BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"Indika – Sistema de administración de *indicadores de resultados* para la medición de los procesos de una organización"

Tesis que Presenta:
Julio Alberto Rojas Montoya

Para Obtener el Título de:
Licenciatura en Ingeniería en Ciencias de la Computación

Asesor:
M.C. Pedro Bello López

Puebla, Pue. Abril, 2010

TABLA DE CONTENIDO

LISTA DE FIGURAS

AGRADECIMIENTOS

A mis padres y hermana por todo el apoyo, consejos y enseñanza que me han brindado toda la vida.
El esfuerzo constante que hacen, siempre vale la pena.

A todos los profesores de la FCC que me ayudaron a seguir un sueño.

A familiares, amigos y mi cariño, por su paciencia en este pequeño camino en el que me aventuré.

El hombre es, a la vez, padre de sus obras futuras e hijo de las pasadas.
René Félix Allendy

INTRODUCCIÓN

"INDIKA es un sistema que mide la eficacia y eficiencia de los procesos de una organización, para que así los administrativos puedan ver su comportamiento y de esta forma tomen decisiones de prevención, todo esto por medio de indicadores de resultados."

En las empresas siempre se quiere llevar a cabo la medición de su gestión, actividades, productividad, recursos, etc. Es decir, quieren ver la **situación en la que se encuentra la organización** en un momento determinado, con el objetivo de tomar decisiones.
Asociado a esto, hay organizaciones que quieren certificarse en alguna norma y estas normas tienen clausulas las cuales hay que cumplir.

Las normas son un modelo, un patrón, ejemplo o criterio a seguir; es una fórmula que tiene valor de regla y tiene por finalidad definir las características que debe poseer un objeto y los productos que han de tener una compatibilidad para ser usados a nivel internacional.

Todas las normas tienen alguna clausula en la que deba de medirse algo en los elementos de la organización, como la norma NMX-CC-9001-IMNC-2000 del Sistema de Gestión de la Calidad que nos señala en la clausula 4.1 e) que se debe realizar el seguimiento, la medición y el análisis de los procesos. De igual forma la clausula 5.6.2 señala que se debe contar con la información necesaria para la revisión directiva y su toma de decisiones. O la clausula 8 que se centra en la "Medición, análisis y mejora" [1].

También el Modelo de Equidad de Género MEG:2003 señala en el punto 4.4 que debe existir una evaluación, seguimiento y mejora para sugerir acciones cuando se detecten áreas de oportunidad [2].
O también existen normas para el área de informática, como el ISO/IEC 27000 que son un conjunto de estándares que proporcionan un marco de gestión de la seguridad de la información utilizada, para cualquier tipo de organización; en los cuales se señala que debe haber una medición de eficacia para así poder implantar mejora [3].

Para todas las normas es necesario ver el desempeño, este se puede ver midiendo procesos, actividades, recursos, etc. y las

mediciones se pueden ver por medio de indicadores (ver figura 0.1).

Figura 0.1 Relación desempeño – indicadores

Así pues, un indicador se define como una **herramienta para la medición y seguimiento** de un proceso, producto y/o la misma organización.

Algunas formas en que las empresas crean y miden sus indicadores es a través de varios medios, como son las hojas de cálculo, las cuales deben ser descargadas por el usuario que se encarga de alimentarlas, llenarlas de acuerdo a reportes y posteriormente enviarlas al director o jefe inmediato.

Así mismo, unos indicadores pueden depender de otros indicadores, por lo que para alimentar un indicador, es necesario que se termine un indicador ligado a este y no tenemos información acerca del indicador anterior.

El problema surge de la necesidad de llevar un **control de los procesos de una organización**, ya que esta para poderse certificar en alguna norma, necesita cumplir una serie de cláusulas y varias de ellas proponen la medición de los procesos.

Un **proceso** se puede considerar como una actividad dentro de la organización, que utiliza recursos y que se gestiona, con el único fin de que los elementos de entrada (económicos, materiales…) se transformen en resultados, en resumen, convierten un insumo en un producto.

Algunos ejemplos de procesos son:

El proceso de fabricación de bebidas alcohólicas.

Figura 0.2 Proceso de fabricación de bebidas alcohólicas.

El proceso de otorgamientos de créditos

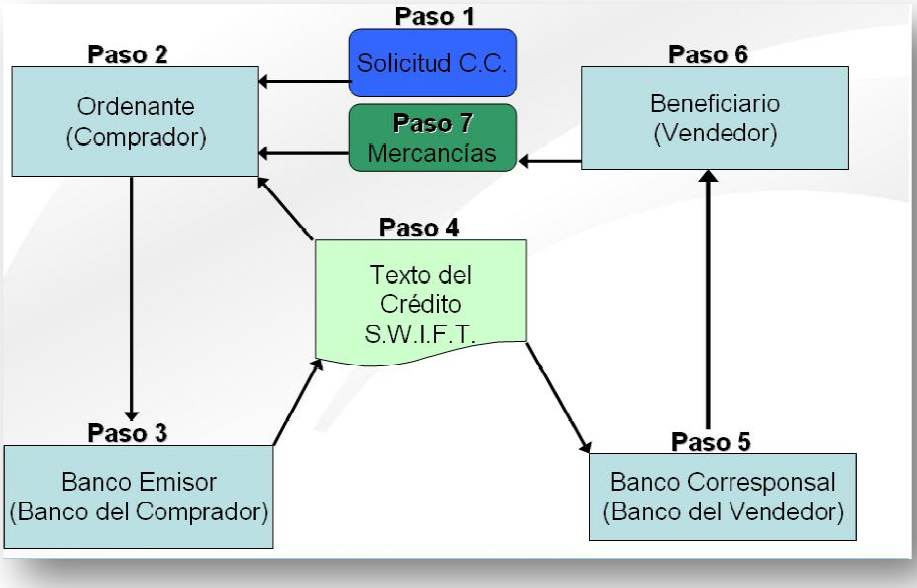


Figura 0.3 Proceso de otorgamientos de crédito.

El proceso de atención clientes

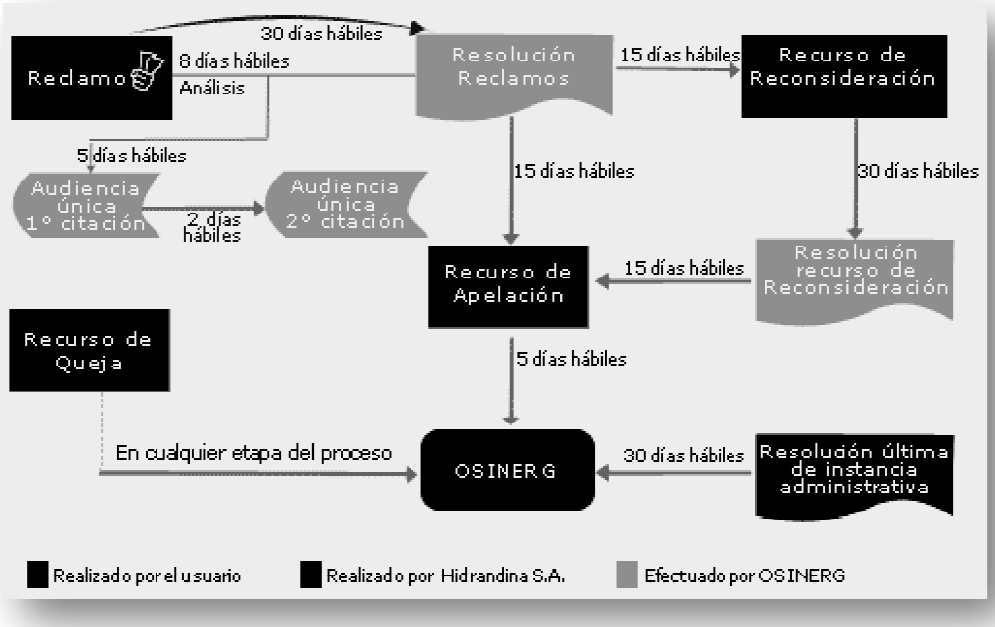


Figura 0.4 Proceso de atención a clientes

Como el sistema se desarrolló con tecnología .NET, se buscaron herramientas y complementos derivados de esta tecnología con el fin de sacar el mayor provecho, utilizando algunas características que nos ofrece hasta el momento.

