerdo a las personas encargadas de realizar las actividades de las auditorías, según la especialización del personal será asignada dentro de las distintas divisiones y por la que serán asignadas para su labor.

Una división tiene un director especifico, también la integran especialistas, encargados, asistentes.

El representante es la persona que provee información de la unidad auditada y a la que representa, brinda la información de los avances de la auditoría y el que a la vez brindara información de la unidad auditada.

El costo total de auditoría estará en dependencia del pago por hora de cada empleado, y por el total de horas en que se ejecutó una auditoría.

Los informes serán impresos en el proceso o culminación de una auditoría o de sus fases o actividades. Ejemplos de reporte (ver Anexos 9.6.7).

5.4. Estudios de factibilidad

A continuación se presenta un breve estudio de factibilidad con objetivo de examinar si es viable la creación del Sistema de Información de Planes Anuales y Trabajos de Auditoria.

5.4.1. Factibilidad técnica

Para implementar el sistema SIAPTRA es necesario contar con un equipo informático con las siguientes características mínimas:

- Hardware

Descripción	Características
Procesador	al menos 1.8 GHz
Memoria RAM	512 MB
Disco Duro	40 Gb
Monitor	15”
Teclado	USB o PS/2
Mouse	USB o PS/2
Impresora	Láser o burbuja
Batería con estabilizador	650VA

- Software

Descripción	Características
Sistema Operativo	Microsoft Windows Xp o Microsoft Windows Vista
Herramienta de programación	Microsoft Visual Studio.Net 2005
Gestor de base de datos:	SQL Server 2000.

5.4.2. Factibilidad operacional

El sistema SIAPTRA está diseñado para ser utilizado por firmas o unidades internas de auditorías, por lo que generalmente, el personal que está previsto a hacer uso del sistema, está compuesto por un Director General que es la máxima autoridad del personal, Directores Específicos de cada división de auditoría, Encargados de auditorías, Supervisores, Asistentes y Especialistas; cada uno de ellos con su nivel de uso dentro del sistema.

5.4.3. Factibilidad económica

Para la implementación de SIAPTRA, se realizaron cotizaciones en varias empresas nacionales, de la que se eligió aquella que ofrecía a un precio más moderado del equipo a utilizar.

Se eligió la cotización (Anexos 9.7) que ofrece la empresa COMTECH que consiste en:

- Una PC con características optimas para crear el Sistema

Descripción	**Características**
Procesador	INTEL CELERON 1.8 GHZ
Memoria RAM DDR2	1Gb
Disco Duro SATA 7200rpm	160Gb
Monitor	AOT CRT 17 ‘’
Teclado	PS/2
Mouse	Óptico PS/2
Impresora	HP DESKJET D1560
Batería con estabilizador	UPS 700V A CDP 400W

Total de equipo cotizado en $ 458.85 con IVA incluido.

- Software que se instalara en la PC para implementar el sistema

Software	*Características*	*Precio*
Sistema Operativo	Microsoft Windows Xp o Microsoft Windows Vista	$150
Herramienta de programación	Microsoft Visual Studio 2005 Standard Edition (Retail)	$180
Gestor de base de datos	Microsoft SQL Server 2000 Standard with Product Key	$99.9

Con un total 429.9 incluyendo la licencia Microsoft

- Gastos totales del equipo de desarrollo

Descripción	**Precio**
Impresiones	$120
Memorias USB 2.0 de 1Gb (3 unidades)	$21
Útiles	$50
Viáticos	$100
Contrato de uso de Internet (4 meses)	$50

Con un gasto total efectuado por el equipo de desarrollo de $341

El estudio de factibilidad económica revela que el sistema SIAPTRA tendría un costo total de **$799.85** sin incluir licencias de Software; o un costo total de **$1229.75** agregando el costo de Licencias de Software.

5.5. Pruebas del software

5.5.1. CASO DE PRUEBA

Descripción del Caso de Prueba

En este caso hacemos una prueba al sistema, el tipo de prueba que se hizo es de entrega.
Las pruebas a realizar en el sistema: **Registrar datos reales de una auditoria.**

1- Datos incorrectos: Valores negativos en las horas de ejecución.
2- Datos incorrectos: Cantidad de horas de ejecución no permitidas.
3- Datos incorrectos: Ingresar letras en las horas de ejecución.
4- Datos incorrectos: No hay valor de horas de ejecución en la Fecha de Inicio Real.
5- Datos incorrectos: No hay valor de horas de ejecución en la Fecha Final Real.

Condición de ejecución

Registrar Datos reales de una auditoria

Una auditoria es registrada en el sistema únicamente por el Gerente General. La auditoria puede pertenecer a uno o más planes dependiendo de la planificación realizada de la misma. Los datos reales se muestran una vez desplegando el plan correspondiente de una auditoria, se muestra en la parte superior de la pantalla con un icono con las letras DR.

Para realizar esta prueba utilizaremos una auditoria que contiene ya una planificación almacenada en el sistema. En este caso usaremos la auditoria con código 2, cuyo inicio planificado es el Moral de compras de cobro de licencia de productos de estereotipos de magnitudes. Cuya fecha de inicio planificado es el 10 de junio del 2010 y cuya fecha final planificada es el 23 de julio del 2010.

1- Datos incorrectos: Valores negativos en las horas de ejecución.

Una vez seleccionadas la fechas Inicio Real y fecha Final Real. Se despliegan el rango de fechas de ejecución que van desde el día de inicio hasta el día final.

- **Precondiciones:** Se desplegaran el rango de las fechas que van desde el día de inicio hasta el día final, en el campo de horas deberá ingresarse las cantidad de horas de ejecución de la actividad realizadas ese día.

- **Valores de entrada:**

    - Introducimos “10/06/2010” en el campo Inicio Real.
    - Introducimos “15/06/2010” en el campo Final Real.
    - Introducimos “-1” en el campo Horas que corresponde al día “10/06/2010”.
    - Pulsamos el botón “Grabar”.
    - El sistema automáticamente cambiará el valor a 0.
    - Pulsamos el botón “Cancelar” y finalizamos el proceso de Horas Reales.

- **Resultados esperados:** El proceso termina y no se almacena los datos reales de la actividad.

- **Postcondiciones:** El sistema permanece en el mismo estado.

2- Datos incorrectos: Cantidad de horas de ejecución no permitidas.

Una vez seleccionadas la fechas Inicio Real y fecha Final Real. Se despliegan el rango de fechas de ejecución que van desde el día de inicio hasta el día final.

- **Precondiciones:** Se desplegaran el rango de las fechas que van desde el día de inicio hasta el día final, en el campo de horas deberá ingresarse las cantidad de horas de ejecución de la actividad realizadas ese día.

- **Valores de entrada:**

    - Introducimos “10/06/2010” en el campo Inicio Real.
    - Introducimos “15/06/2010” en el campo Final Real.
    - Introducimos “9” en el campo Horas que corresponde al día “10/06/2010”.
    - Pulsamos el botón “Grabar”.
    - El sistema automáticamente cambiará el valor a 8, para que pueda aceptar ese valor deberá de indicarlo como una hora extra.
    - Pulsamos el botón “Cancelar” y finalizamos el proceso de Horas Reales.

- **Resultados esperados:** El proceso termina y no se almacena los datos reales de la actividad.

- **Postcondiciones:** El sistema permanece en el mismo estado.

3- Datos incorrectos: Ingresar letras en las horas de ejecución.

Una vez seleccionadas la fechas Inicio Real y fecha Final Real. Se despliegan el rango de fechas de ejecución que van desde el día de inicio hasta el día final.

- **Precondiciones:** Se desplegaran el rango de las fechas que van desde el día de inicio hasta el día final, en el campo de horas deberá ingresarse las cantidad de horas de ejecución de la actividad realizadas ese día.

- **Valores de entrada:**

    - Introducimos “10/06/2010” en el campo Inicio Real.
    - Introducimos “10/06/2010” en el campo Final Real.
    - Introducimos “ddddd” en el campo Horas que corresponde al día “10/06/2010”.
    - Pulsamos el botón “Grabar”.
    - El sistema automáticamente cambiará el valor a 0.
    - Pulsamos el botón “Cancelar” y finalizamos el proceso de Horas Reales.

- **Resultados esperados:** El proceso termina y no se almacena los datos reales de la actividad.

- **Postcondiciones:** El sistema permanece en el mismo estado.

4- Datos incorrectos: No hay valor de horas de ejecución en la Fecha de Inicio Real.

Una vez seleccionadas la fechas Inicio Real y fecha Final Real. Se despliegan el rango de fechas de ejecución que van desde el día de inicio hasta el día final.

- **Precondiciones:** Se desplegaran el rango de las fechas que van desde el día de inicio hasta el día final, en el campo de horas deberá ingresarse las cantidad de horas de ejecución de la actividad realizadas ese día.
- **Valores de entrada:**

    - Introducimos “10/06/2010” en el campo Inicio Real.
    - Introducimos “15/06/2010” en el campo Final Real.

- No introducimos valores en el campo Horas que corresponde al día “10/06/2010”.
- Pulsamos el botón “Grabar”.
- El sistema muestra un mensaje indicándonos que la fecha de Inicio Real debe tener un valor mayor a cero.
- Pulsamos el botón “Cancelar” y finalizamos el proceso de Horas Reales.

- **Resultados esperados:** El proceso termina y no se almacena los datos reales de la actividad.

- **Postcondiciones:** El sistema permanece en el mismo estado.

5- Datos incorrectos: No hay valor de horas de ejecución en la Fecha Final Real.

Una vez seleccionadas la fechas Inicio Real y fecha Final Real. Se despliegan el rango de fechas de ejecución que van desde el día de inicio hasta el día final.

- **Precondiciones:** Se desplegaran el rango de las fechas que van desde el día de inicio hasta el día final, en el campo de horas deberá ingresarse las cantidad de horas de ejecución de la actividad realizadas ese día.

- **Valores de entrada:**

  - Introducimos “10/06/2010” en el campo Inicio Real.
  - Introducimos “15/06/2010” en el campo Final Real.
  - No introducimos valores en el campo Horas que corresponde al día “15/06/2010”.
  - Pulsamos el botón “Grabar”.

- El sistema muestra un mensaje indicándonos que la fecha Final Real debe tener un valor mayor a cero.
- Pulsamos el botón “Cancelar” y finalizamos el proceso de Horas Reales.

- **Resultados esperados:** El proceso termina y no se almacena los datos reales de la actividad.

- **Postcondiciones:** El sistema permanece en el mismo estado.

6. Conclusiones

Un sistema con el que se pueda controlar el cumplimiento de varias auditorías, calcular sus costos preliminares y registrar así mismo el personal disponible para ejecutarlas, es sin duda alguna, un instrumento muy útil para cualquier empresa u organismo que realice auditorías, ya sean de tipo internas o externas.

Gracias a que existen las herramientas necesarias, alcanzamos crear y automatizar un modelo para la administración de planes anuales y trabajos de auditorías mediante un sistema de información, y podemos mencionar que la utilización de este beneficiará a los usuarios, al optimizar el manejo de los datos de las auditorías, además permitiría generar reportes de forma más fácil, rápida y efectiva.

Podemos afirmar que un sistema de este tipo, fortalecería las estrategias de negocio de una empresa, proporcionando más seguridad en la administración de la información y eficiencia en la generación de sus resultados.

7. Recomendaciones

- Para utilizar de forma efectiva este sistema informático es preciso cumplir con los requerimientos de hardware y software que se mencionan en la factibilidad técnica, inciso 5.5; esto es para que no exista ninguna dificultad en el rendimiento de la PC al momento de ejecutar la aplicación.

- Debido a que el sistema utiliza la fecha de la PC local, es necesario tenerla configurado correctamente la fecha y el formato de la fecha (Español (Nicaragua)). *El formato de fecha se configura en Panel de control, Configuración regional y de idioma, Opciones regionales.*

- Para utilizar este Software sin complicaciones en la interfaz, es preferible configurar la resolución de pantalla a 1024 x 768 píxeles o mayor

- Es recomendable que las personas que vayan a utilizar el sistema, consulten el manual de usuario para comprender mejor su funcionamiento y le sirva de ayuda ante alguna dificultad.

8. Bibliografía

[1]. Blanco Luis Miguel
Programación en Visual Basic.NET,
Grupo Eidos.

[2]. González Alvarado
Sistemas de Bases de Datos
Primera edición, 1996
Editorial Tecnológica de Costa Rica

[3]. Hynes Richard
Programación de bases de datos con Visual BASIC .net, primera edición 2003
Pearson Prentice Hall

[4]. Kendall & Kendall
Análisis y diseño de sistemas, Sexta Edición, 2005
Prentice Hall Hispanoamericana S.A.

[5]. Larman Craig
UML y Patrones, Introducción al análisis y diseño orientados a objetos, Primera Edición, 1999
Pearson Prentice Hall

[6]. Pressman Roger
Ingeniería del Software: Un enfoque práctico, Quinta Edición, 2002
MacGraw Hill.

[7]. Schmuller Joseph
Aprendiendo UML en 24 Horas
Prentice Hall

[8]. Sommerville Ian
Ingeniería del software, Séptima Edición, 2005
Pearson educación.

[9]. Compendio Normativo de la Contraloría General de la República de Nicaragua.

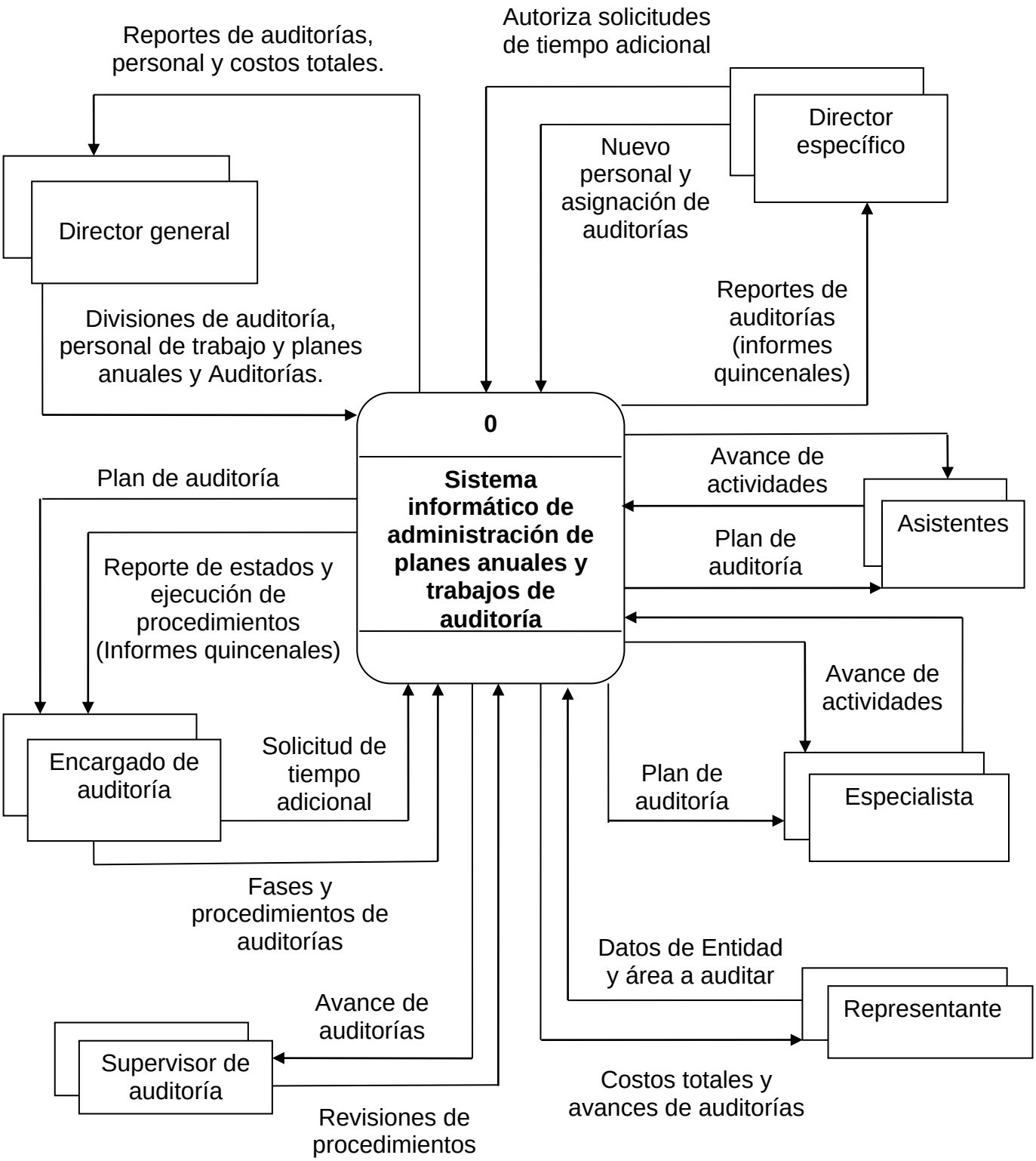
[10]. Contraloría General de la República, Manuales de Auditoria Gubernamental- Manual Básico de Auditoria, Parte Nº VII, Auditoria de sistemas de información, Agosto, 2006

Web Grafía

1. Formas Normales, Bases de Datos. Encontrado en el sitio web: www.microtecnologico.com/Main/FormasNormalesBasesDatos

Anexos

9.1. Diagrama de flujo de datos (Nivel de contexto)



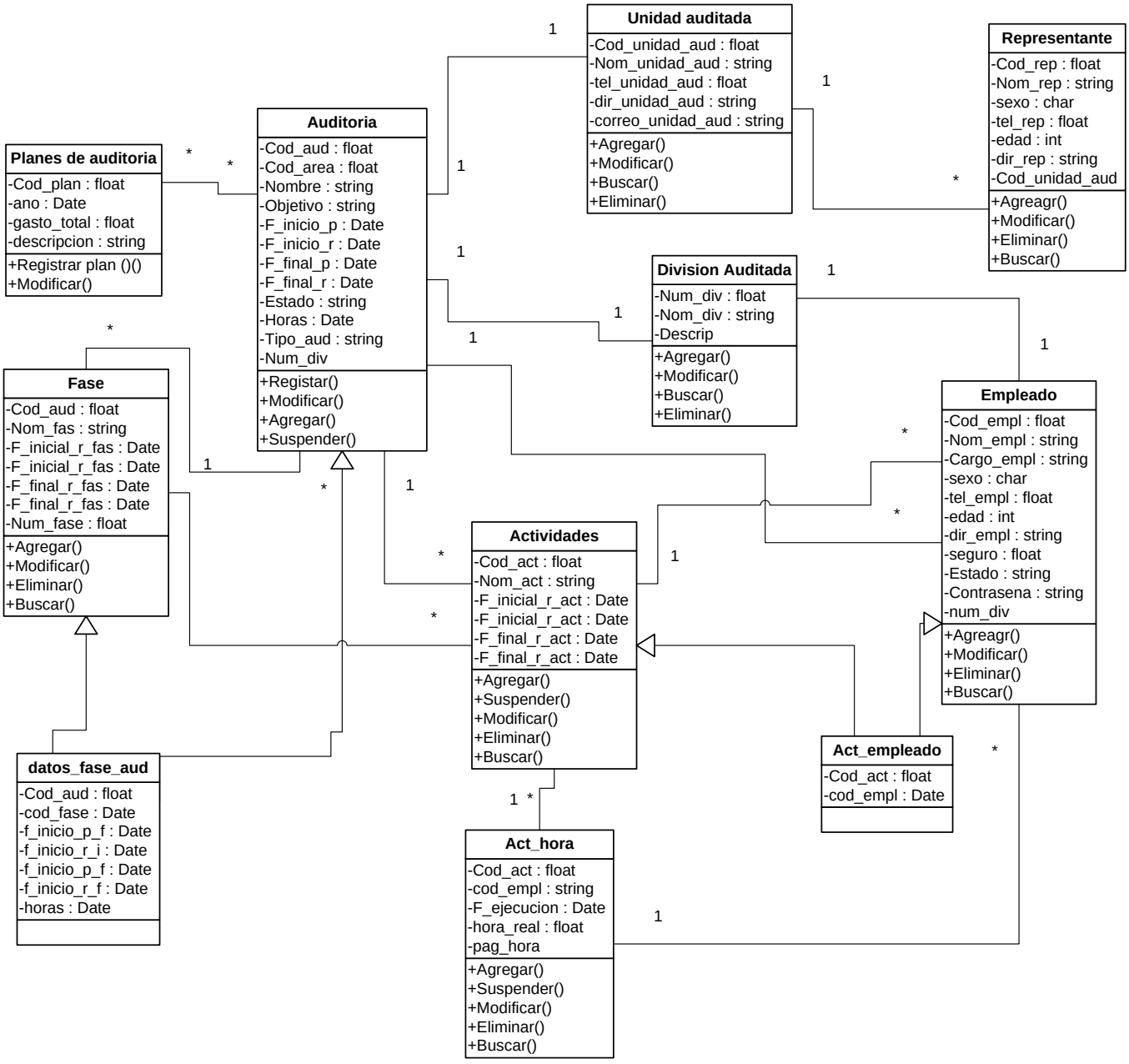
En este diagrama de nivel de contexto podemos verificar el flujo de información que entra y sale del sistema y quienes envían o reciben datos.

Las entidades que envían información y de alguna manera se ven beneficiados por el uso y resultados del sistema están: Director general, Director específico, Supervisor de auditoría, Encargado de auditoría, asistentes, especialistas y representantes de entidades a auditar.