Una de sus limitaciones es que para su instalación será necesario un servidor Windows, el cual tenga instalado IIS y Microsoft Net Framework y algunas actualizaciones o complementos que se utilizaron a lo largo de la elaboración del sistema.

Quizá no sea posible su visualización en dispositivos móviles, pero se desarrolló con un estilo de programación (Programación por capas) el cual permitirá la adecuación a estos de manera mas eficiente.

Por lo tanto, el sistema que se desarrolló (Indika) tiene como objetivo administrar indicadores de procesos de una organización, en aras de llevar el control y que se puedan tomar medidas preventivas por parte del personal, al percatarse de algún problema actual o por venir.

Considerando lo expuesto anteriormente el objetivo general del presente trabajo consiste en el desarrollo de un sistema que crea y administra indicadores de procesos de una organización, para que

los directores puedan ver la eficiencia de los mismos, por medio de un historial y semáforo de metas, y así pueda tomar decisiones ante un hecho actual o para la prevención, como objetivos específicos se tienen:

- Creación de una base de datos que maneje:
  - Datos de la organización.
  - Usuarios que podrán utilizar el sistema.
  - Procesos de la organización.
  - Información de los indicadores.
- Administración del sistema:
  - Administración de la información de la organización.
  - Administración de los puestos de la organización.
  - Administración de los usuarios del sistema.
- Control de acceso a módulos de acuerdo a privilegios de usuarios.
- Indicadores
  - Creación de un indicador
  - Manejar las variables para la creación de las fórmulas
  - Asignar a un usuario la responsabilidad de alimentar las variables de un indicador.
- Visualización del indicador
  - Mostrará una gráfica con el historial de las variables y la fórmula, en un periodo de tiempo seleccionado.
  - Mostrará el estado mediante un semáforo, conforme a las metas de cada indicador.
- Interface gráfica amigable para una fácil adaptación al sistema.
- Sistema autoadministrable, esto es, que no dependan del programador para realizar alguna tarea dentro del sistema.

A la mitad del desarrollo ocurrió un cambio de requerimientos por parte del cliente, por lo que ahora se manejarán sucursales las cuales tendrán sus propios indicadores.

Para este cambio se necesitará de una administración de sucursales, y la asignación de una sucursal con un indicador.

C a p í t u l o 1

MARCO TEÓRICO

En este capítulo se reúne toda la información documental posible, así como definiciones de algunos conceptos que se utilizan a lo largo del presente trabajo de tesis y algunas otras técnicas interesantes las cuales podrían interesarle al lector.

1.1 Ingeniería del Software

La ingeniería de software intenta ofrecer métodos y técnicas para desarrollar y mantener software de calidad [4].

Los objetivos de está son:

- Mejorar la calidad de los productos de software
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

Definiciones [A]:

- Ingeniería de Software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978)
- Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como Desarrollo de Software o Producción de Software (Bohem, 1976).
- Ingeniería de Software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1972).
- Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del

software; es decir, la aplicación de la ingeniería al software (IEEE, 1993).

Etapas de proceso

"La parte más dura en la construcción de un sistema software es decidir cómo construirlo…Ninguna parte del trabajo mutila el resultado del sistema si está hecho mal. Ninguna parte es más dificultosa para rectificarlo después" [B]

1.1.1 Análisis de requerimientos

Todo empieza con una decisión de emprender el proyecto, ya que muchas veces no se muestra el suficiente interés por parte de los clientes, no se convencen al 100 por ciento y esto podría causar conflictos posteriores.

Continuamos con recabar toda la información sobre el proyecto. Esta acción pretende entrevistar a los usuarios con más experiencia en cada uno de los puestos que utilizará el sistema, esto para ver las necesidades que tienen en cuanto al problema que resolverá del sistema.

Generamos un reporte de toda la información recabada e informamos de las necesidades que se tienen para el proyecto.

Ahora se tendrá que estudiar la viabilidad del proyecto y se deberían evaluar todas las alternativas para resolver los problemas encontrados como:

- Comprar un producto de software comercial como apoyo al proyecto
- Desarrollar el producto de forma interna
- Desarrollar el producto de forma externa, mediante el contrato con una empresa especializada (outsourcing).
- Automatizar parcialmente el sistema para reducir gastos.

Presentar un plan tentativo del proyecto
Aquí deberíamos plantear:

- Áreas de riesgo
- Presupuestos
- Mostrar calendario de trabajo
- Planes de trabajo del personal
- Asignación de tareas
- Soporte
- Técnicas de comunicación

- Formas de interacción con el cliente
- Modos de hacer entrevistas
- Prototipos
- Observaciones
- Como se estudiará toda la documentación
- La creación de cuestionarios para los usuarios que utilizaran el sistema.

Para la adquisición de todos los puntos anteriores se podrían plantear actividades como tormenta de ideas, dinámicas en grupo, participaciones profundas con usuarios, etc.
En esta parte veremos una técnica interesante que es el JAD (Joint Application Design)

JAD [C]

Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Incluye enfoques para mejorar la participación de los usuarios, agilizar el desarrollo y mejorar la calidad de las especificaciones.

Está basada en cuatro principios fundamentales:

- *Dinámica de grupo*
- *Uso de ayudas visuales para mejorar la comunicación*
- *Mantener un proceso organizado y racional*
- *Una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que ve es lo que obtiene), es decir, durante la entrevista se trabajará sobre lo que se generará.*

Maneja varias sesiones, y en cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación. Durante la sesión se discute en grupo sobre estos temas llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado. Pero requiere un grupo de participantes bien integrados y organizados.

Participantes de un JAD

- LÍDER DE LA SESIÓN. Organiza, coordina, establece agendas, resuelva conflictos y desacuerdos y solicita ideas (no opina ni aporta ideas).
- USUARIOS. Usuarios claves, conocen el sistema sobre la base diaria.
- GERENTES. Quienes administran y/o usan el sistema, proveen luces sobre las nuevas direcciones organizacionales, motivaciones e impactos.
- PROMOTOR. Debe realizarlo alguien de alto nivel en la organización.
- ANALISTA(S) DE SISTEMAS. Asiste(n) pero participa(n) limitadamente. Aprende(n) de usuarios y gerentes.
- ESCRIBIENTE. Toma las notas usando un procesador de palabras.
- PERSONAL DE SI. Pueden asistir y contribuir con ideas sobre la factibilidad técnica o limitaciones del sistema actual.

¿Qué son los requisitos y el análisis de requisitos?

Los requisitos son condiciones que debe cumplir un sistema para satisfacer un contrato, una norma o una especificación, de igual forma los necesita el usuario para poder resolver un problema o conseguir un beneficio determinado [5].