En el diagrama podemos verificar que el Director general ingresa las divisiones de auditoría y su personal de trabajo, los planes anuales y auditorías a realizarse, Obtiene reportes de auditorías, personal y costos totales. El Director específico agrega nuevo personal en su división de trabajo, asigna personal de trabajo para realizar las auditorias y solicita permiso para autorizar solicitudes de tiempo adicional, obtiene reportes de auditoría (informes quincenales). El encargado de auditoría ingresa las fases y procedimientos que se ejecutarán en la auditoría, verifica el plan de auditoría, estados y ejecución de procedimientos y solicita tiempo adicional para llevar a cabo la auditoria. El supervisor de auditoría ingresa revisiones de procedimientos y verifica avances de auditorías. El asistente y especialista registran y verifican avances de actividades (procedimientos), también verifican el plan de la auditoria. El representante provee información de la Entidad a auditar y obtiene reportes de costos totales y avances de auditoría.

9.2. Modelado Unificado UML

9.2.1. Diagrama de clases



En este diagrama se representan las clases que el sistema utiliza en el modelaje del sistema.

9.2.2. Diagrama de casos de uso

9.2.2.1. Actores

Actor	Descripción
Director General	Encargado de registrar planes anuales, las auditorías, personal de trabajo y salario por hora.
Director específico	Encargado de verificar y registrar personal de trabajo de auditoría por divisiones.
Supervisor de auditoría	Verifica el plan anual, planes de auditorías y avances de las auditorías
Encargado de auditoría	Encargado de registrar la planeación de auditoría y avance de las fases y sus actividades.
Asistente	Verifica la planeación de la auditoría y registra el avance de las actividades.
Especialista	Verifica la planeación de la auditoría y registra el avance de las actividades.
Representante	Provee información de las Unidades y áreas a ser auditadas y obtiene reportes de costo totales y avances de auditoría

9.2.2.2. Casos de Uso

Casos de Usos	Descripción
Registrar una nueva división de auditoría	Con el fin de organizar el personal de trabajo se registrarán divisiones para la elaboración de trabajos de auditorías.
Agregar un nuevo personal a la división	Una vez registrada la división se podrá agregar personal que trabajará en esa división.
Agregar director específico a división	Cuando se crea una nueva división se podrá agregar un director específico para que controle esa división
Trasladar personal a otra división	Un personal podrá ser trasladado, cuando se necesite que labore en otra división de auditoría.
Registrar datos del nuevo personal de auditoría	Cuando se agrega un personal a la división se registran sus datos.
Asignar cargo al personal	Cuando se agrega un personal a la división se registrará el cargo que este tendrá.
Registrar Entidad	Cuando se lleva a cabo una auditoría se debe registrar los datos de la entidad con el cual se trabajará.
Registrar áreas de Entidad	Cuando se registra una Entidad se debe registrar las áreas a ser auditada de esa entidad.
Registrar representantes de Entidad	Cuando se registra una Entidad se debe registrar las representantes de esa entidad.

Casos de Usos	Descripción
Añadir un nuevo plan anual	Se registran los planes anuales de auditorías que realizará la entidad.
Añadir un nuevo plan de auditoría	Cuando se va a realizar una auditoría se debe registrar la planeación.
Asignar personal a las auditoría	Cuando se lleva a cobo una auditoría se debe asignar un personal de trabajo que la realizará.
Registrar salario por hora	Se registrará el salario por hora que se le pagará al personal de trabajo de la auditoría.
Registrar fechas de la fases de auditoría	Cuando se registra nuevo plan de auditoría se debe registrar fechas planificadas de auditorías
Registrar actividades de las auditorías	Cuando se registra un nuevo plan de auditoría se debe registrar actividades de la auditoría.
Asignar personal a las actividades	Cuando se lleva a cobo una auditoría se debe asignar un personal de trabajo que realizará las actividades.
Registrar avances de las auditorías	Se debe registrar las fechas reales conforme se vayan ejecutando las actividades y fases de las auditorías.

Casos de Usos	Descripción
Imprimir reportes de personal por auditoría	Se obtendrán reportes detallados del personal de trabajo de cada auditoría
Calcular costo total por auditoría	El sistema calculará el costo total por cada auditoría que realice la entidad.
Verificar costo total por auditoría	Se podrá emitir un costo total y detallado por cada auditoría
Verificar plan Anual	Se podrá verificar todas las auditorías que realiza la entidad anualmente.
Verificar avances de la auditoría	Mediante la representación de gráficos porcentuales se verificará el avance de las auditorías.
Verificar avances de las actividades de las auditorías	Mediante la representación de gráficos porcentuales se verificará el avance de las actividades de las auditorías.
Verificar planes de auditoría	Se verificará los planes de cada una de las auditorías.

9.2.2.3. Diagrama de casos de usos General: SIAPTRA

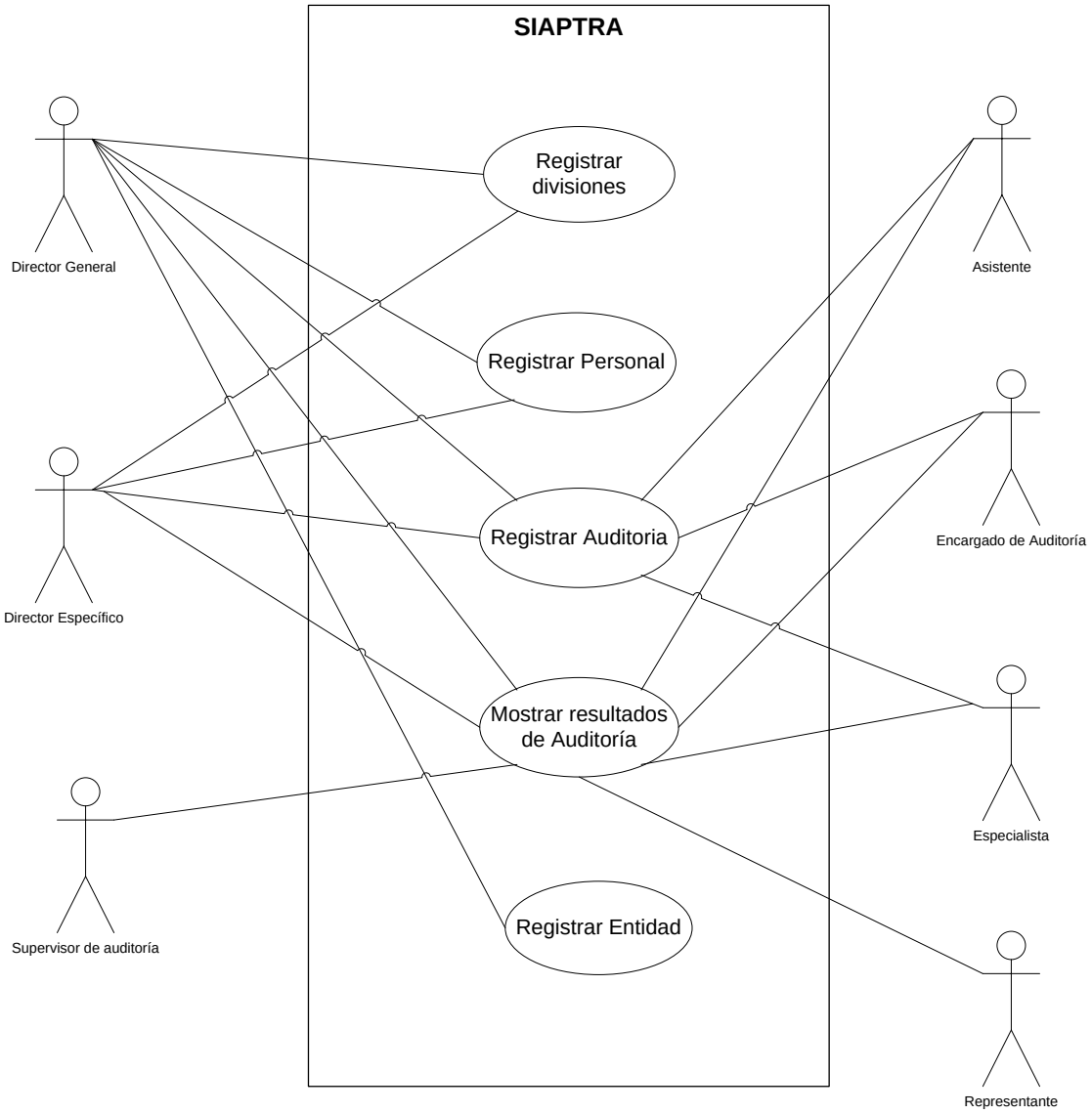
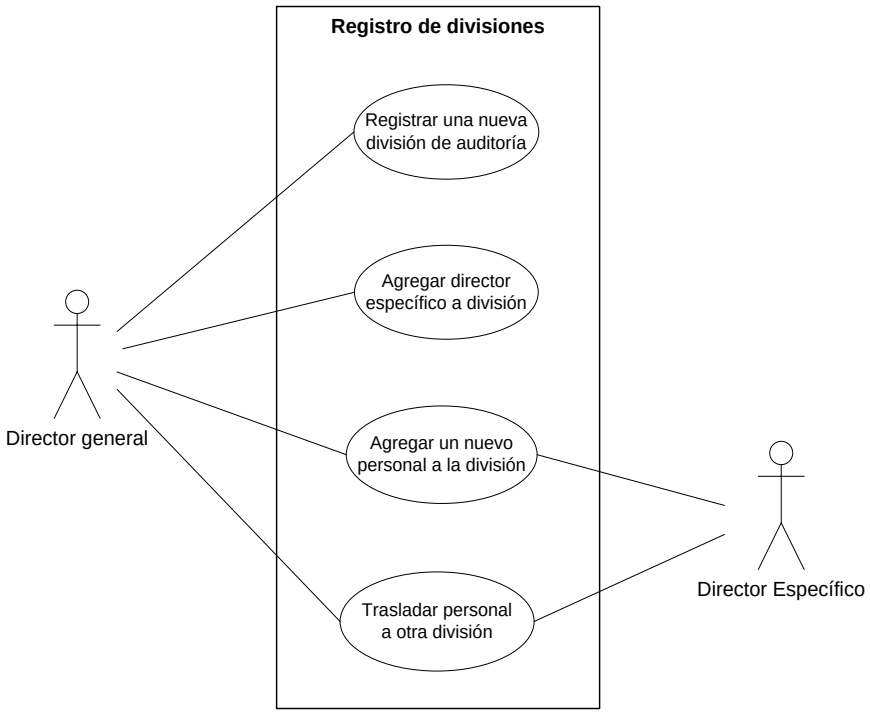


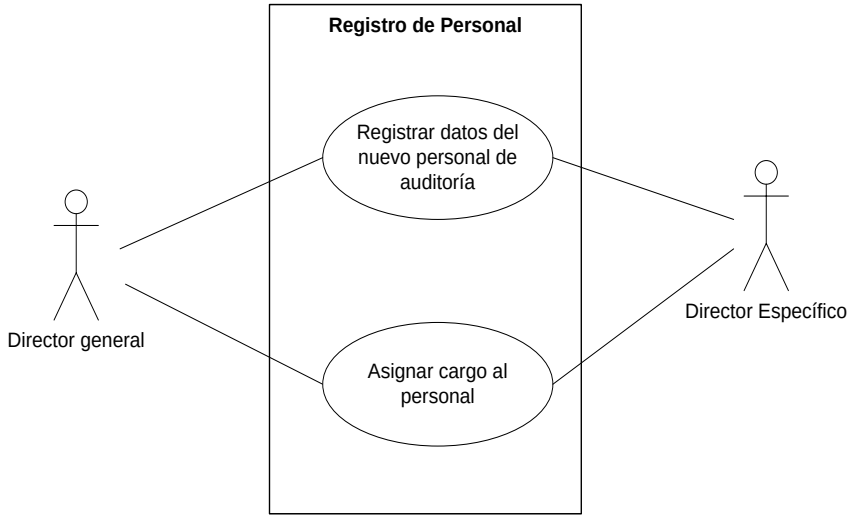
Diagrama de casos de uso general de SIAPTRA para registrar divisiones, personal, auditorias, entidades y mostrar resultados de auditoría.

9.2.2.4. Diagrama de casos de usos: Registro de divisiones



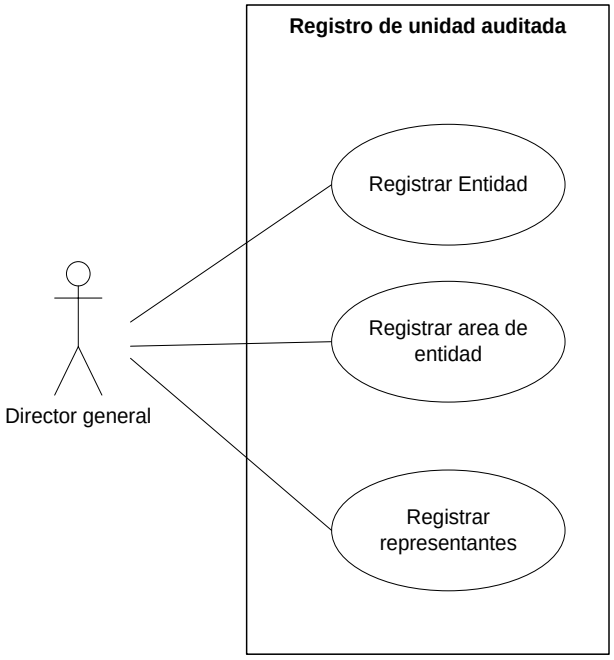
Para organizar el personal de la entidad, se divide en grupos de trabajos para realizar auditorías. El director general es el encargado de registrar esas divisiones y su personal de trabajo (director específico, encargados de auditorías, asistentes, especialistas y supervisores), también el director específico puede agregar personal a su división siempre y cuando cuente con la aprobación del director general. También se podrá trasladar a un personal de una división a otra cuando se necesite.

9.2.2.5. Diagrama de caso de uso: Registro de Personal



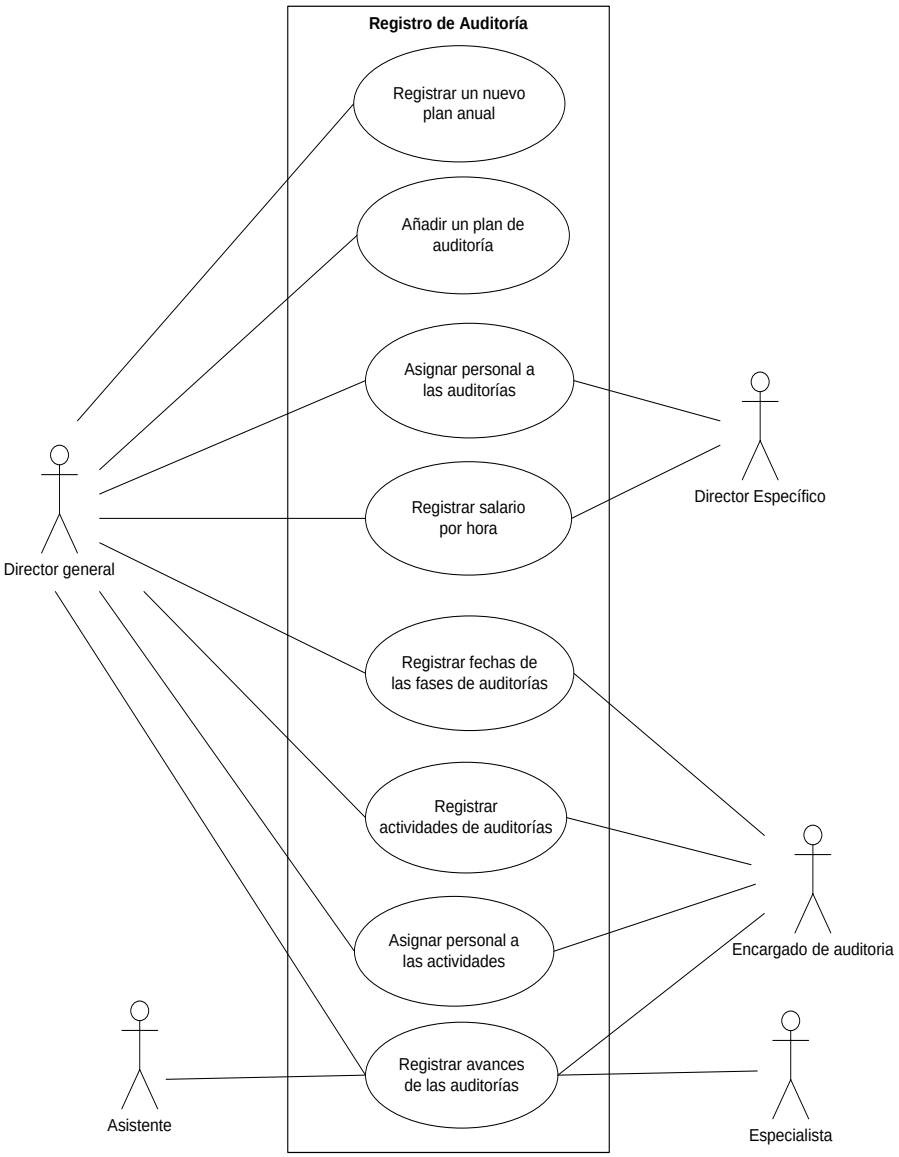
Cuando el director general o director específico agrega un personal a la división, se debe registrar sus datos personales y cargo que este tendrá.

9.2.2.6. Diagrama de caso de uso: Registro de Entidad



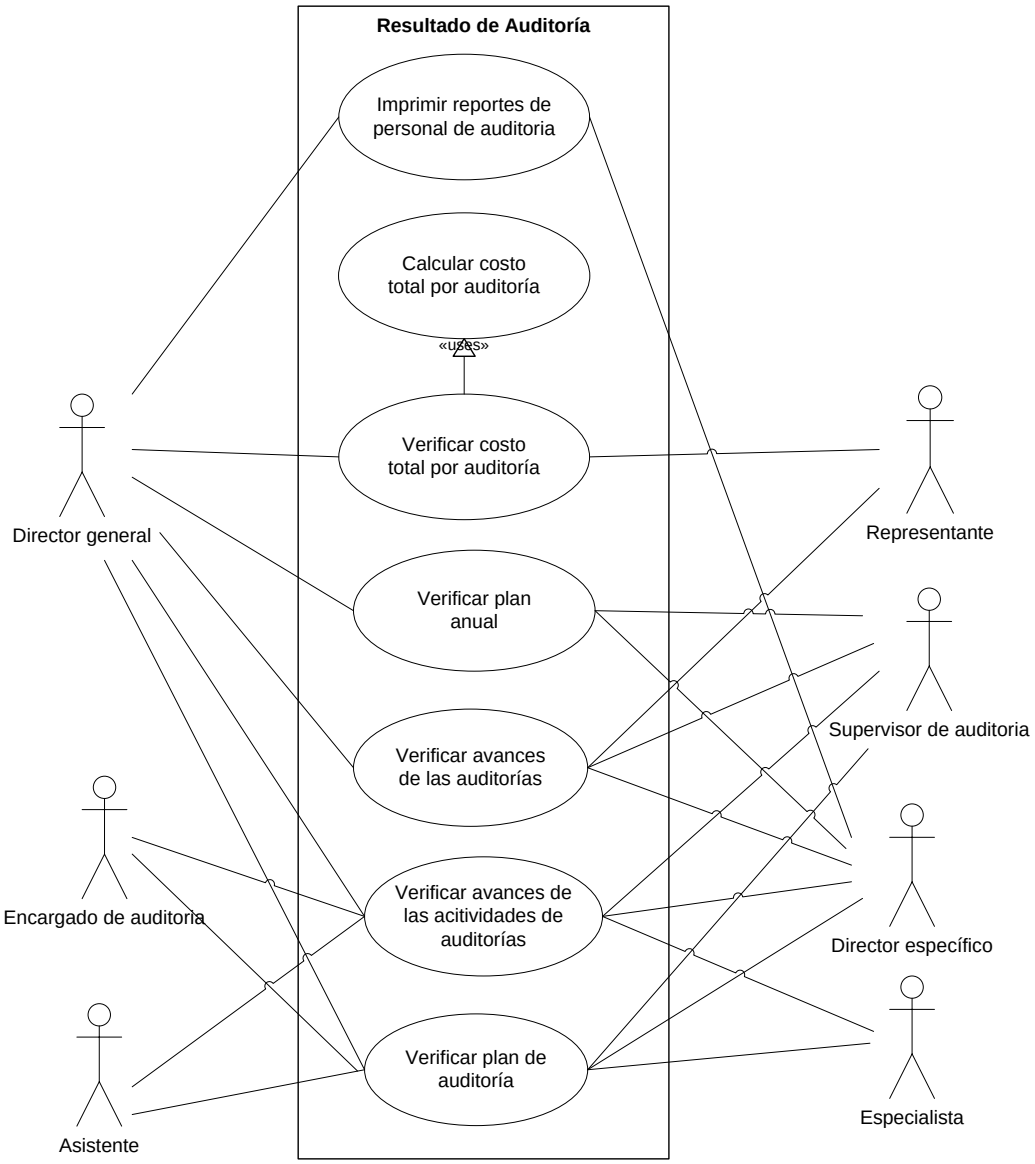
Para realizar auditorías, el director general debe registrar los datos de las entidades y sus áreas a ser auditadas así como también los representantes de esta.