El análisis de requisitos es el proceso de estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, de hardware o de software. También lo podríamos definir como el proceso de estudio y refinamiento de dichos requisitos [6].

Importancia del análisis de requisitos

Esta parte es muy importante ya que el 37% de los problemas en los sistemas se debe a un mal planteamiento de los requisitos, aunado a otros factores que aumenta el coste del proyecto como se puede observar en la figura 1.1.

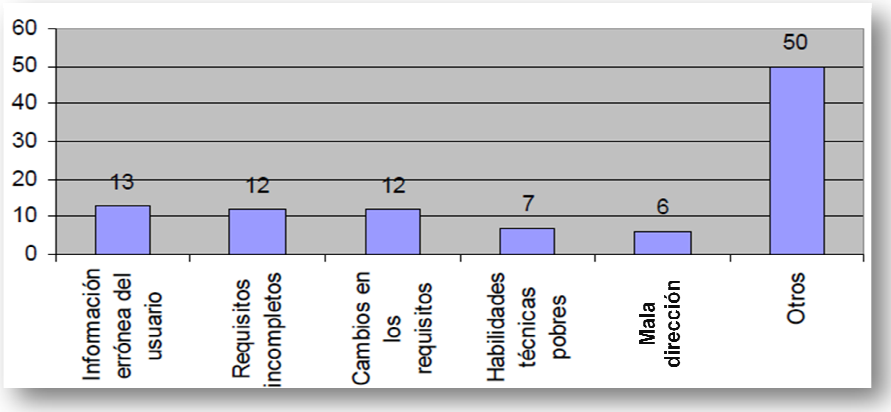


Figura 1.1 Factores del coste en proyectos software reales

La causa del fracaso más frecuentemente nombrada por programadores es “requisitos de usuario cambiantes”.

Tenemos dos tipos de requisitos, funcionales, que describen la funcionalidad o los servicios que se espera que el sistema proveerá, sus entradas y salidas, excepciones, etc. y no funcionales, que se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida, y la representación de datos que se utiliza en las interfaces del sistema.

En general las actividades de análisis de requisitos son:

- Extracción o determinación de los requisitos: es la etapa en la que los clientes descubren, revelan y comprenden los requisitos que desean.
- Análisis de requisitos: Es el proceso en el que se razonan los requisitos obtenidos en la etapa anterior, resolviendo posibles inconsistencias.
- Especificación de requisitos: Es el proceso de registro de los requisitos, se podría decir que ya son los requisitos definitivos. Se pueden mostrar por medio de lenguaje natural, gráficos, etc.
- Validación de los requisitos: Los usuarios finales confirman que los requisitos especificados son validos, consistentes (que los requerimientos no tienen definiciones contradictorias) y completos.

1.1.2 Especificación

El objetivo de este servicio es la formalización (definición, análisis y verificación) de los requisitos del sistema, a partir de los requisitos de los usuarios, objetivos de negocio, restricciones de diseño, estándares externos, etc. identificado en la fase de viabilidad de la misma, generando los siguientes documentos:

- Documento de Requisitos funcionales y técnicos de la aplicación.
- Plan de revisión y seguimiento del documento durante el desarrollo del proyecto.
- Matriz de trazabilidad de requisitos (casos de uso). Lista que referencia cada requisito con los puntos o párrafos de los documentos de especificación de los usuarios.
- Plan de pruebas.

1.1.3 Arquitectura

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

En esta parte se hace referencia de un Arquitecto de software, el cual va a añadir valor a los procesos de negocios con su aporte para soluciones en:

- La integración de infraestructura
- El desarrollo de aplicaciones, bases de datos y herramientas gerenciales

El Arquitecto de software requiere de capacidad y liderazgo para poder conceptualizar y proyectar a futuro los problemas de hoy [D].
En general, la arquitectura es una actividad de planeación tomando en cuenta la conectividad, hardware y software. Diseña los componentes de una aplicación (entidades del negocio) y sus

interacciones, todo esto podría ser diagramado por medio de diagramas de secuencia.

Para su documentación podemos utilizar diagramas como:

- Diagramas de clases
- Diagramas de base de datos
- Diagramas de despliegue
- Diagramas de secuencia
- Diagramas de infraestructura física

1.1.4 Programación

La programación es un proceso por el cual se escribe, se prueba, se depura y se mantiene el código fuente de un programa informático.

La programación debe perseguir la obtención de programas de calidad, para ello se establecen factores como:

- Corrección
- Claridad
- Eficiencia
- Portabilidad

Dentro de la ingeniería de software, reducir el diseño del código fuente y hacerlo eficaz y eficiente es una de las metas que persigue. La duración y complejidad de esta etapa depende de los lenguajes de programación que utiliza y de los puntos previamente planteados.

1.1.5 Prueba

Son los procesos que permiten verificar y revelar la calidad del producto. Estas se integran dentro de diferentes fases del ciclo del software. De esta forma se pretende ejecutar el programa y mediante técnicas experimentales se tratan de descubrir los errores que pueda contener.

En la cadena del desarrollo de un software específico, el proceso de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. Hoy en día es crucial verificar y evaluar la calidad de lo construido de modo de minimizar el costo de su reparación. Mientras antes se detecte una falla, más barata es su corrección.

Tipos de pruebas:

- Pruebas unitarias
- Pruebas funcionales
- Pruebas de Integración
- Pruebas de validación
- Pruebas de sistema
- Caja blanca (sistemas)
- Caja negra (sistemas)
- Pruebas de aceptación
- Pruebas de regresión
- Pruebas de carga
- Pruebas de prestaciones
- Pruebas de recorrido
- Pruebas de mutación

1.1.6 Documentación

Es todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones (UML), diagramas, pruebas, manuales de usuario, manuales técnicos, etc; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

1.1.7 Mantenimiento

Trata sobre mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores, o bugs. La mayor parte consiste en extender el sistema para hacer nuevas cosas.

1.2 Modelos de ciclo de vida

Un modelo de ciclo de vida de software es una vista de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas.

Existen diversos modelos de ciclos de vida que podemos utilizar para el desarrollo de un sistema. El tipo de modelo que se elija influye en el éxito que logre el proyecto como cualquier otra decisión de planificación que se tome.

Características:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo.

El modelo de ciclo de vida apropiado puede orientar al proyecto de forma adecuada y ayudar a asegurar que cada paso se acerque más al objetivo planteado.

Cada modelo de ciclo de vida tiene diversas ventajas, como aumentar la velocidad de desarrollo, mejorar la calidad, el control y seguimiento del proyecto, minimizar gastos y riesgos o mejorar la relación con los clientes.

Si se selecciona un modelo ineficaz, corremos el riesgo de que nuestro trabajo sea lo contrario a las ventajas señaladas anteriormente, por lo que se debe de analizar el proyecto y así poder definir el modelo de ciclo de vida más adecuado, y así evitar un sistema lento, repetitivo, innecesario o frustrante.

1.2.1 Modelo de cascada

Este es el más básico de todos los modelos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. Fue propuesto por Winston Royce en el año de 1970. Es un ciclo de vida que admite iteraciones, contrariamente a la creencia de que es un ciclo de vida secuencial como el lineal.

Después de cada etapa se realiza una o varias revisiones para comprobar si se puede pasar a la siguiente. Es una modelo rígido, poco flexible y con muchas restricciones.

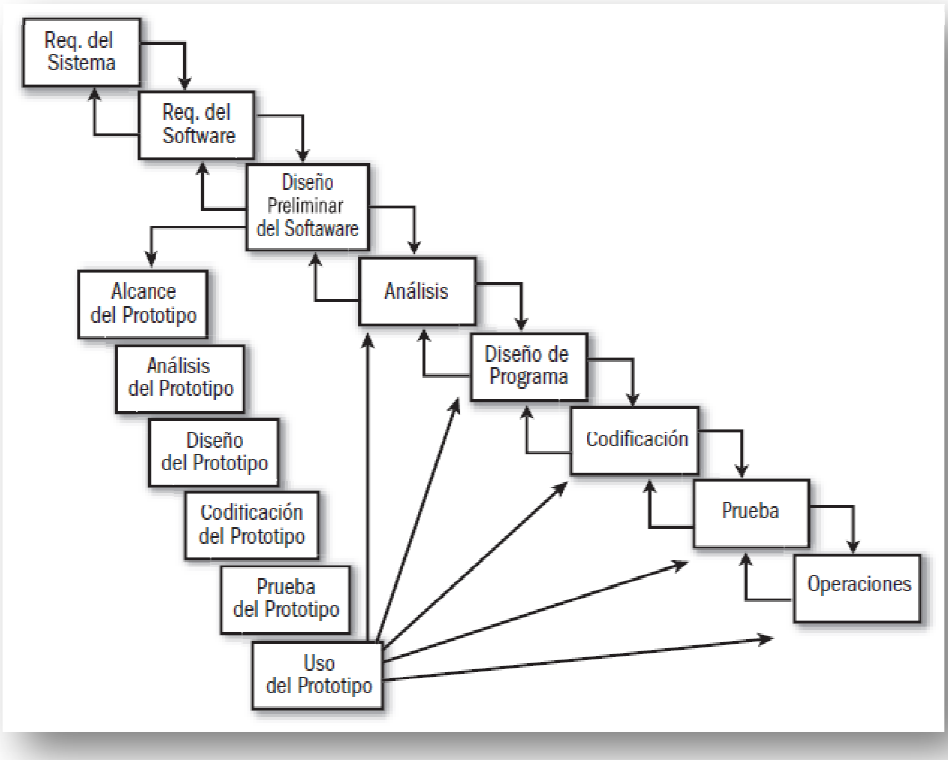


Figura 1.2 Ciclo de vida en cascada puro

La visión del modelo cascada del desarrollo de software es muy simple; dice que el desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuyen a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la fase. Se diagrama por medio de flechas que muestran el flujo de información entre las fases. La flecha de avance muestra el flujo normal. Las flechas hacia atrás representan la retroalimentación. Un ejemplo de modelo en cascada se puede ver en la figura 1.2.

Una de las contribuciones más importantes del modelo cascada es para los administradores, posibilitándoles avanzar en el desarrollo, aunque en una escala muy bruta.

Pero tiene desventajas:

1.- Dificultad para especificar claramente los requerimientos al comienzo del proyecto (no permite flexibilidad en los cambios).

2.- Para un proyecto de desarrollo rápido, el modelo de cascada puede suponer una cantidad excesiva de documentación.

3.- Si se intenta mantener la flexibilidad, la actualización de la especificación se puede convertir en un trabajo a tiempo completo.

4.- No es imposible volver atrás utilizando el modelo de cascada pura, pero si difícil.

5.- Genera pocos signos visibles de progreso hasta el final.

Así, cualquier error de diseño detectado en la etapa de prueba, conduce necesariamente al rediseño y nueva programación del código afectado, aumentado los costos de desarrollo.

1.2.2 Modelo de desarrollo evolutivo

Se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

En particular, este modelo está basado en 3 macro procesos, que son:

- La toma de requerimientos
- Actividades de desarrollo
- Entrega de avances y evaluación

Cada uno de estos procesos tiene interacción con otros en la creación del software y la forma correcta de llevarla es la siguiente:

Se toma requerimientos por parte de los usuarios finales y se crea un esbozo de la descripción, de ahí pasa a las actividades concurrentes que interactúan entre si como es la especificación, el desarrollo y la validación. Cada una de estas etapas se comunica con otra para hacer retroalimentación y cada etapa de una pequeña versión del software.

Para la especificación, crea una versión inicial, con la que interactúa; en el desarrollo crea versiones inmediatas y la validación genera la versión final (ver figura 1.3).

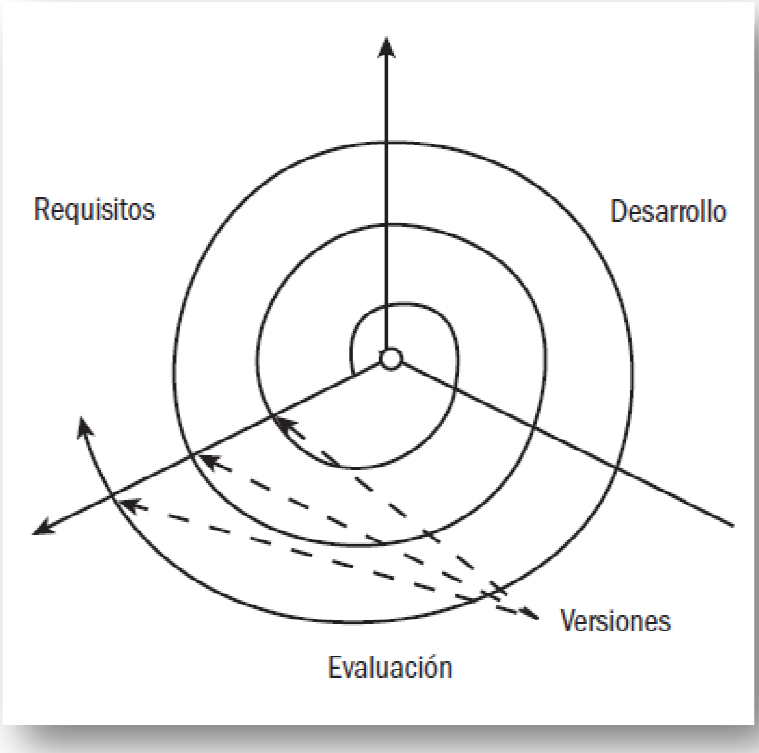


Figura 1.3 Modelo de desarrollo evolutivo

Existen dos tipos de desarrollo evolutivo:

- **Desarrollo exploratorio**: donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona conforme se añaden nuevas características propuestas por el usuario.
- **Enfoque utilizando prototipos**: En el que el objetivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

1.2.3 Modelo espiral

El desarrollo en espiral es un modelo de ciclo de vida del software definido por primera vez por Barry Boehm en 1988, utilizado generalmente en la Ingeniería de software. Las actividades de este modelo se conforman en una espiral, en la que cada bucle o

iteración representa un conjunto de actividades las cuales generan madurez en el producto final. A medida que el ciclo se cumple (el avance del espiral), se van obteniendo prototipos sucesivos que van ganando la satisfacción del cliente o usuario.

Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Se suele interpretar como que dentro de cada ciclo de la espiral se sigue un Modelo Cascada, pero no necesariamente debe ser así.

En cada vuelta o iteración hay que tener en cuenta:

Los Objetivos: Que necesidad debe cubrir el producto.

Alternativas: Las diferentes formas de conseguir los objetivos de forma exitosa, desde diferentes puntos de vista como pueden ser:

1. Características: experiencia del personal, requisitos a cumplir, etc.
2. Formas de gestión del sistema.
3. Riesgo asumido con cada alternativa.