9.2.2.7. Diagrama de casos de uso: Registro de Auditoría



El director general debe registrar el plan anual a llevarse a cabo, para luego agregar planes de auditoría, una vez agregado tanto él como el director especifico podrán asignar un personal al plan de auditorías (encargado, asistentes, supervisor, especialista) y registrar el salario por hora que se le pagará, el encargado registrará fechas para sus fases así como también actividades y encargados de ejecutarlas. Cabe aclarar que el director general también puede realizar estas operaciones.

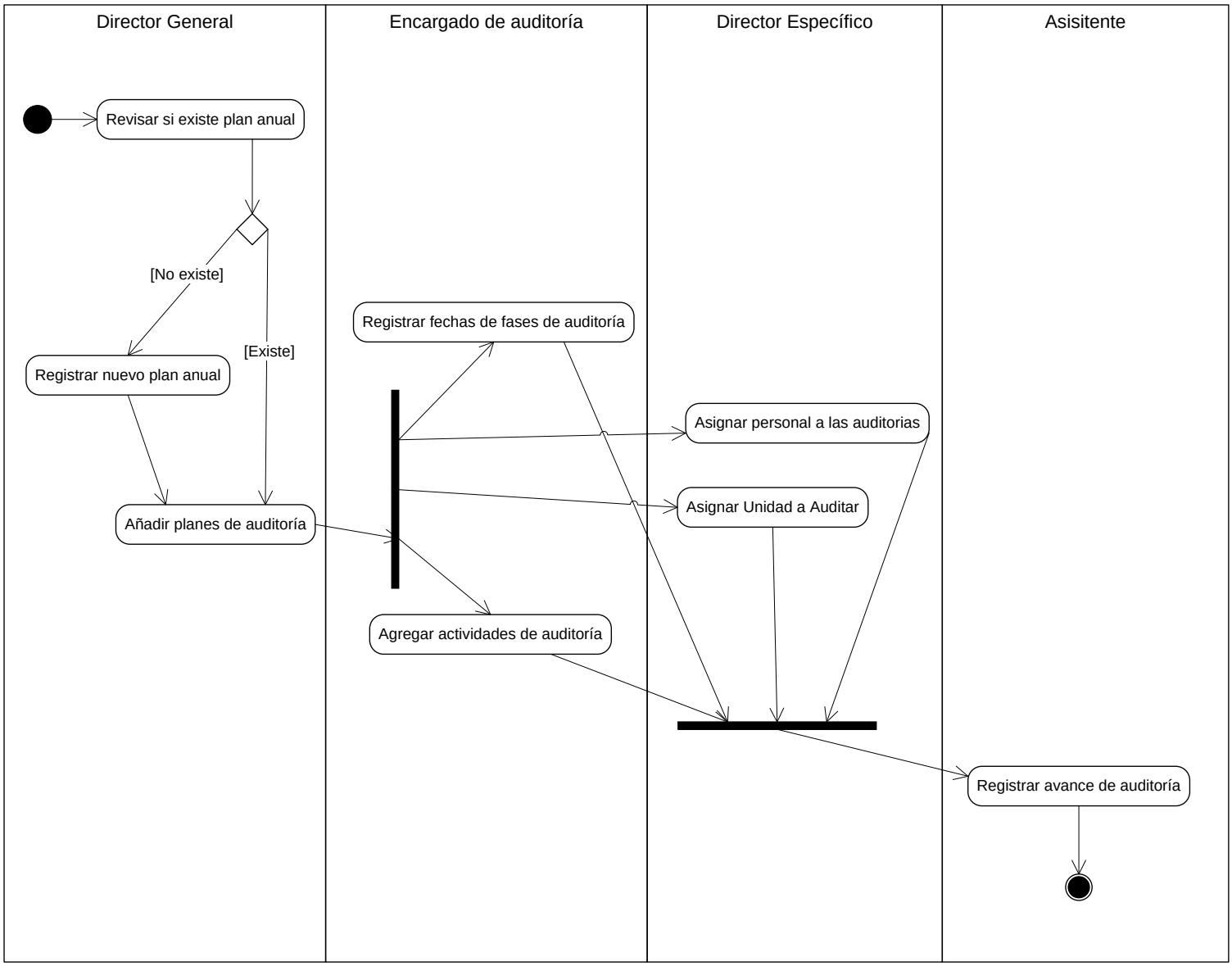
9.2.2.8. Diagrama de caso de uso: Resultado de auditoría



El director general podrá obtener reportes detallados del personal que trabaja en la entidad también podrá verificar los planes anuales y costos totales por auditoría. El director específico y supervisor de auditoría podrán verificar planes anuales y de auditorías, avances de estas y sus actividades. El encargado, asistentes y especialista podían verificar el plan de auditoría y avances de las actividades. Los representantes de las entidades auditadas obtendrán reportes de costos totales y avances de auditorías.

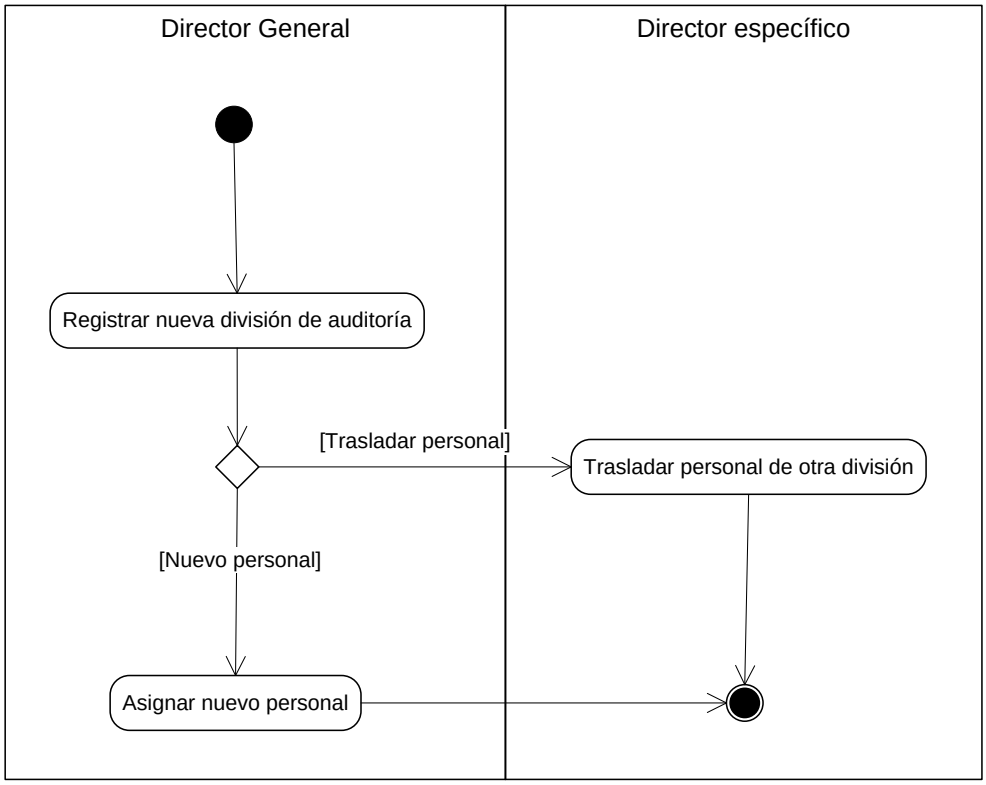
9.2.3. Diagramas de actividades

9.2.3.1. Diagramas de actividades: Registro de Auditoría



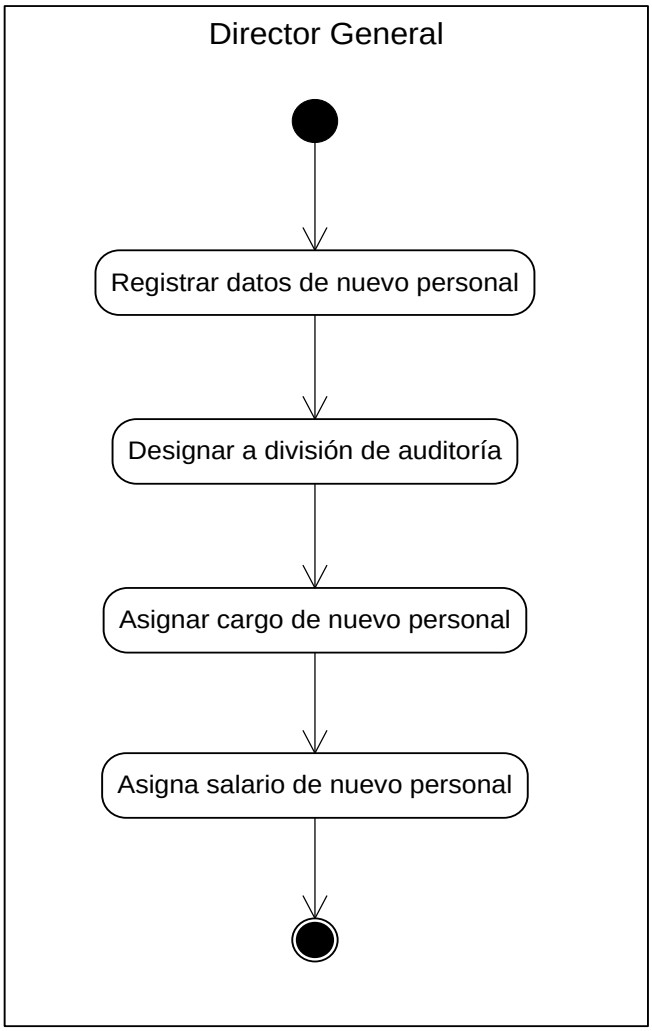
Para registrar una auditoría, primero verificamos si existe un plan anual acorde al año planificado, si no la hay, el director general crea uno, si lo hay este mismo ya puede registrar una nueva planificación de auditoría, luego se puede asignar las fechas de las fases, el personal, entidad a auditar, crear actividades correspondientes. Una vez hecho todo esto se procede a registrar el avance de las auditoras.

9.2.3.2. Diagramas de actividades: Registro de división de auditoría



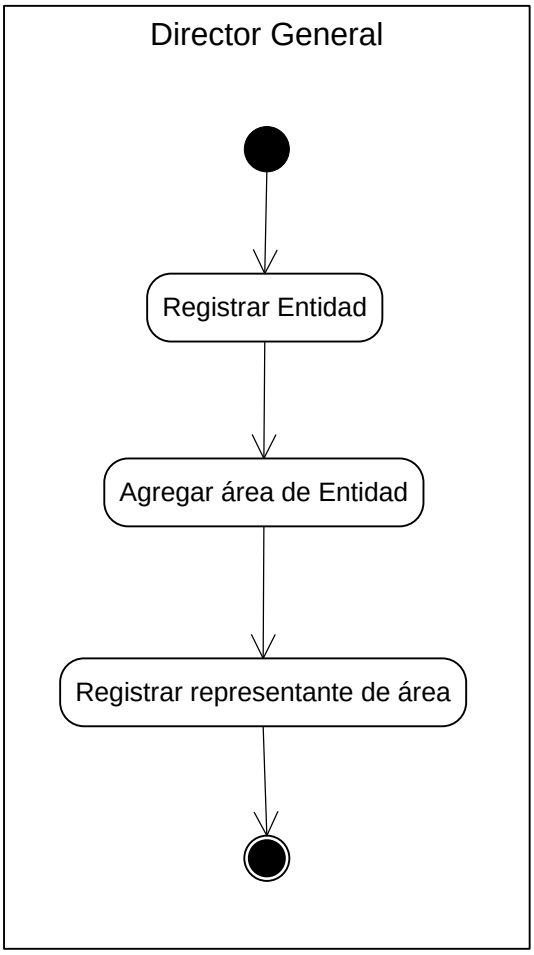
Se registra una división de auditoría, luego se contrata personal para asignarlo a esta división y/o se traslada personal de otra división existente.

9.2.3.3. Diagramas de actividades: Registro de personal



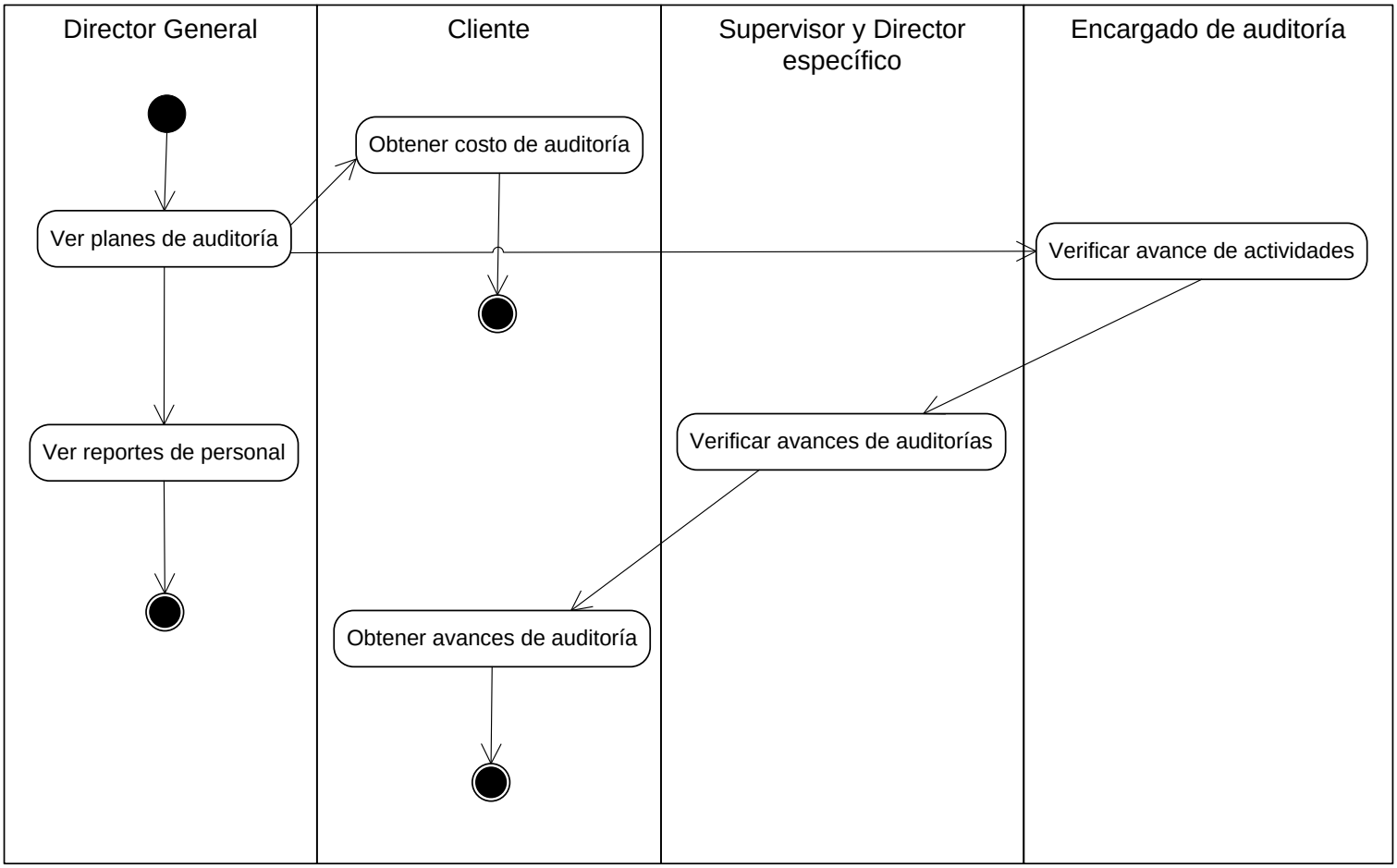
Para añadir personal, se registran los datos personales de este, luego se designa a la división a la que pertenecerá en caso de ser empleado, se le asigna un cargo y su salario.

9.2.3.4. Diagramas de actividades: Registro de Entidad



El director general registra una nueva Entidad, agrega las áreas que la componen y un representante por cada área de la entidad.

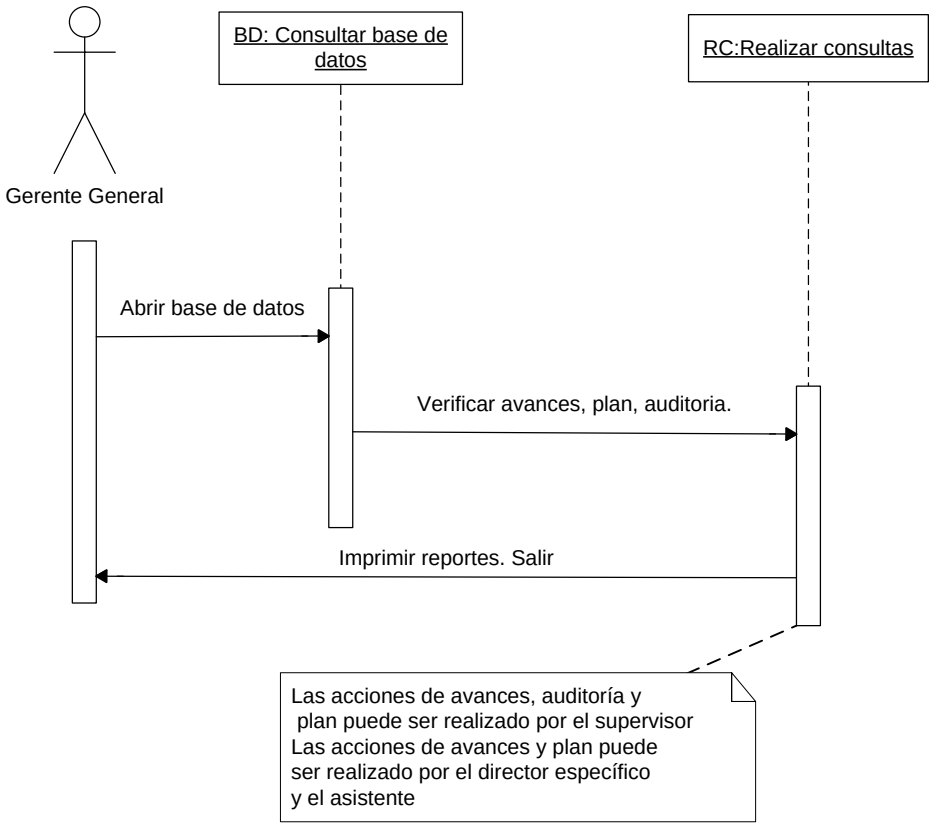
9.2.3.5. Diagramas de actividades: Resultados de auditoría



El director general puede imprimir reportes del personal, enviar al cliente un costo preliminar de la auditoría y enviar reportes del avance de la auditoría.

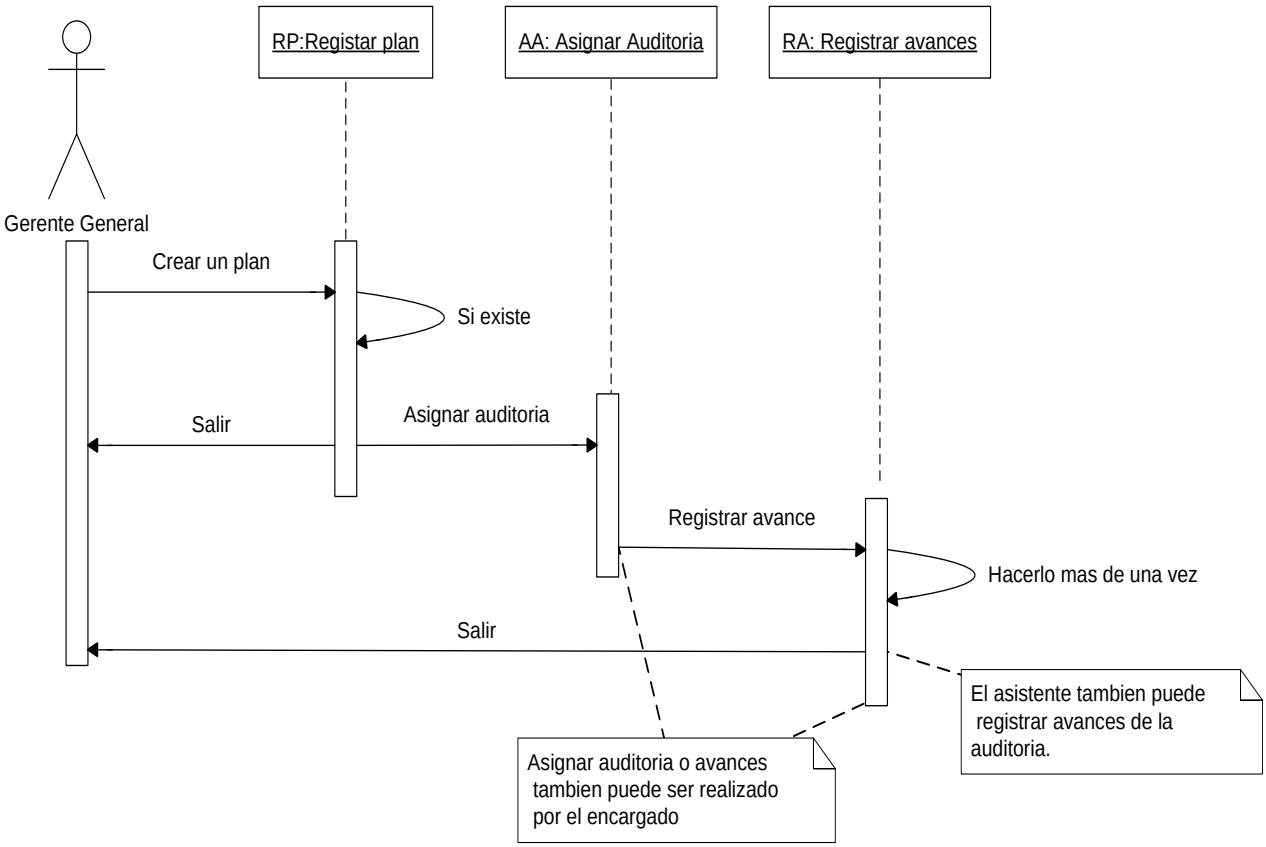
9.2.4. Diagrama de secuencia

9.2.4.1. Diagrama de secuencia: Resultados de auditoría



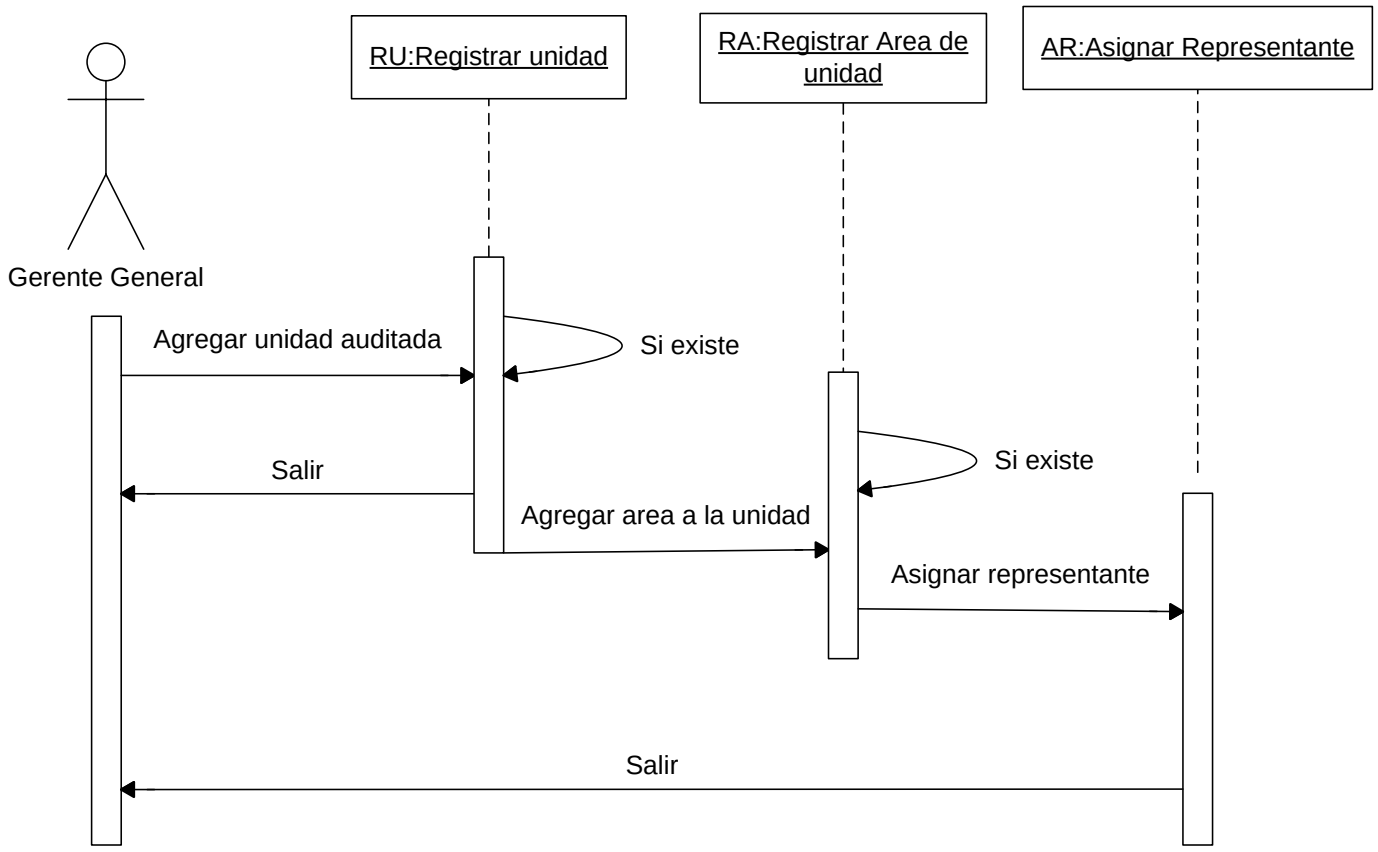
El diagrama de secuencia representa las acciones que el gerente general puede realizar en el sistema, en este caso hace consultas en la base de datos en tal punto verifica los avances de auditorías, planes y actividades.

9.2.4.2. Diagrama de secuencia: Registrar un plan y su correspondiente avance.



Permite crear un plan únicamente por el superusuario, en este caso el gerente general, quien puede asignarle directamente las auditorías a ejecutarse, así como también los avances de la misma.

9.2.4.3. Diagrama de secuencia: Registra una auditoría, para luego asignarle las correspondientes actividades.



En este diagrama registra se registra la auditoría en el sistema, se verifica su existencia y se le asigna las personas involucradas en ella.

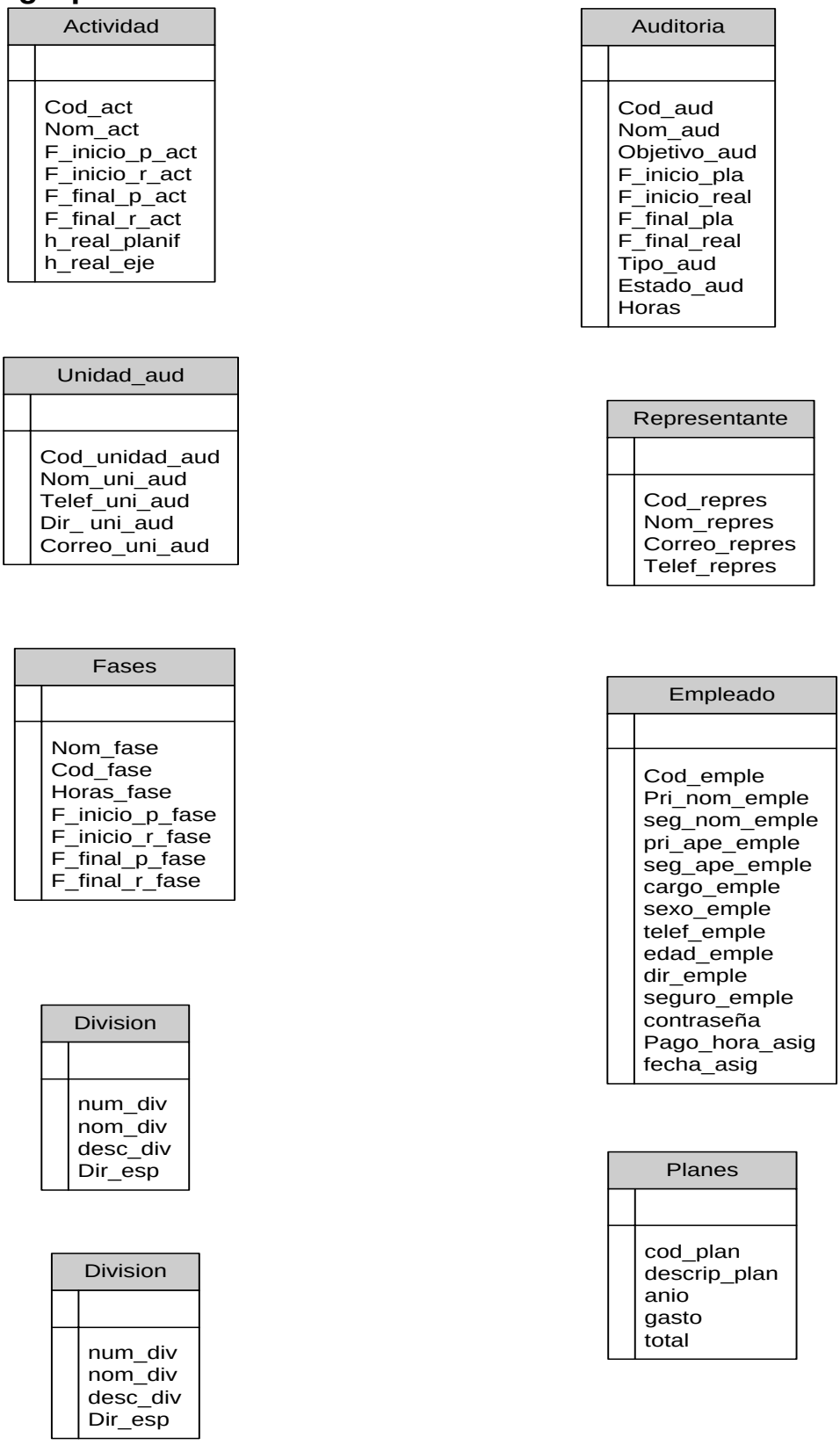
9.3. Normalización de la base de datos

9.3.1. Atributos de la Base de Datos

Cod_plan
Anio
Gasto_total
Descrip_plan
Cod_aud
Nom_aud
Objetivo_aud
Cod_div
F_inicio_pla
F_inicio_real
F_final_pla
F_final_real
Estado_aud
Cod_area
Horas
Tipo_aud
Cod_aud
Cod_area
Nomb_area
Cod_unidad_aud
Descrip_area
Cod_unidad_aud
Nom_uni_aud
Telef_uni_aud
Dir_ uni_aud
Correo_uni_aud
num_div
Nom_div
desc_div
Cod_repres
Nom_repres
Correo_repres
Telef_repres
Cod_unidad_aud
Nom_fase
Cod_fase
Horas_fase
F_inicio_p_fase
F_inicio_r_fase
F_final_p_fase
F_final_r_fase

Nom_act
F_inicio_p_act
F_inicio_r_act
F_final_p_act
F_final_r_act
Cod_act
Cod_emple
Pri_nom_emple
seg_nom_emple
pri_ape_emple
seg_ape_emple
cargo_emple
sexo_emple
telef_emple
edad_emple
dir_emple
seguro_emple
estado
contrasena
h_real_planif
h_real_eje
pago_hora
fecha_asignacion
Pago_hora_asig
fecha_asig
fecha_hist

9.3.2. Agrupación de los Atributos de la Base de Datos



9.3.3. Aplicación de la Primera Forma Normal (1FN)

- En esta etapa de normalización se crean las claves primarias y se verifican que todos los atributos sean atómicos.

Actividad	
PK	**Cod_act**
	Nom_act F_inicio_p_a F_inicio_r_a F_final_p_a F_final_r_a cod_aud nom_aud num_fase login nom_emp h_real_planif h_real_eje

Unidad_aud	
PK	**Cod_unidad_aud**
	Nom_uni_aud Telef_uni_aud Dir_ uni_aud Correo_uni_aud Cod_rep nom_rep ape_rep correo_rep telef_rep

Act_hora	
PK **PK**	**cod_act** **fecha_ejec**
	h_real_eje h_real_planif pago_hora login

Historial_emp	
PK **PK** **PK**	**Cod_emple** **Cod_div** **fecha_hist**

Auditoria	
PK	**Cod_aud**
	Nom_aud Objetivo F_inicio_p F_inicio_r F_final_p F_final_r Tipo_aud Estado Horas_pla cod_area num_div

Fases	
PK	**num_fase**
	nom_fase des_fase

Auditor_audi	
PK **PK**	**Cod_aud** **login**
	pago_h fecha_asig nom_aud

Plan_aud	
PK **PK**	**Cod_plan** **Cod_aud**

Fases_aud	
PK **PK**	**cod_aud** **num_fase**
	f_inicial_p f_fin_p f_inicial_r f_final_r horas_pla horas_reales

Area	
PK	**Cod_area**
	nom_area telef ubicacion cod_unid_aud nom_unid_aud

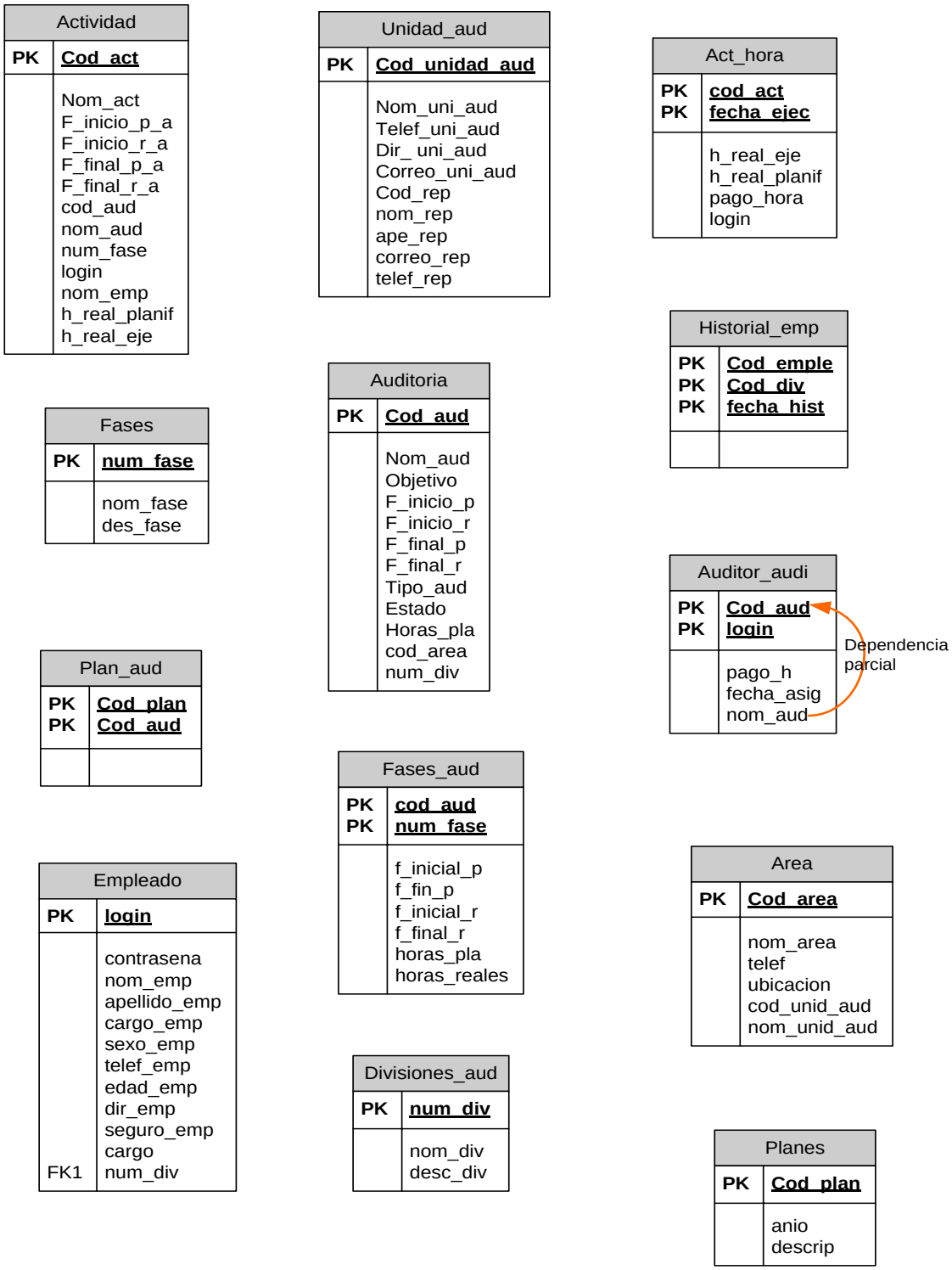
Empleado	
PK	**login**
	contrasena nom_emp apellido_emp cargo_emp sexo_emp telef_emp edad_emp dir_emp seguro_emp cargo
FK1	num_div

Divisiones_aud	
PK	**num_div**
	nom_div desc_div

Planes	
PK	**Cod_plan**
	anio descrip

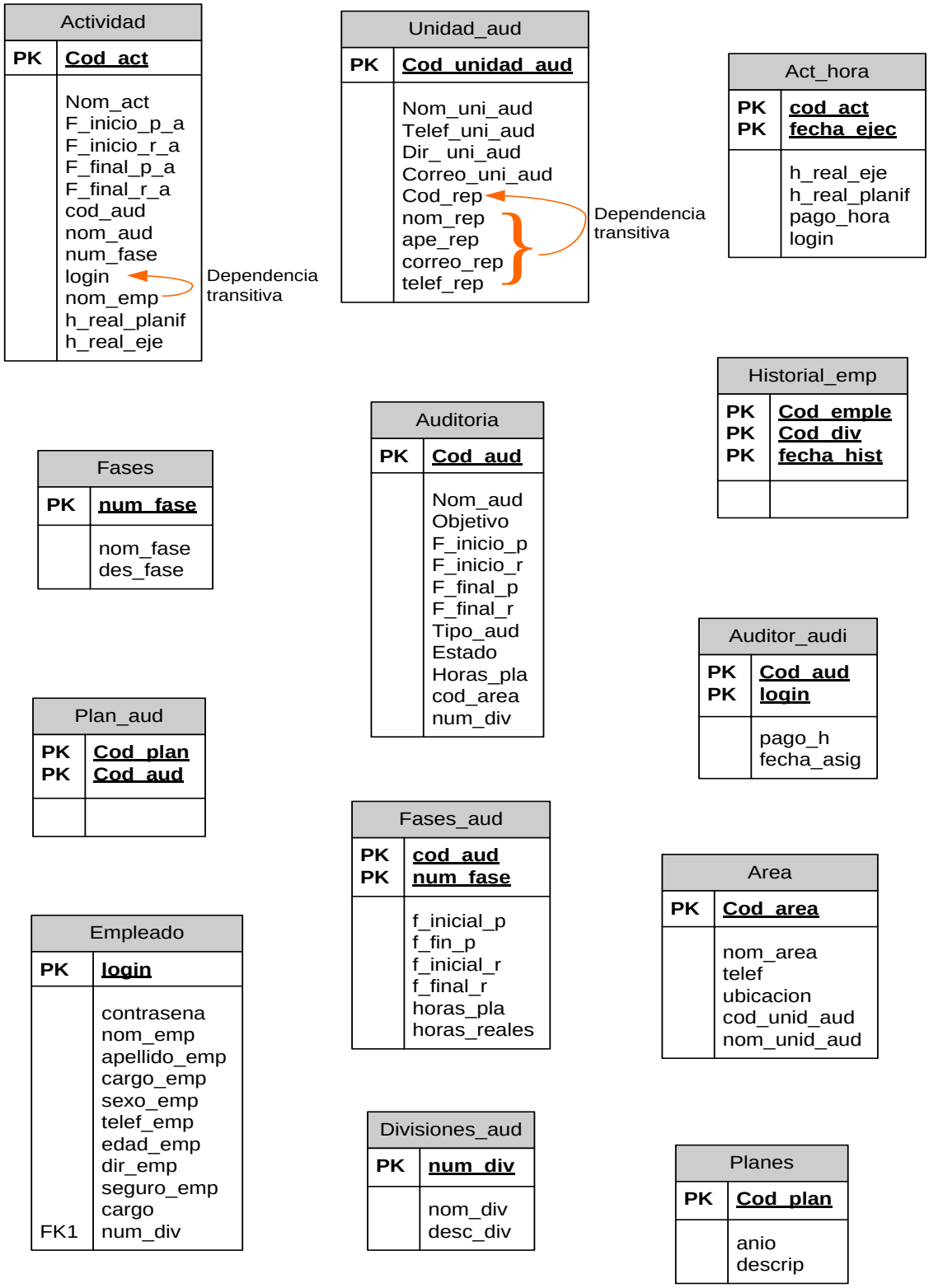
9.3.4. Aplicación de la Segunda Forma Normal (2FN)

En esta fase se debe eliminar la dependencia parcial.



9.3.5. Aplicación de la Tercera Forma Normal (3FN)

Esta fase requiere la eliminación de la dependencia transitiva.