Desarrollar y Verificar: Programar y probar el software.

Si el resultado no es el adecuado o se necesita implementar mejoras o funcionalidades

Se planificarán los siguientes pasos y se comienza un nuevo ciclo de la espiral. La espiral tiene una forma de caracola y se dice que mantiene dos dimensiones, la radial y la angular:

1. Angular: Indica el avance del proyecto software dentro de un ciclo.
2. Radial: Indica el aumento del coste del proyecto, ya que con cada nueva iteración se pasa más tiempo desarrollando.

Este sistema es muy utilizado en proyectos grandes y complejos como puede ser, por ejemplo, la creación de un Sistema Operativo.

Al ser un modelo de Ciclo de Vida orientado a la gestión de riesgo se dice que uno de los aspectos fundamentales de su éxito radica en que el equipo que lo aplique tenga la necesaria experiencia y habilidad para detectar y catalogar correctamente los riesgos.

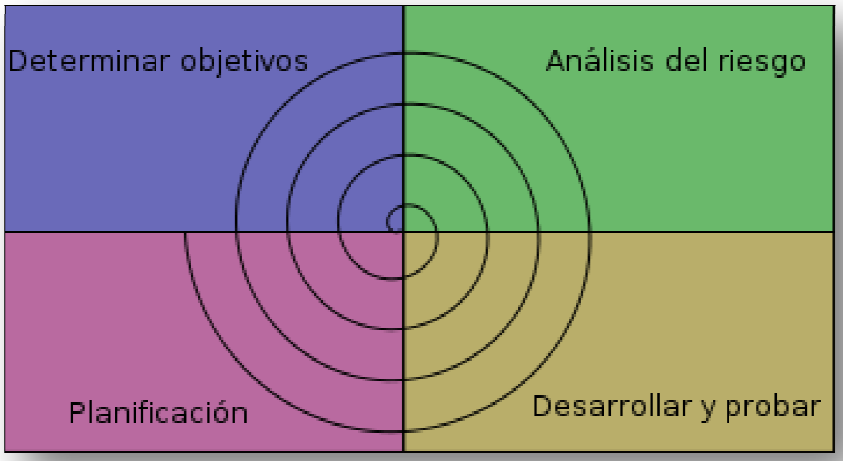


Figura 1.4 Modelo en espiral

1.2.4 Modelo Orientado a Objetos

La técnica fue presentada en la década de los 90. Y es muy parecido al paradigma de la programación orientada a objetos, es decir, cada funcionalidad, o requerimiento solicitado por el usuario es considerado un objeto.

Un objeto está considerado como un conjunto de propiedades llamadas atributos y su comportamiento, llamados métodos.

1.3 Modelo a utilizar - Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos.

1.3.1 Características [E]

Iterativo

Divide el proyecto en iteraciones. En cada una se desarrollan prácticamente todas las actividades del proyecto y se produce un sistema utilizable. Con cada iteración el sistema evoluciona, incluyendo nueva funcionalidad o mejorando la existente.

Es necesario que el resultado de cada iteración sea utilizable porque el Proceso Unificado requiere que tras cada iteración los usuarios, el cliente y el equipo valoren el sistema. Esta valoración se utiliza para redefinir los objetivos del proyecto. De esta forma, con iteraciones frecuentes, se garantiza la adaptación del sistema a los cambios durante el desarrollo.

Dirigido por los casos de uso

Los Casos de Uso constituyen la guía para el desarrollo del sistema en el Proceso Unificado. Cada iteración se planifica para añadir un conjunto limitado de Casos de Uso al sistema haciéndolo crecer iteración tras iteración.

Como es un proceso iterativo, tras cada iteración se revisan los Casos de Uso y se eligen los que se desarrollarán en la siguiente iteración.

Centrado en la arquitectura

La arquitectura del sistema se implementa en las primeras iteraciones y, sobre ella, se va añadiendo progresivamente la funcionalidad de los casos de uso.

Flexible

El Proceso Unificado establece las fases del proyecto, las formas generales de trabajo y recomienda técnicas para cada actividad. Pero deja abiertas muchas opciones y posibilidades para adaptar la metodología a las necesidades y características de cada organización, incluso de cada proyecto.

Planificación

El Proceso Unificado divide el proyecto en cuatro fases, cada una de ellas formada por una o más iteraciones.

1. **Lanzamiento**. De aquí se obtiene la decisión sobre la viabilidad técnica y económica del proyecto. Se especifican los principales casos de uso, se identifican riesgos y posibles arquitecturas.
2. **Elaboración**. Se desarrolla la arquitectura básica del sistema y se capturan la mayoría de los requerimientos del sistema. Se desarrollan los principales casos de uso para cubrir la arquitectura básica del sistema. Se obtienen los

requerimientos del sistema y de esta fase se obtienen diagramas de casos de uso y diagramas conceptuales de clases. Se define cual será la arquitectura del sistema y sus requerimientos, el plan y presupuesto.

3. **Construcción**. Es el desarrollo del resto del sistema, para esto se deben tener desarrollados completamente los casos de uso. Aquí podrían dar como resultado aparte técnicas de diseño, diagramas de secuencia y de interacción.
4. **Transición**. Es la fase final y se debe entregar el sistema a los usuarios finales. Esta fase incluye la migración y carga de datos, formación de usuarios, soporte técnico, etc.

1.4 Bases de datos

Las bases de datos son el método preferido para el almacenamiento estructurado de datos. Desde las grandes aplicaciones multiusuario, hasta los teléfonos móviles y las agendas electrónicas utilizan tecnología de bases de datos para asegurar la integridad de los datos y facilitar la labor tanto de usuarios como de los programadores que las desarrollaron.

Desde la realización del primer modelo de datos, pasando por la administración del sistema gestor, hasta llegar al desarrollo de la aplicación, los conceptos y la tecnología asociados son muchos y muy heterogéneos [F].

Una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen unos programas denominados sistemas gestores de bases de datos, abreviado SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Los SGBD que actualmente están en el mercado pretenden satisfacer un conjunto de objetivos directamente deducibles como:

- Consultas no predefinidas y complejas
- Flexibilidad e independencia
- Eliminación de la redundancia
- Integridad de los datos
- Concurrencia de usuarios
- Seguridad

1.4.1 Tipos de base de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, o la utilidad de la misma:

1.4.1.1 Según la variabilidad de los datos almacenados

- **Bases de datos estáticas**. Son de sólo lectura, primordialmente se usan para datos históricos.
- **Bases de datos dinámicas**. Estas bases de datos se modifican con el tiempo y permiten las operaciones básicas como actualización, borradas e ingreso de datos, además de consultas.

1.4.1.2 Según el contenido

- Bases de datos bibliográficas.
- Bases de datos de texto completo
- Directorios
- Bases de datos o "bibliotecas" de información Biológica

1.4.2 Modelos de Bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la

información), así como de los métodos para almacenar y recuperar información 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.

Algunos modelos con frecuencia utilizados en las bases de datos son:

- Bases de datos jerárquicas
- Base de datos de red
- Bases de datos Transaccionales
- Base de datos relacional
- Bases de datos multidimensionales
- Bases de datos orientadas a objetos
- Bases de datos documentales
- Base de datos deductivas

1.4.3 Normalización de base de datos

El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- 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:

- Cada columna debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

Formas normales:

- Primera Forma Normal (1FN), una tabla está en Primera Forma Normal sólo 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.
- Segunda Forma Normal (2FN) Dependencia Funcional
    - Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales.
- Tercera Forma Normal (3FN)
    - La tabla se encuentra en 3FN si es 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente, de la clave primaria.