9.3.6. Tablas están en Tercera Forma Normal (3FN)

Plan_aud	
PK,FK1 **PK,FK2**	**cod_aud** **cod_plan**

Planes	
PK	**Cod_plan**
	anio descrip

Unidad_auditada	
PK	**cod_unidad_aud**
	nom_unidad_aud telef_unidad_aud dir_unidad_aud correo_unidad_aud

Area	
PK	**cod_area**
   FK1	nom_area telef_area ubicacion cod_unidad_aud

Representante	
PK	**cod_rep**
    FK1	nom_rep ape_rep correo telef_rep cod_unidad_aud

Auditoria	
PK	**cod_aud**
	nom_aud objetivo f_inicio_p f_inicio_r f_final_p f_final_r estado horas_pla tipo_aud cod_area
FK2	num_div

Auditor_auditoria	
PK,FK2 **PK,FK1**	**login** **cod_aud**
	pago_h fecha_asignacion

Fases	
PK	**num_fase**
	nom_fase des_fase

Empleado	
PK	**login**
	contrasena nom_emp apellido_emp cargo_emp sexo_emp telef_emp edad_emp dir_emp seguro_emp cargo
FK1	num_div

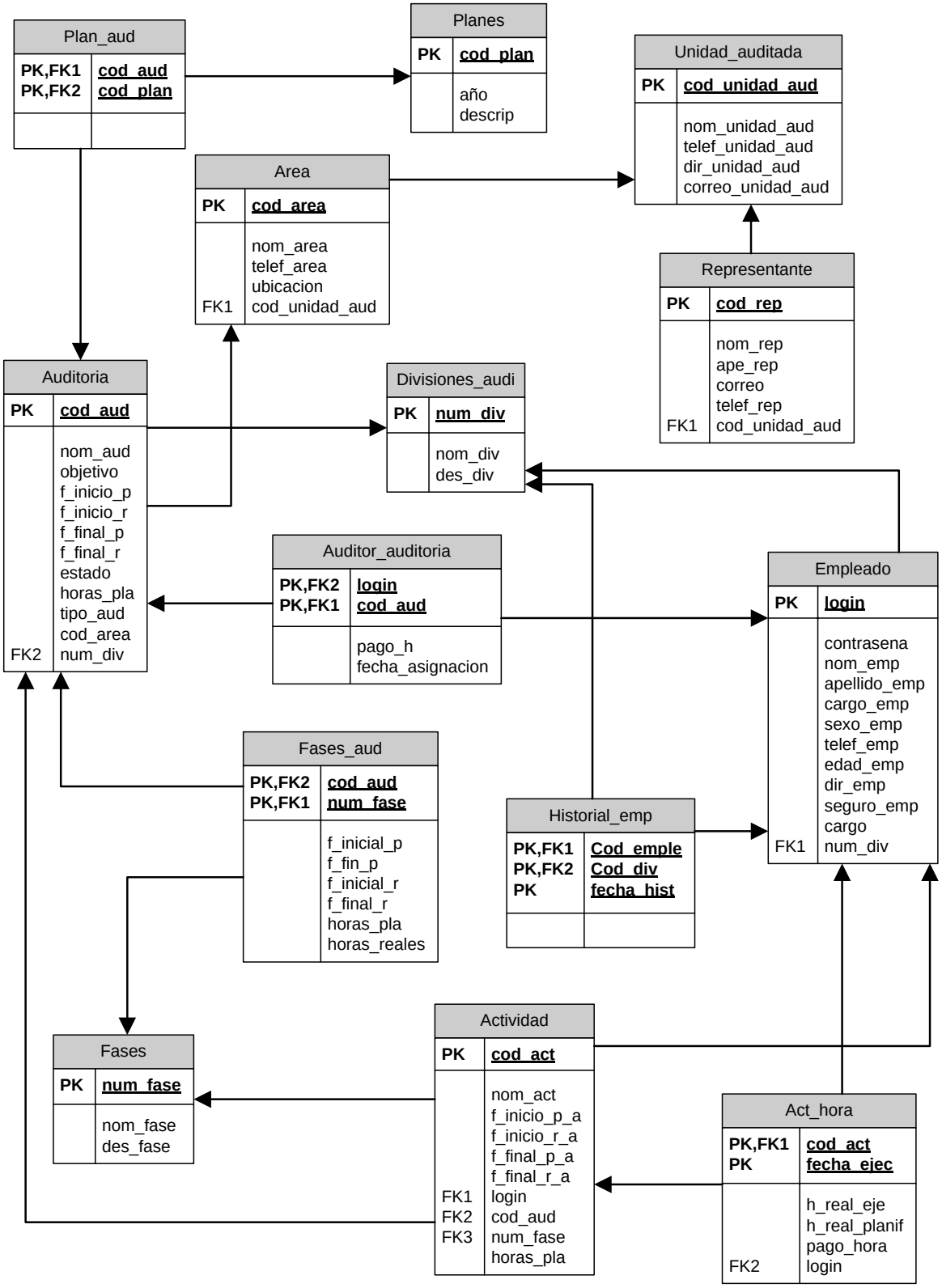
Fases_aud	
PK,FK2 **PK,FK1**	**cod_aud** **num_fase**
	f_inicial_p f_fin_p f_inicial_r f_final_r horas_pla horas_reales

Historial_emp	
PK,FK1 **PK,FK2** **PK**	**Cod_emple** **Cod_div** **fecha_hist**

Actividad	
PK	**cod_act**
     FK1 FK2 FK3	nom_act f_inicio_p_a f_inicio_r_a f_final_p_a f_final_r_a login cod_aud num_fase horas_pla

Act_hora	
PK,FK1 **PK**	**cod_act** **fecha_ejec**
   FK2	h_real_eje h_real_planif pago_hora login

9.4. Base de Datos



9.

9.5. Diccionario de datos

Tabla	Nombre del campo	Tipo de dato	Longitud	Descripción
Actividad	**cod_act**	float	8	Es un código que representa a cada una de las actividades que se está realizando de en cada una de las fase de una auditoría.
	f_inicio_p_act	datetime	8	Describe la fecha planificada para el inicio de una actividad en una fase de auditoría.
	f_inicio_r_act	datetime	8	Describe la fecha real de inicio de una actividad en una fase de auditoría.
	f_final_p_act	datetime	8	Describe la fecha planificada para finalizar una actividad en una fase de la auditoría.
	f_final_r_act	datetime	8	Describe la fecha real en que se finalizó una actividad en una fase de la auditoría.
	login	varchar	50	Código único que representa a un Empleado que realiza esta actividad.
	horas_pla	datetime	8	Total de horas previstas para realizar esta actividad.
	horas_real	datetime	8	Total de horas reales en que se realizó esta actividad.
	cod_fase_aud	numérico	16	Código de fase y auditoría al que pertenece esta actividad.
	nom_act	varchar	100	Nombre de la actividad que se va a ejecutar.
Act_hora	**cod_act**	float	8	Es un código que representa a cada una de las actividades
	login	varchar	50	Código único que representa a un Empleado que realiza esta actividad.
	fecha_ejec	datetime	8	Fecha que se asignó una actividad a un empleado.
	h_real_eje	datetime	8	Hora real en que se realiza la actividad.

Tabla	Nombre del campo	Tipo de dato	Longitud	Descripción
Act_hora	**h_real_planif**	datetime	8	Hora planificada en que se realizará la actividad.
	pago_hora	float	5	cantidad de dinero que se a pagar por hora de trabajo
Area	**cod_area**	float	8	Representa un código del área donde se ejecutará la auditoría.
	cod_entidad	float	32	Representa un código para la entidad donde se va a realizar la auditoría
	nom_area	varchar	8	Nombre del área donde se va a realizar la auditoría.
	ubicacion	varchar	300	Ubicación de la organización.
Auditoria	**cod_aud**	float	8	Es un valor único asignado a una auditoría.
	cod_area	float	8	Representa un código del área donde se ejecutará la auditoría.
	estado_aud	varchar	20	Define el estado en que una auditoría puede estar.
	f_inicio_pla	datetime	8	Describe la fecha que se planifica para el inicio de la auditoría.
	f_inicio_real	datetime	8	Describe la fecha en que inicio realmente la auditoría.
	f_final_pla	datetime	8	Describe la fecha que se planifica para finalizar la auditoría.
	f_final_real	datetime	8	Describe la fecha en que finaliza realmente la auditoría.
	horas	datetime	8	Define el total de horas a utilizar en una auditoría determinada.
	nom_aud	varchar	100	El nombre de la auditoría que se va a realizar.
	num_div	float	8	Representa un código para la división de auditoría que le corresponde ejecutar la auditoría.
	objetivo_aud	varchar	100	Describe los objetivos para la auditoría a realizarse.
	tipo_aud	varchar	20	Especifica el tipo de auditoría que se va a ejecutar en un área de una entidad.

Tabla	Nombre del campo	Tipo de dato	Longitud	Descripción
Division	**desc_div**	varchar	100	Breve descripción de una división de auditoría.
	nom_div	varchar	25	Nombre de la división de auditoría.
	num_div	float	8	Código único que representa una división de auditoría.
Entidad	**cod_entidad**	float	8	Representa un código para la unidad o el lugar donde se va a realizar la auditoría, puede ser una empresa, o un departamento.
	correo_entidad	varchar	32	Es el correo de la empresa o área que será auditada.
	dir_entidad	varchar	100	Define la dirección donde se encuentra la empresa o el área donde se va ejecutar la auditoría.
	nom_entidad	varchar	30	Especifica el nombre del lugar donde se va a ejecutar la auditoría, puede ser una empresa, o área.
	telef_entidad	numérico	8	El número telefónico de la entidad.
Fases	**num_fase**	float	8	Es un código que representa a cada una de las fases de la auditoría.
	nom_fase	varchar	20	Nombre de las fases de las auditorías.
Fase_aud	**cod_aud**	float	8	Es un valor único asignado a una auditoría.
	num_fase	float	8	Es un código que representa a cada una de las fases de la auditoría.
	cod_fase_aud	numérico	16	Código único de una fase dentro de una auditoría.
	f_inicial_p	datetime	8	Describe la fecha que se planifica para el inicio de una fase de la auditoría.
	f_final_p	datetime	8	Describe la fecha que se planifica para el finalizar una fase de la auditoría.
	f_inicial_r	datetime	8	Describe la fecha en que realmente fue iniciada esta fase.

Tabla	Nombre del campo	Tipo de dato	Longitud	Descripción
Fase_aud	**f_final_r**	datetime	8	Describe la fecha en que realmente concluyo esta fase de auditoría.
	horas_fase	datetime	8	Define el número de horas asignada para cada fase de la auditoría.
Historial_pers	**num_div**	float	8	Código único que representa una división auditoría.
	login	varchar	50	Código que representa a un Empleado.
	fecha_hist	datetime	8	Fecha de historial en que se realiza el cambio de división de un empleado.
Personal	**cargo**	varchar	50	Define el cargo que desempeña una persona en la auditoría.
	contrasena	varchar	32	Contraseña que tiene el empleado para acceder al sistemas.
	dir	varchar	200	Dirección del empleado.
	edad	entero	3	Edad del empleado.
	estado	varchar	15	El estado del empleado, puede ser activo o no activo
	login	varchar	50	Código único que representa a cada Empleado
	num_div	float	8	Numero de la división a la que pertenece el personal.
	pri_ape	varchar	50	Primer apellido del empleado.
	pri_nom	varchar	32	Primer nombre del empleado
	seguro	numérico	6	Código de seguro del empleado.
	seg_ape	varchar	50	Segundo apellido del empleado.
	seg_nom	varchar	32	Segundo nombre del empleado.
	sexo	char	1	Tipo de sexo del empleado.
	telef	numérico	32	Teléfono del empleado

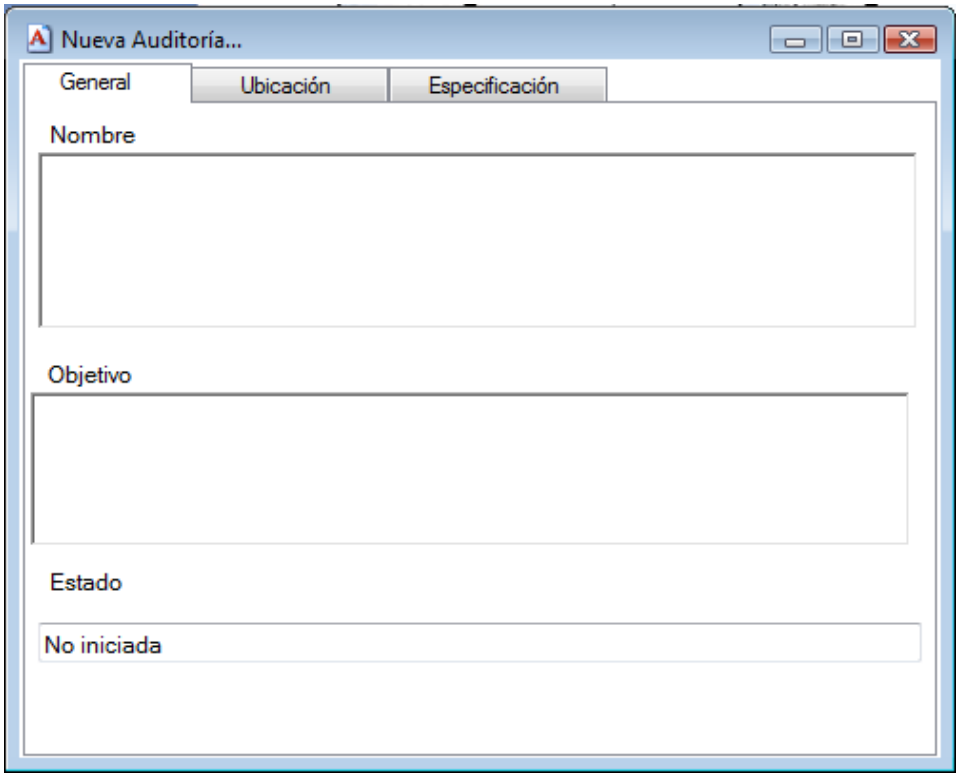
Tabla	Nombre del campo	Tipo de dato	Longitud	Descripción
Personal_aud	**cod_aud**	float	8	Es un valor único asignado a una auditoría.
	login	varchar	50	Código que representa a los Empleados que realizan esta auditoría.
	fecha_asig	datetime	8	Fecha en que se asigna un empleado a una auditoría.
	pago_hora_asig	datetime	3	Cantidad de dinero a pagar por hora de trabajo a un auditor, en una actividad.
Plan	**anio**	datetime	8	Describe el año en el que está realizando el plan que se va a ejecutar.
	cod_plan	float	8	Representa un código para un plan definido, un plan es único por año.
	descrip_plan	varchar	500	En este campo se menciona los objetivos del plan a ejecutarse.
	gasto_total	float	8	Define el total de dinero invertido en las auditorías que se encuentran asignadas a un plan.
Plan_aud	**cod_aud**	float	8	Es un valor único asignado a una auditoría.
	cod_plan	float	8	Representa un código para un plan definido, un plan es único por año.
Representante	**cod_repres**	float	8	Es un código para la persona que representan la unidad o área donde se va ejecutar la auditoría.
	correo_repres	varchar	32	El correo del representante de la unidad auditada.
	nom_repres	varchar	50	Nombre del representante de la unidad auditada.
	telef_repres	numérico	8	Número telefónico del representante de la unidad o área.
	cod_entidad	float	8	Código de la entidad representada

9.6. Pantallas

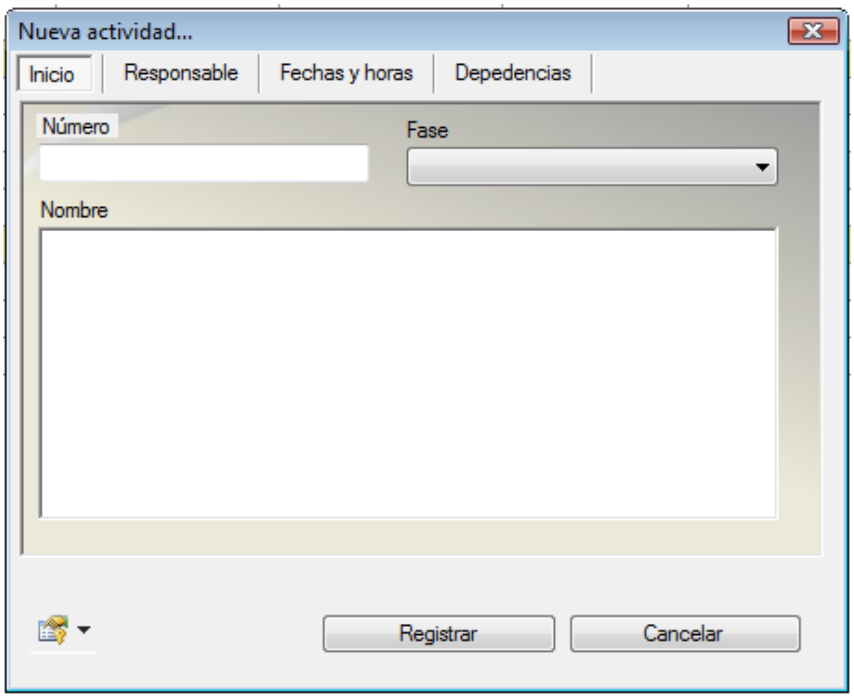
9.6.1. Pantalla de Acceso.

9.6.2. Pantalla Principal.

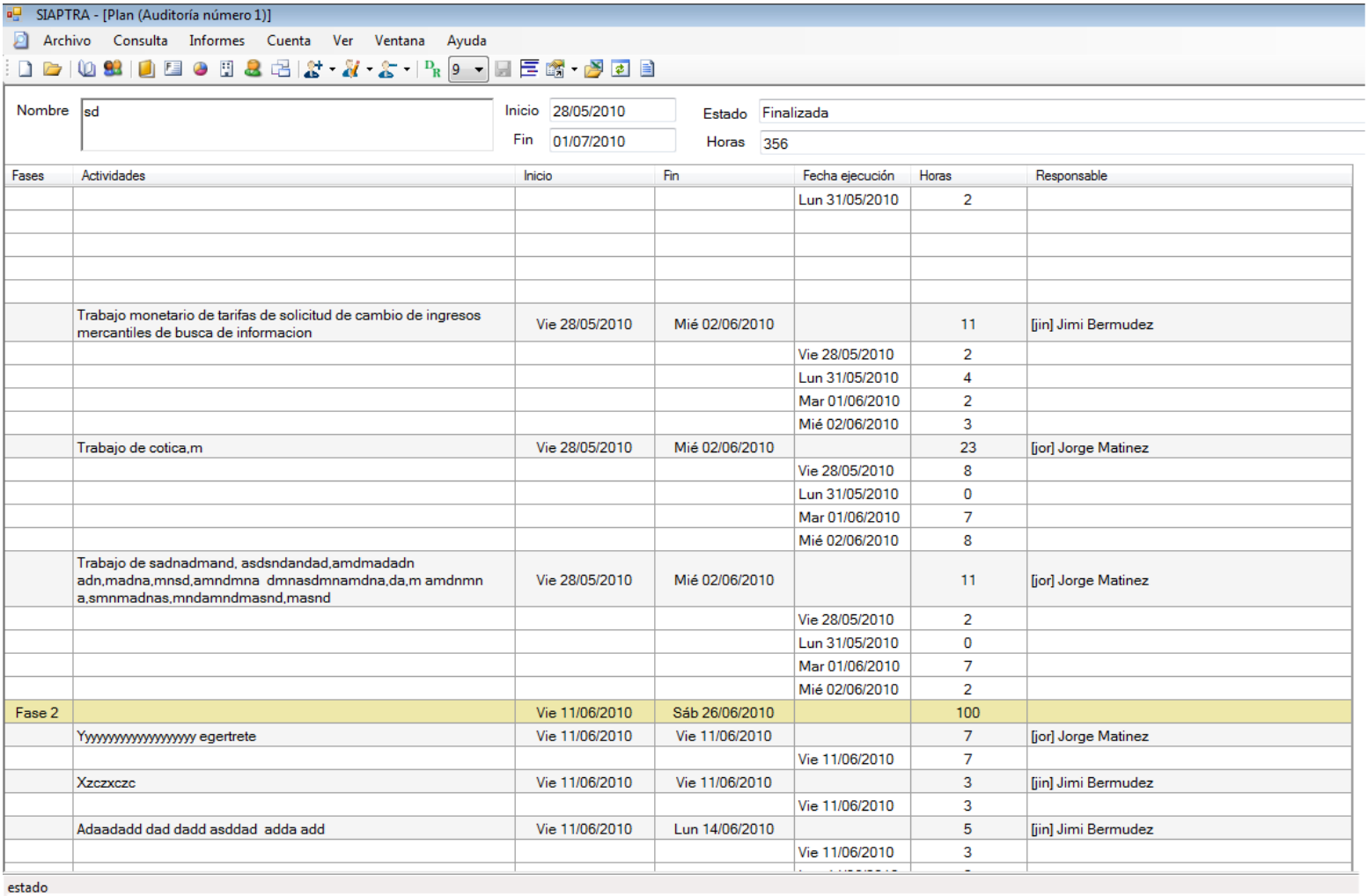
9.6.3. Agregar una Auditoria



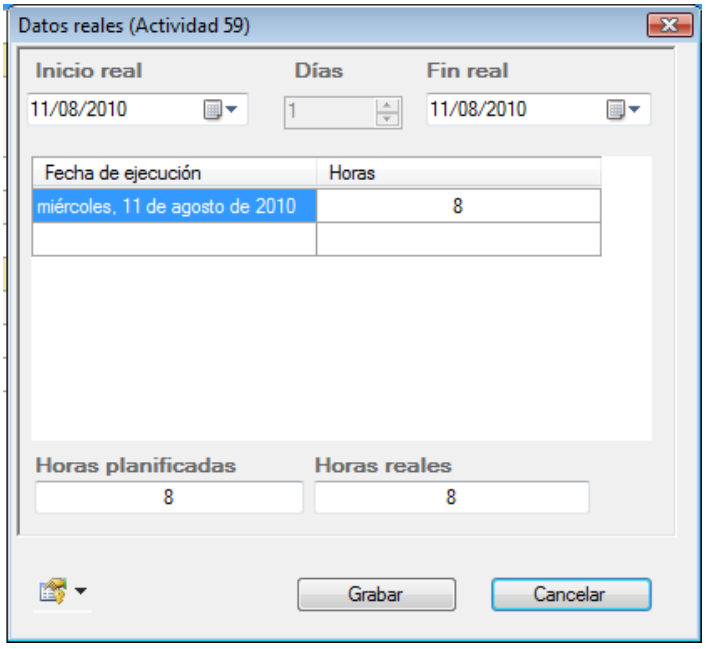
9.6.4. Nueva Actividad



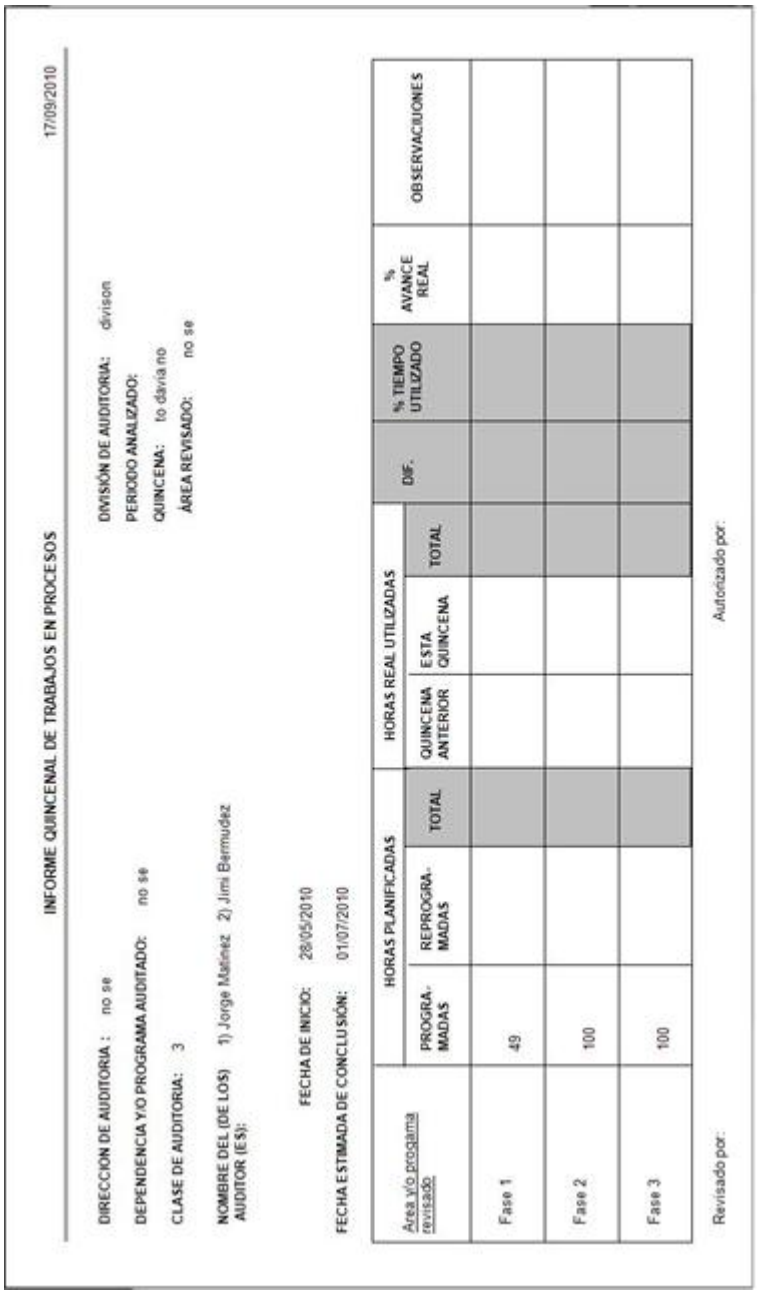
9.6.5. Planes de Auditoria



9.6.6. Datos Reales



9.6.7. Reporte de Informe quincenal de trabajos procesados



INFORME QUINCENAL DE TRABAJOS EN PROCESOS

17/09/2010

DIRECCION DE AUDITORIA : no se

DIVISIÓN DE AUDITORIA: divison

DEPENDENCIA Y/O PROGRAMA AUDITADO: no se

PERIODO ANALIZADO:

CLASE DE AUDITORIA: 3

QUINCENA: to davia no

ÁREA REVISADO: no se

NOMBRE DEL (DE LOS) AUDITOR (ES): 1) Jorge Martinez 2) Jimi Bermudez

FECHA DE INICIO: 28/05/2010

FECHA ESTIMADA DE CONCLUSIÓN: 01/07/2010

Area y/o progama revisado	HORAS PLANIFICADAS			HORAS REAL UTILIZADAS			DIF.	% TIEMPO UTILIZADO	% AVANCE REAL	OBSERVACIONES
	PROGRAMADAS	REPROGRAMADAS	TOTAL	QUINCENA ANTERIOR	ESTA QUINCENA	TOTAL				
Fase 1	49									
Fase 2	100									
Fase 3	100									

Revisado por:

Autorizado por:

9.6.8. Datos de una Auditoria

Auditoría 17/09/2010

Número: 2

Nombre: Mejoramiento ititucionalde estado de comercio electronico

Objetivo: Tratar de mejorar los cobros de datos de apertura de licencia de mercancias de capitulos de estados de cuentas por cobrar

Estado: No iniciada

Plan Anual:

División: Informatica

Periodo auditado

Inicio: 31/05/2010

Fin: 17/07/2010

Inicio real:

Fin real:

Horas planificadas: 150

Horas Reales: 0

Tipo de auditoría: Auditoría fiscal

Unidad auditada

Entidad UNAN

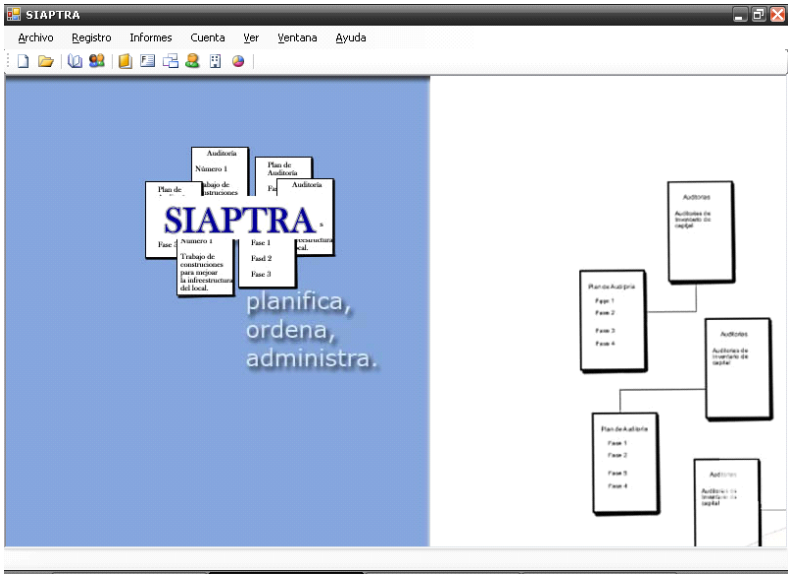
Area UNEN

9.7. Manual de usuario

Siaptra

Bienvenido

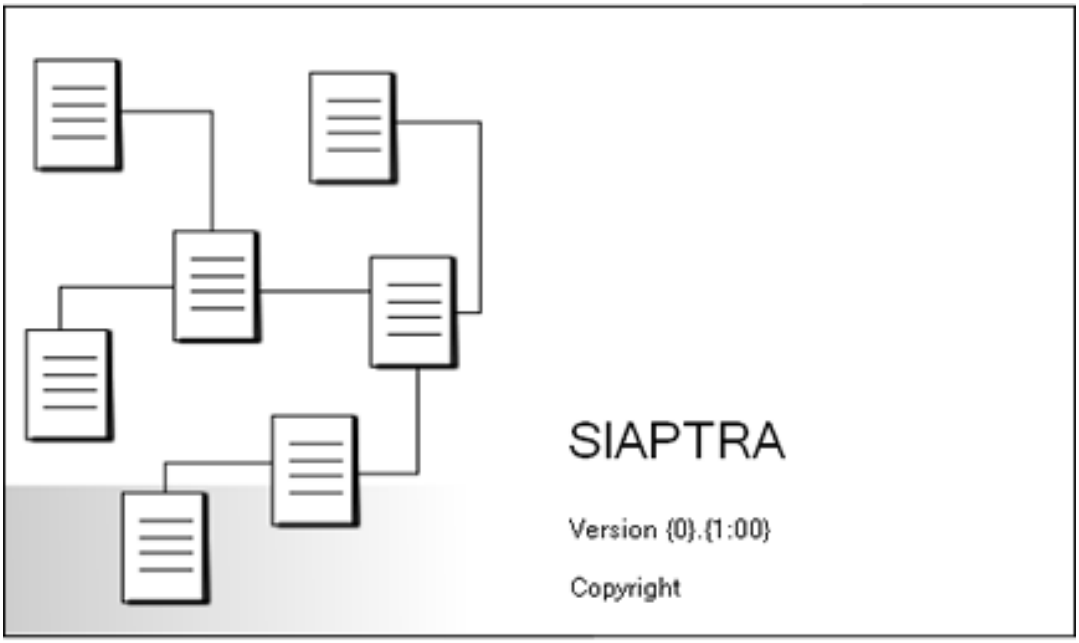
SIAPTRA es una herramienta diseñada para profesionales en el ámbito de auditorías, con el objetivo de facilitar la realización de trabajos planificación y seguimiento de estas.



Índice

SIAPTRA

Sistema de Administración de Planes Anuales y Trabajos de Auditoria



Datos Generales

Con este Sistema informático se pueden realizar trabajos como:

1. Planificación anual de auditorias
2. Planificación y seguimiento de auditorías.
3. Cálculo de costo de cada auditoria
4. Elaboración de gráficos y reportes de las auditorias, incluyendo diagramas de Gantt

Empezar a utilizar Siaptra

Obtener Ayuda:

Obtener respuestas a inquietudes y solucionar problemas con respecto a Siaptra.

Requerimientos de Sistema:

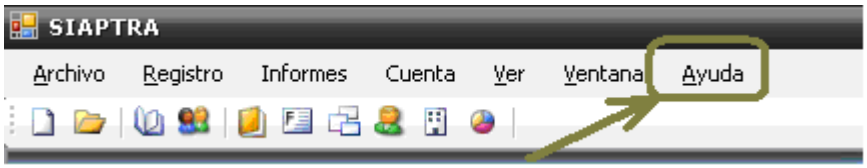
Requisitos mínimos y óptimos para el funcionamiento de Siaptra en su PC.

Interfaz grafica:

Lo que aparecerá una vez iniciada la sesión de usuario del sistema.

Obtener Ayuda

Click en Ayuda en la barra de menú o simplemente presione F1 para obtener ayuda y resolver problemas acerca de Siaptra



Requerimientos de Sistema

Configuración mínima:

1. Sistema operativo: versiones de Windows XP, Vista o Windows 7 de 32 bits
2. Procesador: al menos 1.8 GHz
3. Memoria: 512 MB de RAM
4. HD: 40 GB
5. SQL-Server 2000
6. Plataforma Net Framework 2.0
7. Microsoft Excel
8. Crystal Report

Configuración recomendada:

1. Procesador: 3.0 GHz o mas
2. Memoria: 2 Gb de RAM
3. HD: 200 GB y 50 GB disponible
4. 1024 x 768 de resolución de pantalla o mas
5. Mas la configuración mínima.

Interfaz

Al iniciar aparecerá

Barra de herramientas:

Contenido

Usuarios:

Crea, modifica y cambia el password del usuario, además de comprender los niveles de Usuario de Siaptra...

Planes anuales:

Crear planes anuales e incorporar auditorias a estos.

Unidades auditadas:

Crear, modificar o eliminar una empresa o institución propuesta a realizar diferentes auditorias.

Auditorias:

Agregar, eliminar o realizar modificaciones a las Auditorias o a los planes de Auditoria.

Personal

Crear, modificar o dar de baja auditores de la base de datos.

Informes

Muestra informes de Auditoría, Planes de Auditoria, Personal de una Auditoria, etc.

Usuarios

Para acceder a Siaptra existen cinco niveles de usuario:

1. Gerente
2. Jefe de División
3. Supervisor
4. Encargado de auditoria
5. Auditores

Cada usuario posee una identificación o login, con la cual, en conjunto con la contraseña que el proponga, podrá acceder a Siaptra con un nivel de permiso.

Crear usuario

Para crear un nuevo usuario escribe tu login y tu contraseña en la ventana que aparecerá, al hacer click en la barra de menú en usuario, luego crear usuario. Esta opción solo esta disponible para el administrador del Sistema.

Ver niveles de permisos de Usuarios

Niveles de usuario

Los permisos que posee cada usuario de Siaptra se definen como:

1. **Gerente**: es el máximo nivel de permiso, solo existe un usuario que puede poseer este nivel, es capaz de crear planes anuales, usuarios, auditorias, divisiones de trabajo y personal, así también modificar los atributos de los antes mencionados.

2. **Jefe de División**: es capaz de modificar y agregar planes de auditorías, asignar personal a una auditoria y modificar la información de esta.

3. **Supervisor**: puede administrar una auditoria asignada por el Jefe de división o el gerente.

4. **Encargado de auditoría**: puede imprimir reportes quincenales, diagrama de Gantt, y de las actividades de la auditoria que le corresponde,

5. **Auditores**: Es el que ejecuta físicamente una auditoria, tiene el mismo nivel de permiso que el Encargado de Auditoria.

Planes anuales

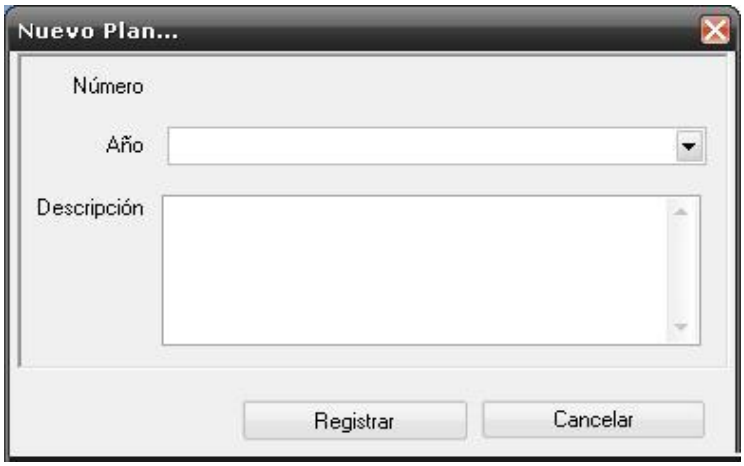
En cada plan anual se registran todas las auditorias que han de iniciar o iniciadas en determinado año.

Este plan anual contiene un número consecutivo, el año del plan, una descripción y la cantidad de auditorías que contiene cada plan.

Ver Crear plan anual

Crear plan anual

En la barra de menú, hacer click en ***Archivo***, luego ***Nuevo plan Anual*** para que aparezca la siguiente ventana:



Ingrese el año del nuevo plan y su descripción (el numero de plan es consecutivo).

Incorporar auditoria a un plan anual

Vincular una Auditoria con uno o más planes anuales, se realiza al momento de ***Crear Auditoria***.

Unidades auditadas

Unidades Auditadas hace referencia a una empresa u organismo, el cual está dividido por Áreas que pueden ser auditadas separadamente. No es posible crear una auditoria, si no se ha creado antes la Unidad (Institución) que se va a auditar.

Crear Unidad a Auditar:

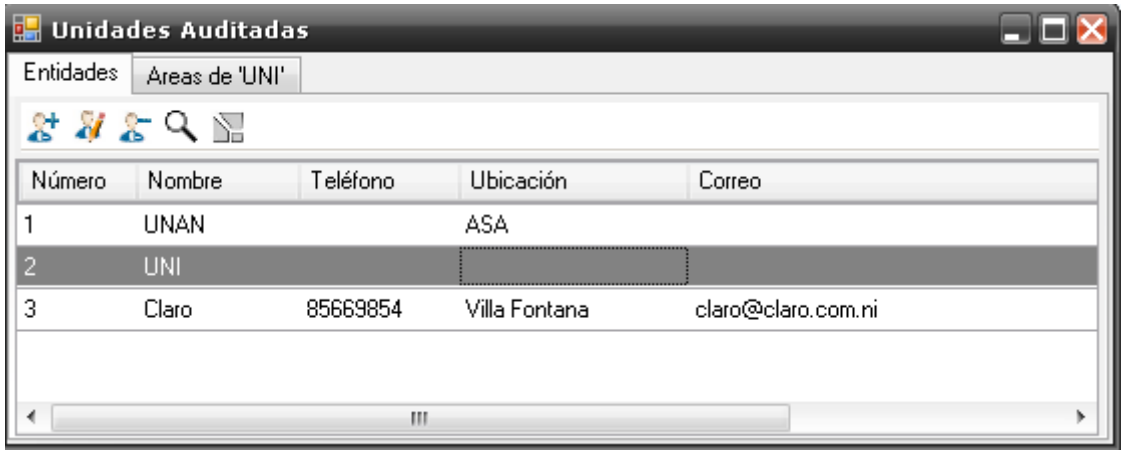
Paso a paso se registran los datos de una Institución, para luego ligarla a una o varias auditorias.

Crear Área:

Se registran las áreas de la institución en la que se realizaran diferentes auditorias.

Crear Unidad

Para crear una unidad auditada, hacer click en el botón  , o bien diríjase a ***Archivo***, luego ***Unidades Auditadas:***



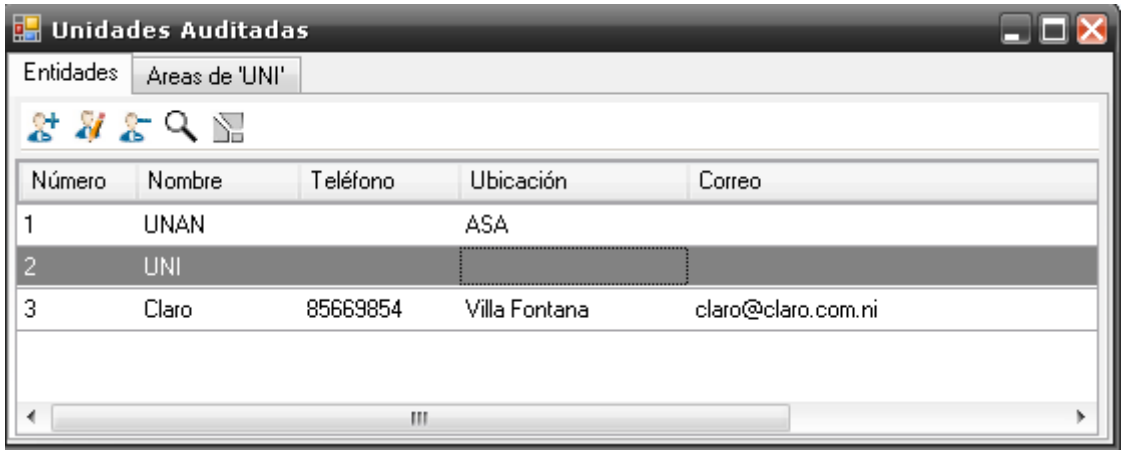
Hacer click en el botón  dentro de la ventana Unidades Auditadas, introduce los datos correspondientes y hacer click en registrar

De igual manera existen los botones para modificar, eliminar, buscar y ver áreas en la

barra de herramientas de la ventana Unidades Auditadas.

Crear Área

Para crear un área de unidad auditada, primero hacer click en el botón , o bien diríjase a ***Archivo***, luego ***Unidades Auditadas:***



Debe seleccionar la unidad que contiene el área que deseamos abrir, hacer click en el botón dentro de la ventana Unidades Auditadas y, si existen, encontraras las áreas que pertenecen a esta unidad, seleccione un área y presiona el botón , introduce los datos correspondientes y hacer click en registrar.

De igual manera existen los botones para modificar, eliminar, buscar las áreas de las Entidades a Auditar, simplemente seleccionando un área y haciendo click el botón.

Auditorias

Temas relacionados:

Crear auditoria:

Se registra una nueva auditoría.

Abrir auditoria:

Se visualizan los datos registrados de una auditoria y a la vez, se pueden modificar dependiendo de su estado.

Plan de auditoría:

Visualiza y modifica un plan de auditoría, ya sean:

Fases de la Auditoria

Actividades de las fases

Personal asignado

Datos reales

Ver diagrama de Gantt de la auditoria:

Muestra un reporte de las actividades en un diagrama de Gantt

Crear auditoria

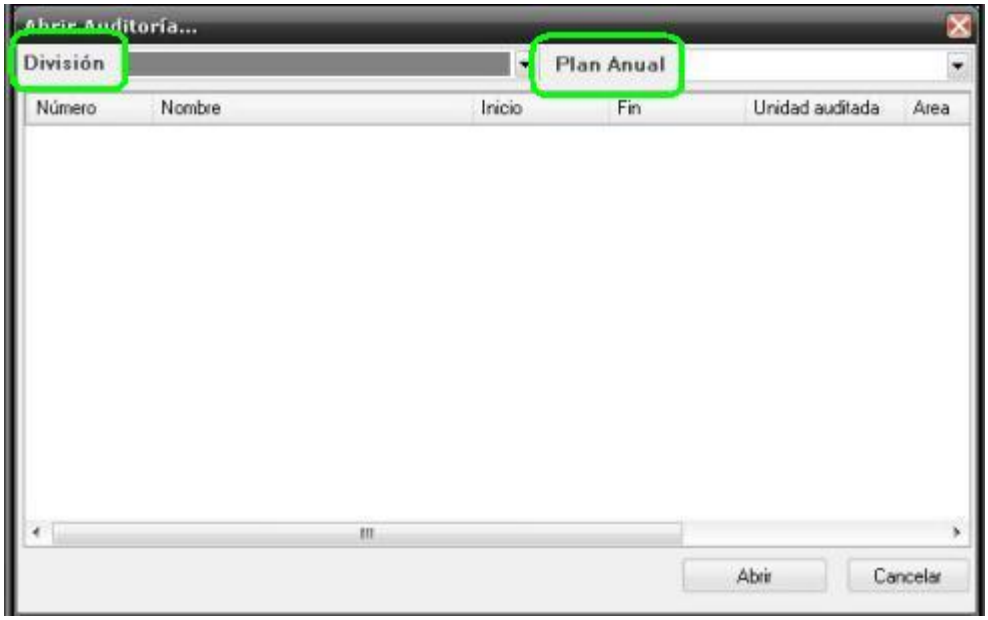
Para crear una nueva auditoría, hacer click en el botón, en la barra de herramientas o ir a ***Archivo, Nueva Auditoria***, luego aparecerá:

Introduce los datos de la nueva auditoría en cada folder y **Guarda** los cambios de esta en la barra de herramientas.