1.5 Modelo cliente-servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras (ver figura 1.5).

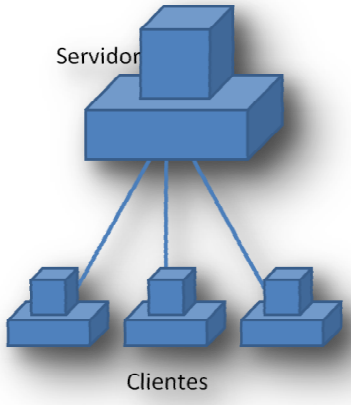


Figura 1.5 Modelo cliente-servidor

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la

gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

Características

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por cliente se conoce como servidor. Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

Ventajas

- Centralización del control
- Escalabilidad
- Fácil mantenimiento
- Varias tecnológica desarrolladas para este tipo de modelo

Desventajas

- Congestión de tráfico.
- Software y hardware de un servidor son muy determinantes, puede requerir muchos recursos.
- El cliente puede no disponer de los recursos que puedan existir en el servidor.

1.6 Modelado a través de UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Diagramas que pueden obtenerse de UML

Los **Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los **Diagramas de Comportamiento** enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estados

Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x)
- Diagrama de tiempos (UML 2.0)
- Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0) [G].

1.7 Programación por capas

Existen muchas prácticas de programación, dependiendo del tipo de software que se va a desarrollar y de la disciplina de programación que se utilice en el desarrollo del producto.

Una de las más utilizadas se llama la programación por capas, que consisten en dividir el código fuente según su funcionalidad principal.

Se organiza principalmente en 3 capas (ver figura 1.6):

- La capa de presentación o frontera.
- La capa lógica de negocio o control.
- La capa de datos.

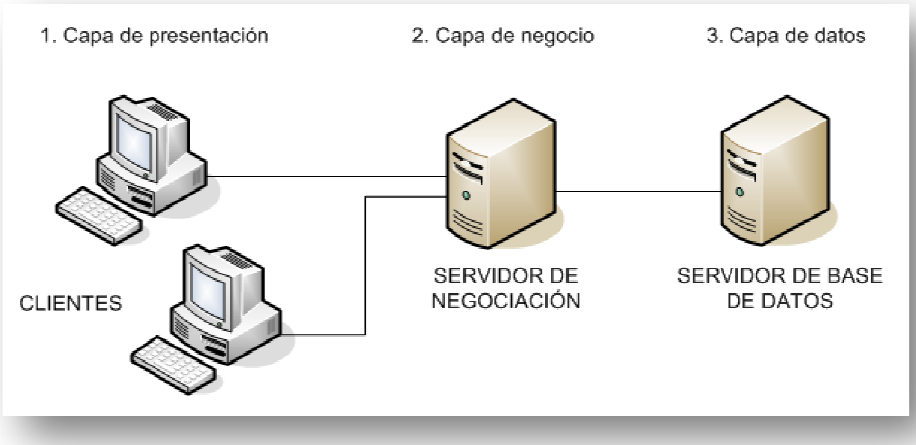


Figura 1.6 Vista de una programación por capas

Al dividirlo de esta forma, el desarrollador se asegura avanzar en la programación del proyecto de una forma ordenada, lo cual beneficia a la reducción de tiempo ya que cada capa puede ser tratada de forma independiente.

Así mismo, una característica más de esta práctica es la facilidad de actualizaciones para la aplicación.

Capa de Presentación o frontera.
Es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información. Esta capa se comunica únicamente con la capa de negocio.

Capa de Lógica de Negocio o Control.
Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y envía las respuestas. Aquí se establecen todas las reglas que deben cumplirse.

Esta capa se comunica con la capa de presentación para recibir peticiones y enviar resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de el.

Capa de datos.
Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes o de recuperación de información.

C a p í t u l o 2

ANÁLISIS Y DISEÑO DEL SISTEMA

La tendencia tecnológica actual se basa en el modelo cliente servidor, ya que entre sus ventajas se encuentra que los recursos están centralizados y pueden compartirse con cualquier computadora que tenga autorización y reducir costos de hardware; se pueden implementar varios mecanismos de seguridad; una administración a nivel servidor, lo que permite fáciles actualizaciones sin que tomen partida los clientes, etc.

Así mismo los Sistemas en Web son un claro ejemplo del modelo cliente servidor, ya que un servidor es el encargado de obtener datos, procesar información y enviar los resultados a un cliente. Además los sistemas web tienen la característica de ser portables, ya que las máquinas clientes solo necesitan de un navegador web y algunos complementos en algunos casos.

El sistema que se desarrolló es un sistema que ayuda al proceso de auditorías de una organización, ya que como se comentó en la introducción, varias de las normas necesitan ver el desempeño y mediante la medición de los procesos se puede ver.

Entre las operaciones que existen en el sistema están:

- Inicio de sesión
- Administración de los datos de la organización
- Administración de puestos de la organización
- Control de usuarios que pueden entrar al sistema
- Administración de los procesos de la organización
- Administración de los indicadores
- Control de las metas de los indicadores por periodos
- Alimentación de los indicadores
- Visualización de los indicadores por medio de graficas

2.1 Planteamiento del problema

El proceso de auditoría de alguna organización es una herramienta que se utiliza para determinar el grado en que se han alcanzado los requisitos del sistema de gestión que se haya implementado.

Los sistemas de gestión son una estructura probada para la gestión y mejora continua de las políticas, procedimiento y procesos de la organización.

Y de esta forma se utilizan las normas para tener un patrón a seguir en el proceso de mejora continua.

2.2 Especificación de requerimientos

Se desarrolló un sistema que:

- Administra todos los usuarios que ingresan al sistema, con sus privilegios personalizados. Existe:
    - Un usuario que sea el administrador del Sistema
    - Usuario(s) director(es) de la organización.
    - Usuarios que se encargan de alimentar algún indicador
    - Usuario(s) que puede(n) ver indicadores
- Los procesos podrían verse como actividades que utilizan recursos y que se gestionan con el fin de permitir que los elementos de entrada se transformen en resultados. Existe una sección en la cual se pueden modificar éstos, ya que los indicadores deberán contribuir a algún proceso de la organización (pe, medirlos). Algunos ejemplos de procesos son:
    - Administración de recursos humanos
    - Administración de recursos financieros
    - Revisión directiva
- Deben existir objetivos para los indicadores y pueden existir varios indicadores para cumplir un objetivo.
- Deben existir tipos para los indicadores, algunos que son muy usados son de:
    - Eficacia
    - Competencia
    - Satisfacción
    - Control
    - Calidad
    - Otro, el cual se podrá especificar un nombre
- Al crearse algún indicador, se deberá especificar:
    - Nombre del indicador
    - Explicación del indicador
    - Tipo (ya planteados o uno nuevo)

    - Proceso al que contribuye (de los administrados con anterioridad)
    - Responsable de su medición y mejora
    - Formula que utilizará
    - Variables de la formula (que deberán ser alimentadas)
        - Las variables de la formula podrán especificarse de forma libre, con algunas reglas
    - Usuarios que pueden ver el indicador
    - Notas del indicador
- El usuario encargado de alimentar las variables de un indicador podrá seleccionar el día en que se alimenta y su valor.
- Los usuarios que pueden ver el indicador serán los seleccionados al crearlo y el director de la organización.
- Al visualizar un indicador:
    - La página deberá desplegar dos calendarios en los que se seleccionará el periodo que visualizará
    - Mostrará una gráfica con los datos evaluados
    - Mostrará una tabla con los datos evaluados, de forma más especifica

2.3 Representación del funcionamiento del sistema

El sistema trabaja con 3 distintos tipos de usuarios: Administrador del sistema, directores de la organización, y usuarios normales (ver figura 2.1).