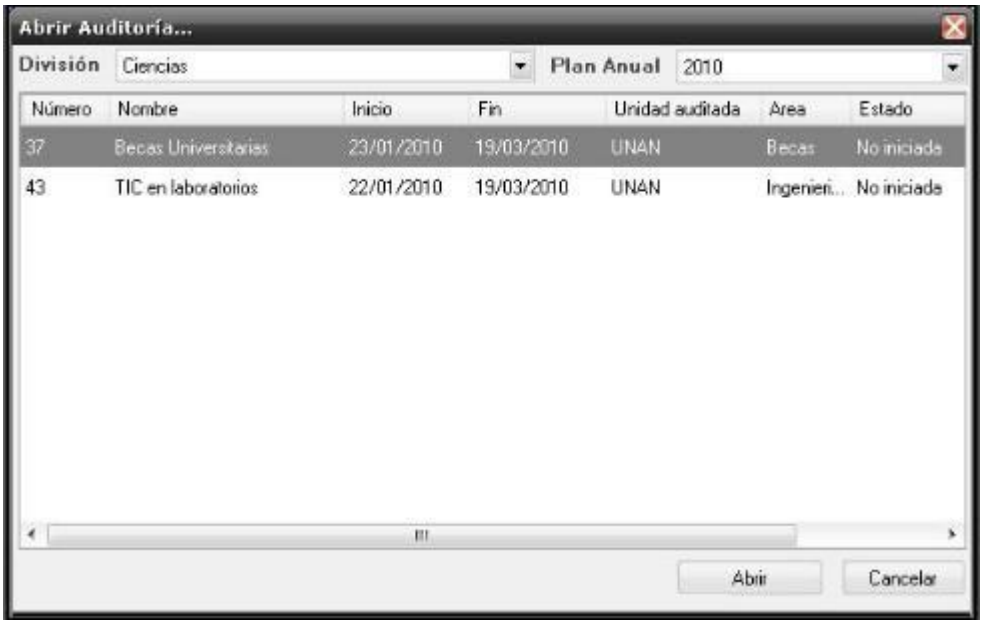
Nota: una auditoria puede pertenecer a dos planes anuales si su fecha de inicio es en determinado año y su fecha de finalización es el año siguiente.

Abrir y modificar Auditoria

En la barra de herramientas presione el botón  o ir a ***Archivo***, luego ***Abrir Auditoria***; aparecerá la siguiente ventana:



A continuación, escoja la división y plan, para visualizar las auditorias correspondientes e estos:



Seleccione una auditoria y haga click en aceptar para ver y/o modificar los datos de la auditoria seleccionada.

Nota: si la auditoria seleccionada ha cambiado estado a **iniciada**, no se pueden modificar sus datos, por la misma razón.

Plan de auditoria

Abrir

Planes de auditoría existentes

_topic_CrearFases

Crear Fases:

Luego de haber creado una auditoria, se agregan las fases que le corresponden

Crear Actividades:

Paso a paso se crean las actividades de cada fase y se agregan sus atributos.

Asignar personal:

Se asignan los auditores que encargados de una auditoria

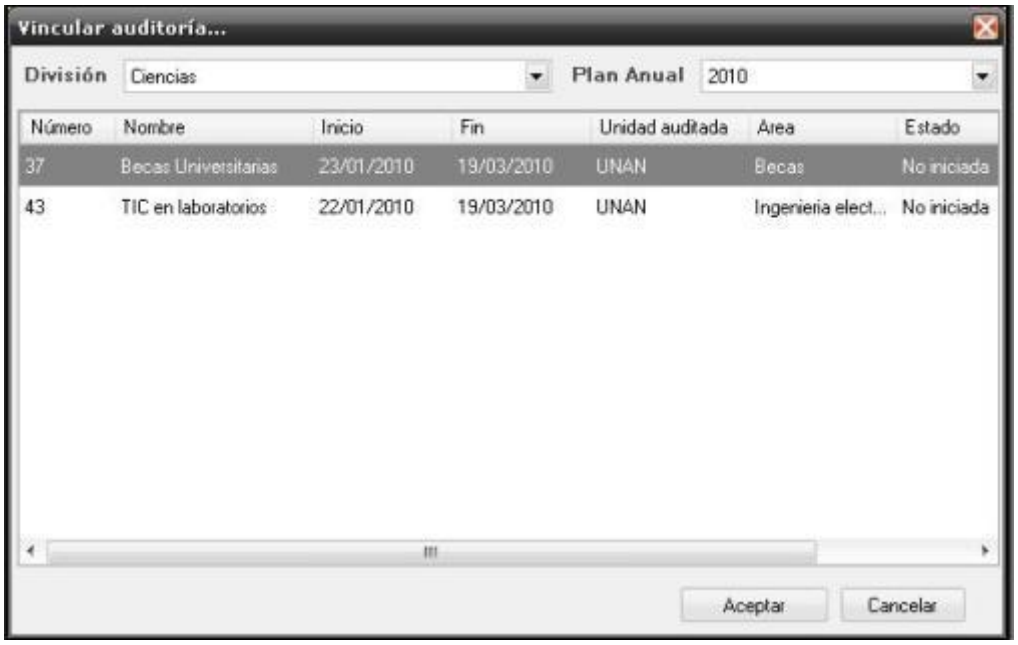
Datos reales:

Se introducen los datos reales al momento y luego de la ejecución de un Plan de Auditoria.

Abrir Plan de Auditoria

Para abrir un Plan de Auditoria existente, primero vaya a ***Archivo*** y hacer click en ***Plan de auditoría*** para que aparezca:

Luego hacer click en el botón ***Vincular***, escoja la división y plan correspondientes a la auditoria que se desea abrir, seleccione la auditoria y hacer click en ***aceptar***:



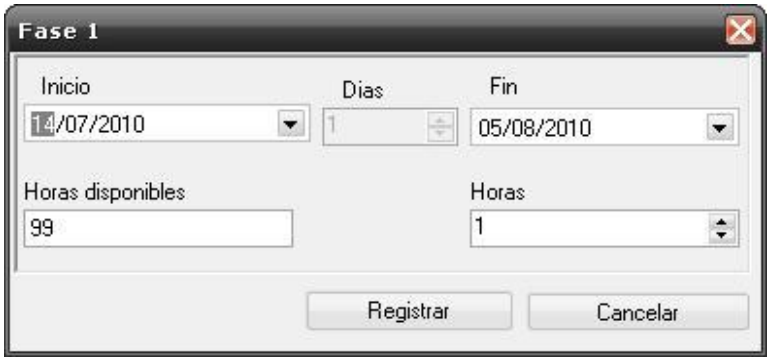
Aparecerá la ventana de plan de auditoría con los datos de la auditoria vinculada:

Crear Fases de una auditoria

Convencionalmente una auditoria consta de cuatro fases, es por ello, que esta es la cantidad máxima de Fases que pueden ser integradas a la auditoria.

Para registrar las fases de auditoría, primero debemos abrir un plan de auditoría, luego de esto, la barra de herramientas mostrará una serie de botones adicionales para Crear, modificar y eliminar fases de la auditoria:

Haga click el botón  , luego seleccione ***Fase***, complete los datos requeridos y presione ***registrar***:

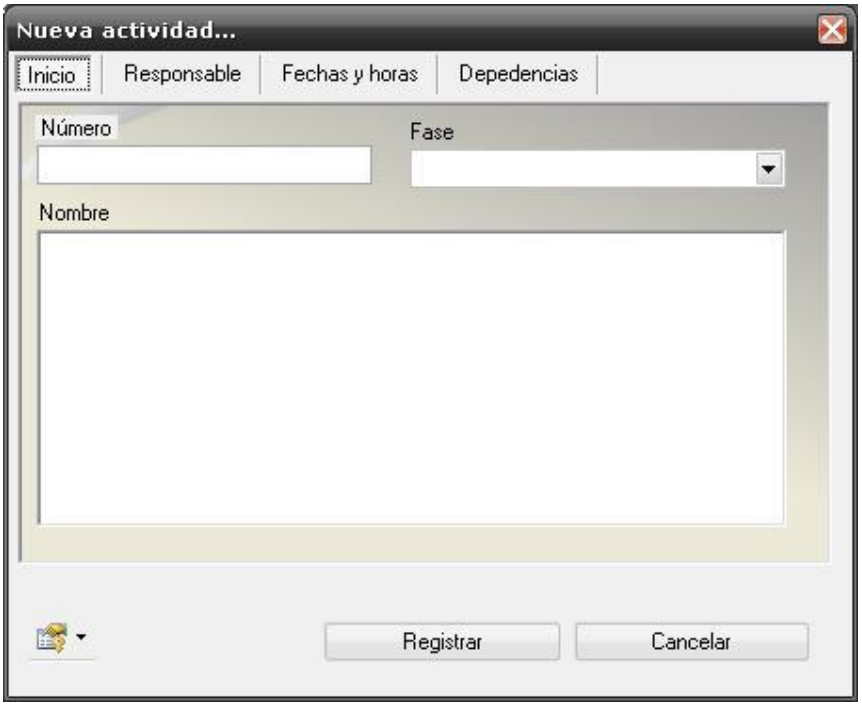


Crear Actividades

Cada fase de una auditoria esta integrada por actividades, debido a ello, solo se pueden crear actividades cuando ya han sido registradas las fases de la auditoria.

Para registrar las fases de auditoría, primero debemos abrir un plan de auditoría, luego de esto la barra de herramientas mostrará una serie de botones adicionales para Crear, modificar y eliminar actividades de cada fase:

Haga click el botón  , luego seleccione ***Actividad***, complete los datos requeridos y presione ***registrar***:

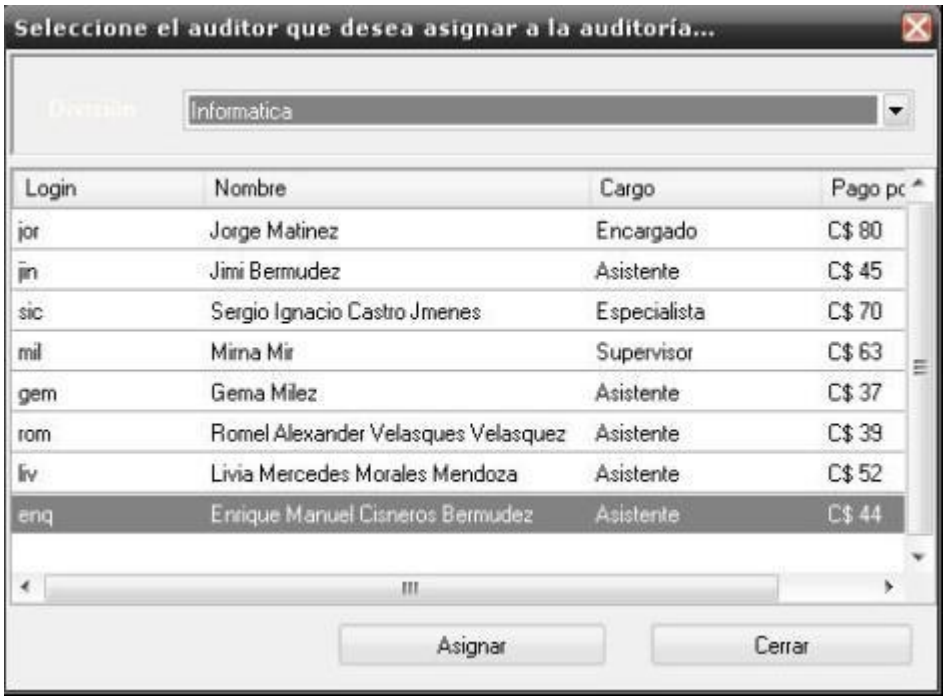


Nota: si la auditoria aun no tiene registrado personal, al presionar el botón, automáticamente aparecerá una ventana para Asignar personal a la auditoria.

Asignar personal a una Auditoria

Para asignar personal a una auditoria, debe Abrir la Auditoria correspondiente, ir a ***Archivo***, ***Personal de Auditoría*** o presionar el botón **,** para que aparezca la ventana:

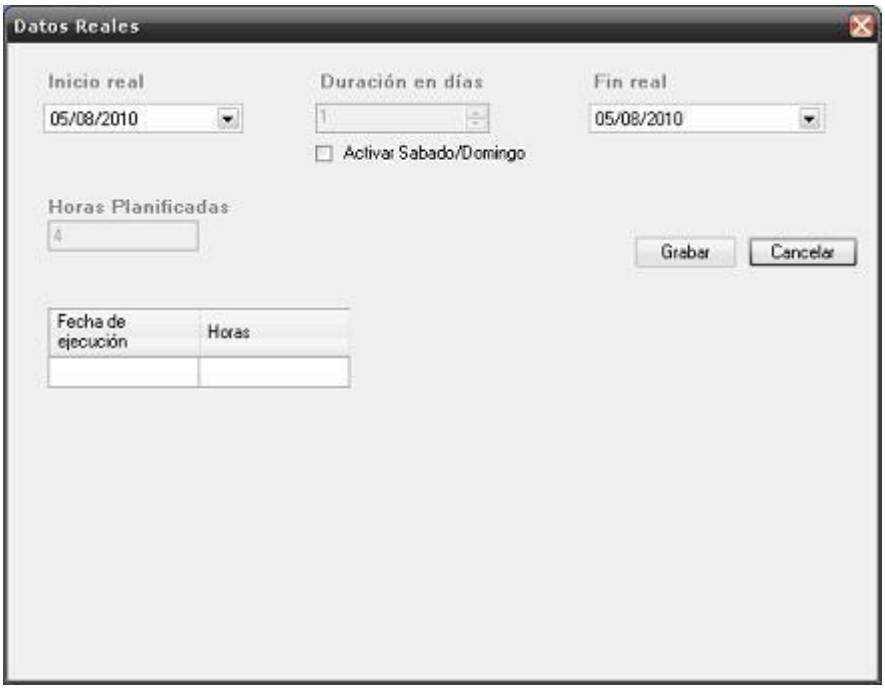
En la barra de herramientas aparecerán los botones para agregar, cambiar y eliminar personal respectivamente, presione el botón y elija quien desea agregar en la auditoria.



Datos reales

En Siaptra se almacenan fechas de planificación en las que se espera que se ejecuten las actividades de las auditorias, pero en el caso que se retrase o adelante de la fecha propuesta, la ejecución de una auditoria, el usuario puede registrar las fechas en que en realidad se inicia y finaliza cada actividad.

Para registrar datos reales, debe abrir el plan de auditoría deseado, luego en la barra de herramientas haga click en el botón  y seleccione ***Real***, aparecerá:



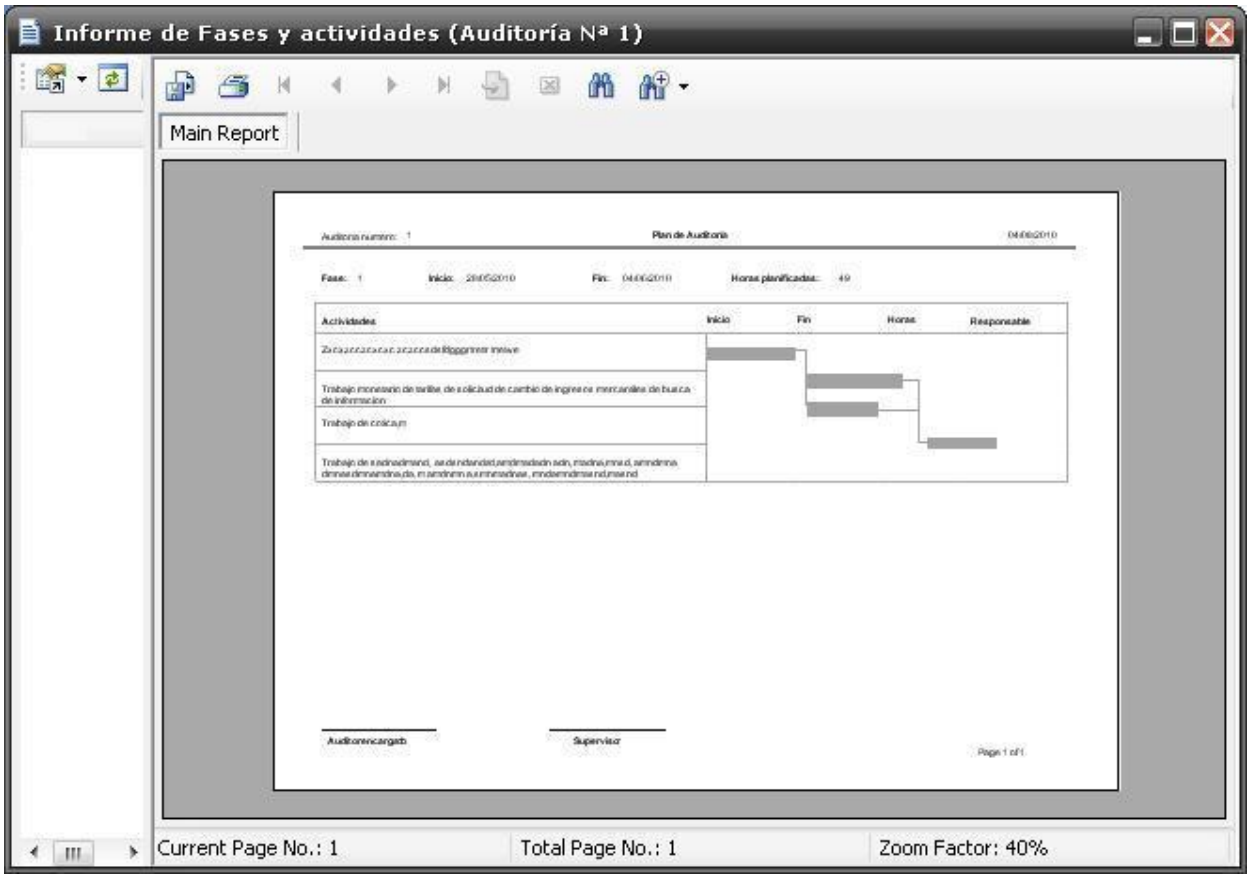
Complete los datos requeridos y presione ***Grabar***.

Nota: Para visualizar los datos reales, hacer click en el botón de la barra de herramientas y seleccionar ***Datos Reales.***

Reflejar en diagrama de Gantt

Muestra un informe de las actividades, distribuidas en un diagrama de Gantt.

Para empezar, debe abrir un plan de auditoría y en la barra de herramientas aparecerá el botón , para que aparezca el reporte.



Calculo de costos

Véase Costo de personal, en donde se refleja el costo (financiero) real de una Auditoria.

Informe

Seleccione una opción:

Ver informe de auditoría:

Refleja los datos generales de una auditoria, como:

- » nombre
- » objetivos
- » estado
- » fecha de inicio y fin
- » unidad auditada
- » horas asignadas

- » plan anual
- » auditor encargado

Ver informe de Plan de Auditoria:

Muestra las actividades y sus datos correspondientes, ordenadas por las fases a que pertenecen.

Ver informe de auditores participantes:

Muestra el personal encargado de cada auditoria, login del auditor y el cargo que ocupa.

Ver informe de costo de personal:

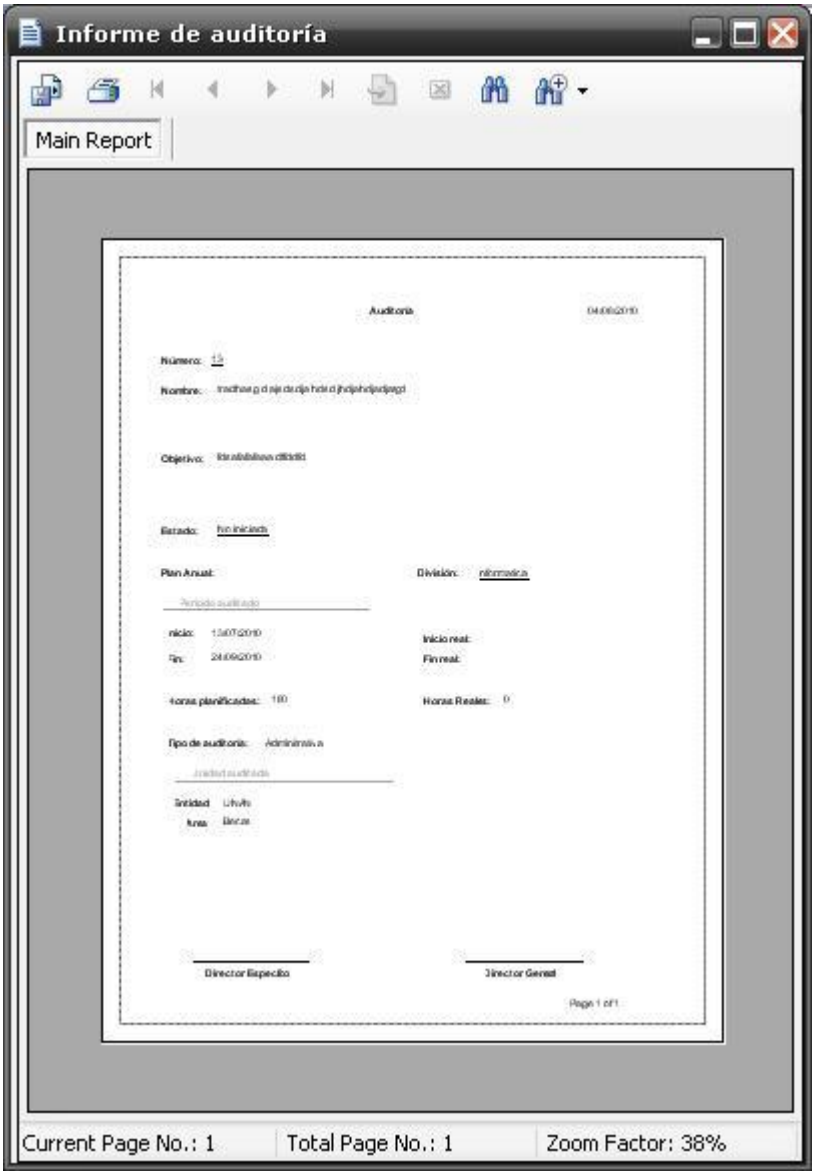
Visualiza de manera detallada en una auditoria, el pago de cada auditor en dependencia del pago por hora de este.

Ver informe Quincenal de Auditoria:

Proyecta un informe del avance quincenal de cada auditoria, que es una herramienta muy útil para el personal auditor y administrativo.

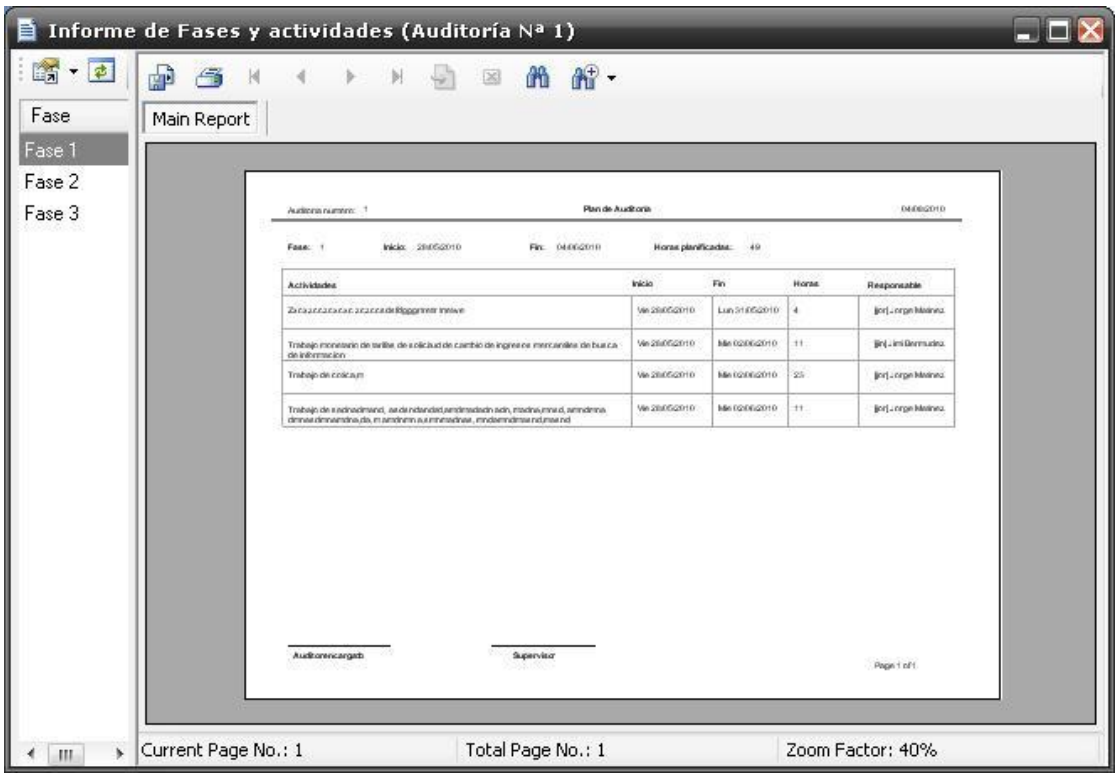
Ver informe de Auditoria

Para ver un informe de auditoría debe abrir la auditoria y presionar el botón , o vaya a ***Informe***, luego presione ***Auditoria*** y seleccione una auditoria, aparecerá un informe como:



Ver informe de Plan de Auditoria

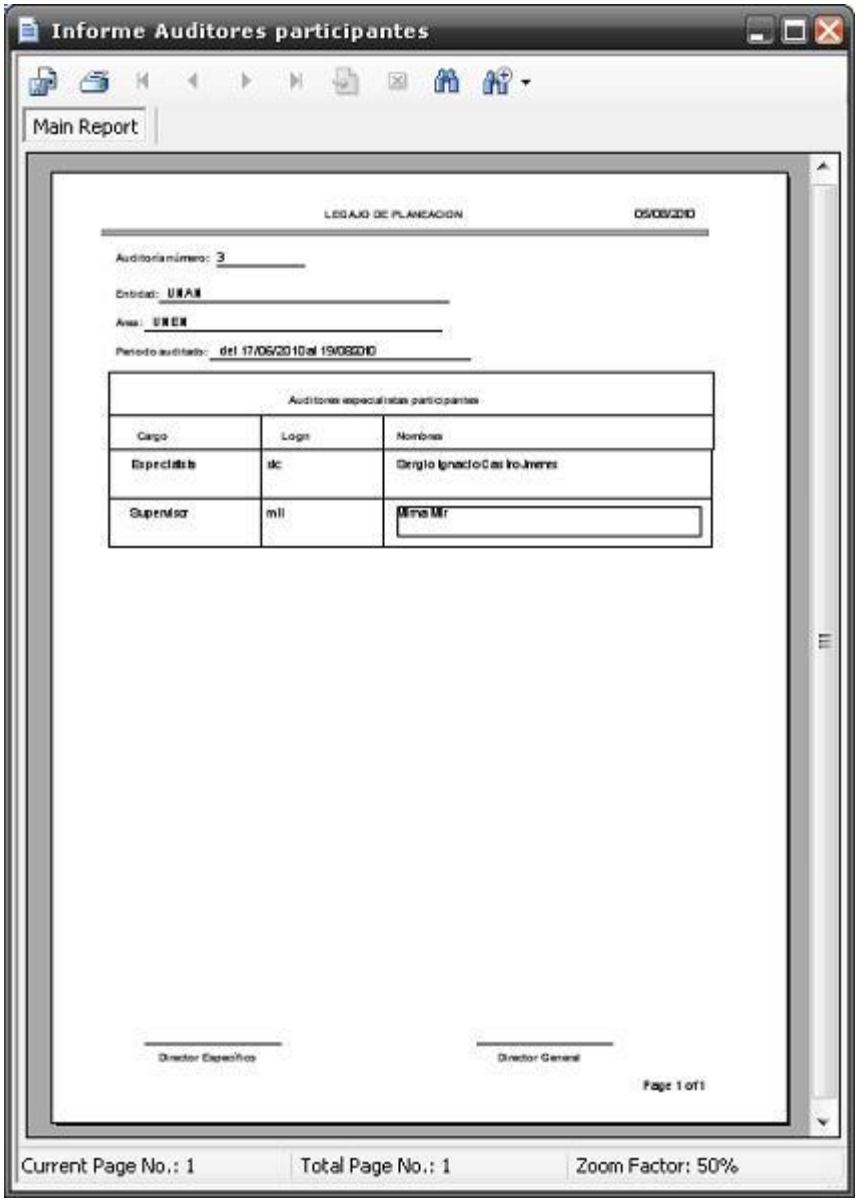
Para ver un informe de plan de auditoría debe abrir el plan de auditoría y presionar el botón  , o vaya a ***Informes***, luego presione ***Plan de Auditoria*** y seleccione una auditoria, aparecerá un informe como:



En donde las fases de la auditoria están ordenadas al lado izquierdo del reporte.

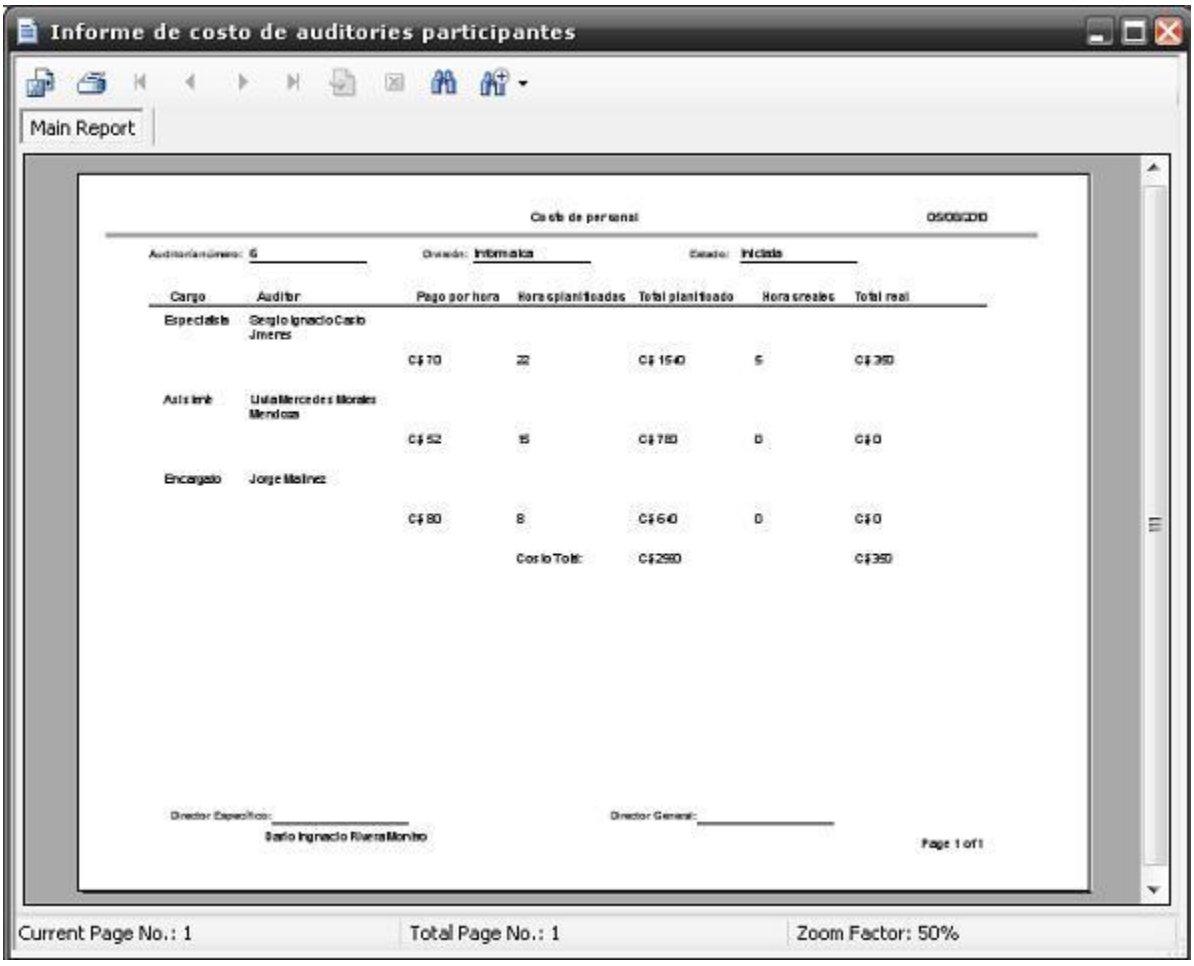
Ver informe de Auditores participantes

Para ver un informe de los auditores encargados de una auditoria, solamente necesita abrir la auditoria, hacer click en el botón  , o vaya a ***Informe***, luego presione ***Auditoria*** y seleccione una auditoria para que aparezca el informe:



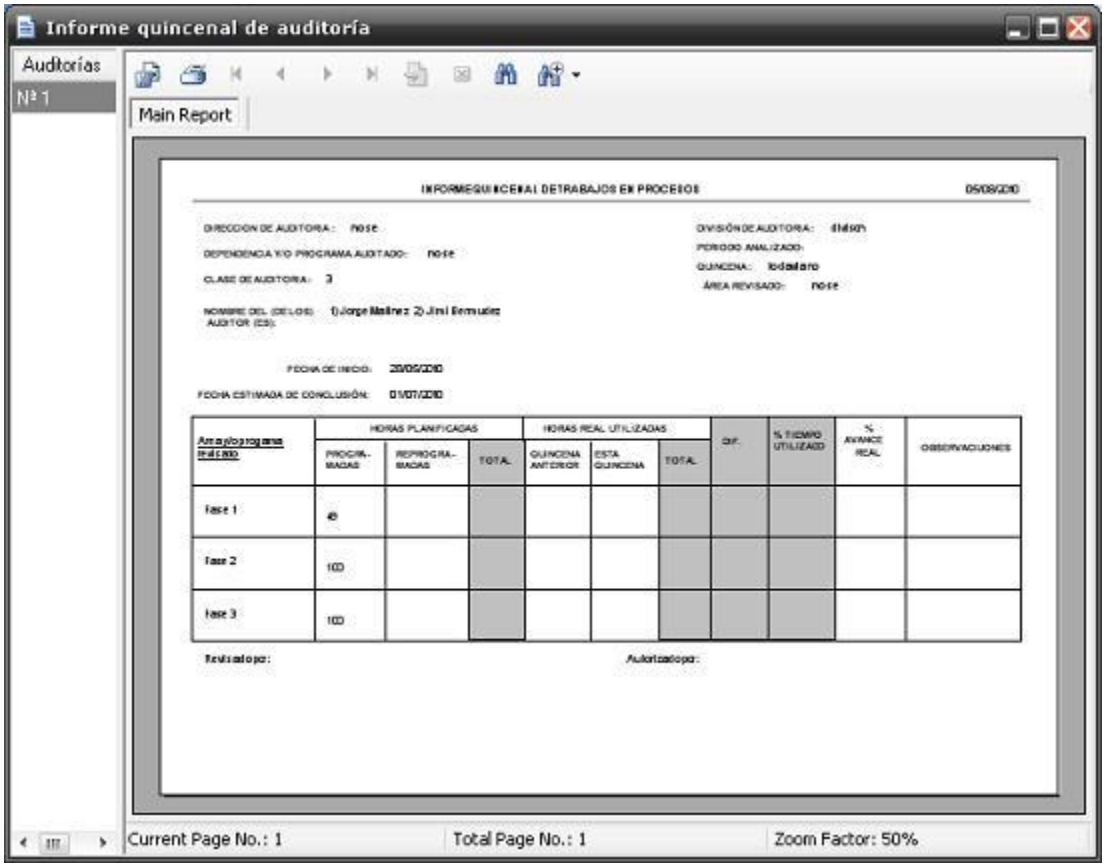
Ver informe de Costo de personal

Para ver un informe del gasto que corresponde a una auditoria, dado el pago por hora de cada miembro del personal, debe abrir la auditoria, hacer click en el botón , o vaya a ***Informe***, luego presione ***Costo de personal*** y seleccione una auditoria para que aparezca el informe:



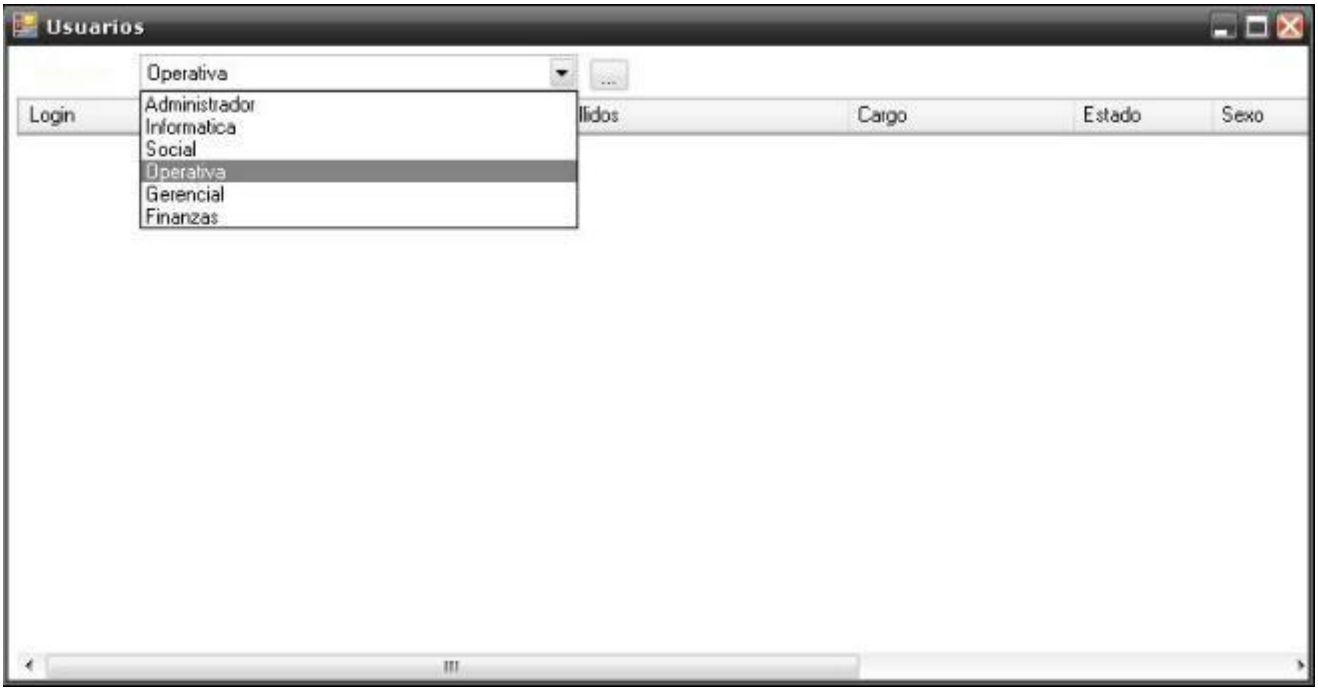
Ver informe Quincenal de Auditoria

Para ver un informe que muestra el avance real quincenal de una auditoria, debe abrir la auditoria, hacer click en el botón , o vaya a ***Informe***, luego presione ***Quincenal de Auditoria*** y seleccione una auditoria para que aparezca el informe:

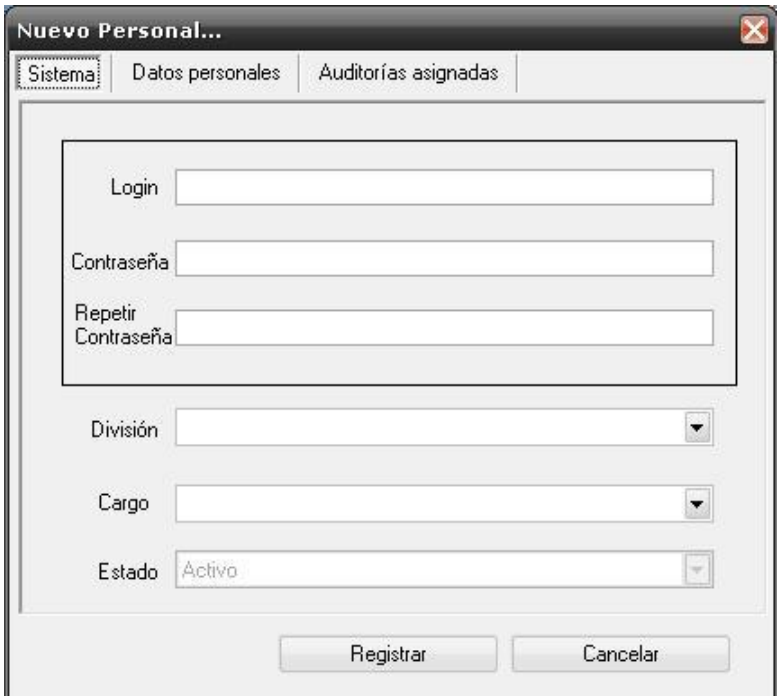


Agregar, modificar y eliminar Personal

Para empezar, cada auditor registrado en el sistema es un usuario del mismo; para Agregar un auditor, entramos en ***Archivo***, y hacemos click en ***Usuarios.***



Seleccionamos la división en que deseamos agregarlo y hacemos click en el botón  en la barra de herramientas. Se introducen los datos solicitados y presionamos ***Registrar.***



Para modificar y/o eliminar un usuario existente, abrimos la ventana usuario y seleccionamos la división para que aparezcan los botones  en la barra de herramientas y así podamos agregar, modificar o eliminar respectivamente.

Nota: Siaptra no permite eliminar un auditor (usuario) que ya ha sido asignado a una auditoria. Primero se debe desvincular de cualquier auditoria asignada.

Contáctenos

Equipo de programación:

- Alejandro López Tapia 86235689 futbolalejandro@yahoo.com
- Lorgia Flores Mendoza 84936380 lorgiaflores@gmail.com
- Frank Mena Mejía 86997999 frankamm2411@yahoo.com