Administrador del sistema: es el que almacenará los datos en todo el sistema, tiene abierta cualquier página para que pueda agregar, editar o eliminar cualquier registro (siempre y cuando no existan restricciones en el sistema), estos usuarios podrán ser personal de informática o de recursos humanos. Funciones:

- Inicio de sesión
- Modificar los datos de la organización
- Control de puestos
- Control de usuarios
- Control de procesos
- Control de indicadores
- Control de metas de indicadores

- Alimentar indicadores
- Visualizar indicadores

Usuario(s) director(es) de la organización: realizará la creación y visualización de los indicadores, esto para que tenga una visión de cómo se encuentran los procesos de la organización. Funciones:

- Inicio de sesión
- Modificar datos de la organización
- Control de los indicadores
- Control de metas de indicadores
- Visualizar indicadores

Usuarios normales: son todos aquellos que se encargarán de alimentar o visualizar algún(os) indicador(es) elegidos en la creación de este. Funciones:

- Inicio de sesión
- Alimentar indicador
- Visualizar indicador

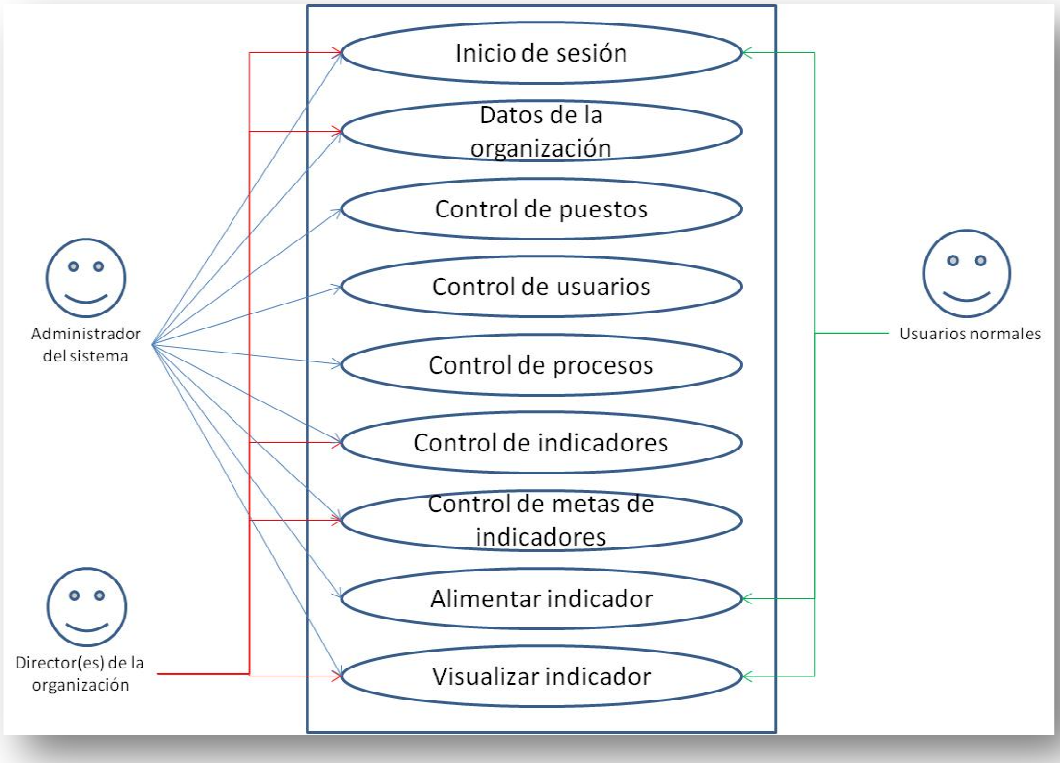


Figura 2.1 Diagrama de usuarios y sus funciones

Como se está siguiendo el estilo de programación por capas, el proyecto se dividió en 3 como se ve en la figura 2.2:

- Capa de datos, en la que se encuentra una clase que hace la conexión a la base de datos y es quien manda y recibe información directamente a la base de datos.
- Capa de negocios, es donde se encuentran todas las clases definidas en un análisis del sistema:
    - Correo: es la clase que envía correo electrónicos
    - Indicadores: clase que crea, almacena y edita indicadores
    - Metas: clase que crea, almacena y edita metas de indicadores
    - Organización: clase que edita la información de la organización
    - ProcesoFormula: clase que mediante paso de variables, genera el resultado evaluado en una formula.
    - Procesos: clase que crea, almacena y edita procesos de la organización
    - Puestos: clase que crea, almacena y edita puestos de la organización
    - Sucursales: clase que crea, almacena y edita sucursales con las que cuenta la organización
    - Usuarios: clase que crea, almacena y edita los usuarios que pueden utilizar el sistema
    - Variables: clase que crea, almacena y edita variables de la formula de un indicador

  Esta capa manda y recibe información tanto en la capa de datos como en la capa de presentación.
- Capa de presentación, es la vista del sistema y se eligió que fuese un Sistema Web, por lo que cada página es una clase. Así cada página recibe datos de entrada por el usuario, esta se comunica con la clase correspondiente de la capa de negocios, le devuelve el resultado y la muestra.

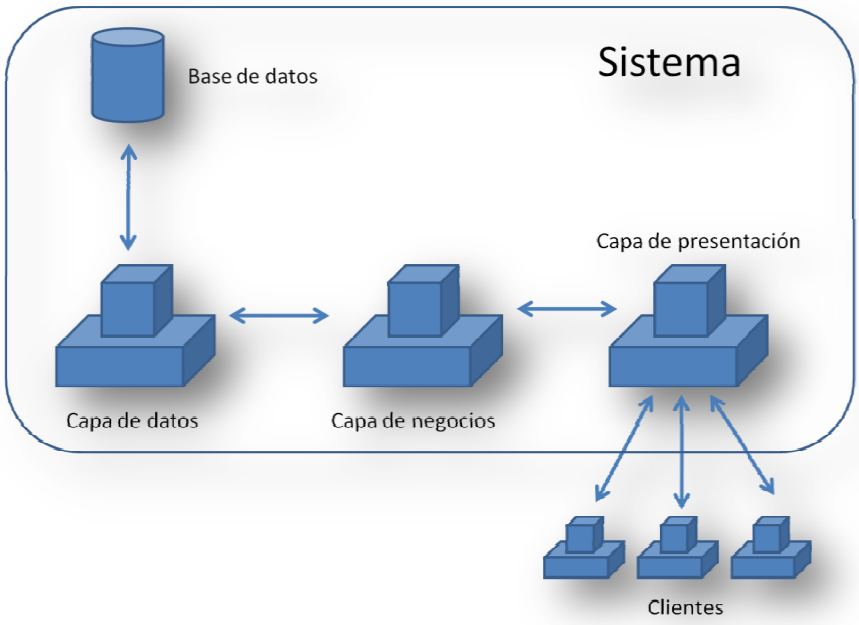


Figura 2.2 Representación del sistema por capas

2.4 Planteamiento de casos de uso

Reglas de negocio:

- Inicio de sesión
- Edición de los datos de la organización
- Control de puestos
  - Agregar puesto
  - Editar puesto
  - Elimina puesto
- Control de usuarios
  - Agregar usuario
  - Editar usuario
  - Elimina usuario
- Control de procesos
  - Agregar proceso
  - Editar proceso
  - Elimina proceso
- Control de indicadores
  - Agregar indicador
  - Editar indicador
  - Elimina indicador
- Control de metas
  - Agregar metas de un indicador
  - Editar metas de un indicador
  - Eliminar metas de un indicador

- Control de sucursales
    - Agregar sucursales
    - Editar sucursales
    - Eliminar sucursales
- Alimentar indicador
- Visualizar indicador
- Enviar correo electrónico

Descripción de los casos de uso:

- Inicio de sesión: Es el caso de uso por el que todos los usuarios deben pasar para utilizar el sistema. Este caso de uso devuelve toda la información del usuario y dependiendo de sus privilegios le quita accesos a algunas de las acciones del sistema.
- Datos de la organización: Como el sistema va a ser genérico, aquí se configura el nombre de la organización y algún mensaje que se quiera mostrar al inicio del sistema.
- Control de puestos: Como se van a tener empleados de la organización registrados, estos necesitan estar ligados a un puesto.
- Control de usuarios: Aquí se registrarán todos los empleados que podrán ingresar al sistema y se dará los privilegios que sean pertinentes.
- Control de procesos: La organización deberá tener procesos para su medición.
- Control de indicadores: Es el objetivo del sistema, se generarán tantos indicadores como se deseen para cada proceso antes registrado, en estos contendrá una fórmula para medir al proceso.
- Control de metas de los indicadores: Cada indicador deberá llevar 3 metas en un periodo, meta mínima, satisfactoria y sobresaliente; estas para conocer el estado en que se encuentra ese indicador en un periodo.
- Alimentar indicador: Periódicamente un usuario seleccionado alimentará las variables de la fórmula planteada en cada indicador.
- Visualizar indicador: Solo los usuarios seleccionados podrán visualizar los indicadores, en un periodo en que ellos señalen.

A continuación se muestra la descripción de algunos casos de uso:

Caso: 1.1 Iniciar sesión

Descripción: Este caso de uso demuestra la forma en que el usuario inicia sesión al sistema para poder utilizarlo.

HISTORIAL DE REVISIONES:

Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

Actor que inicia el caso: Usuario registrado en el sistema.

Pre-condiciones: Haber sido registrado con anterioridad en el sistema o ser el administrador del sistema.

Flujo principal:

Acción del actor	Sistema
El actor escribe en su navegador la dirección del sistema	
	Desplegará un pequeño formulario el cual pedirá en dos campos: el nombre de usuario y la contraseña
Deberá escribir su nombre de usuario y su contraseña	
	En caso de que el nombre de usuario y contraseña sean las mismas que aparecen en un registro de la base de datos, dará acceso a la página principal del sistema. En caso contrario, mostrará un mensaje de error.
Si está mal el nombre de usuario o la contraseña, el sistema desplegará un mensaje y el usuario deberá escribirla de nuevo.	

Flujos alternos:

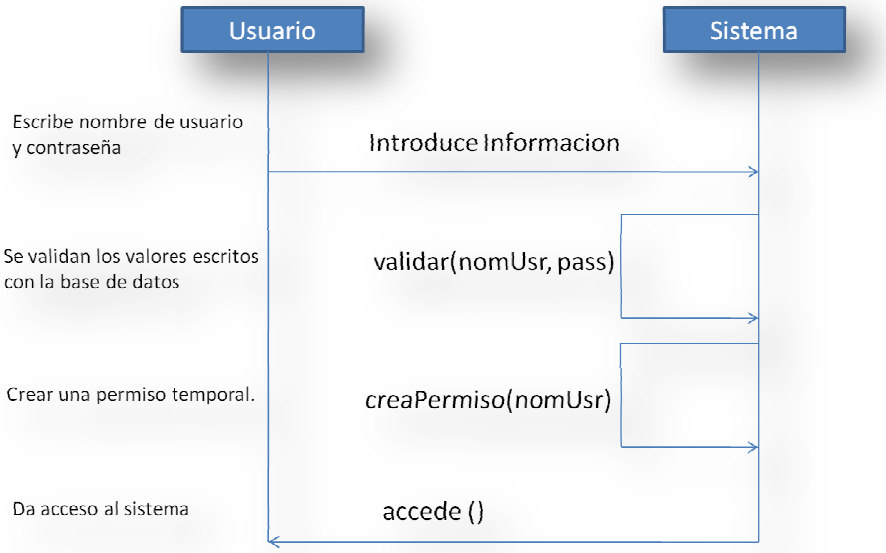
Acción del actor	Sistema
En caso de que el usuario no recuerde su contraseña, este deberá escribir su nombre de usuario y presionar el botón ¿Olvidaste tu contraseña?	

Si el nombre de usuario es correcto, se deberá esperar la contestación por medio de un correo electrónico el cual le enviará una nueva contraseña.	Verificará que el nombre de usuario se encuentre en la base de datos, si este es el caso, cambiara la contraseña por una aleatoria y la enviará a su correo electrónico. En caso contrario, mostrará un mensaje de error.

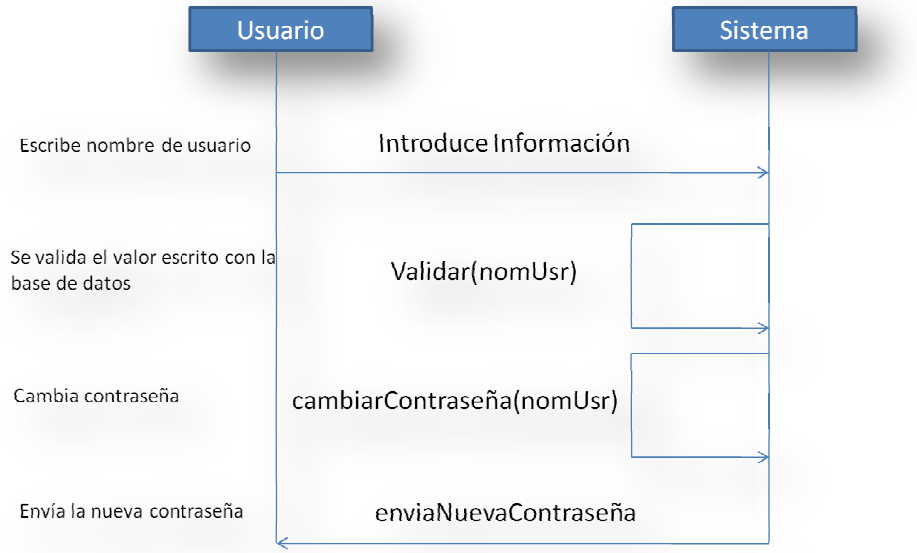
Requerimientos especiales: Estar registrado en el sistema.

Diagramas de secuencia

FLUJO PRINCIPAL



FLUJO ALTERNO



Caso: 1.2 Editar datos de la organización

Descripción: Este caso de uso demuestra cómo el usuario administrador podrá modificar los datos de la organización

HISTORIAL DE REVISIONES:

Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

Actor que inicia el caso: Usuario administrador del sistema.

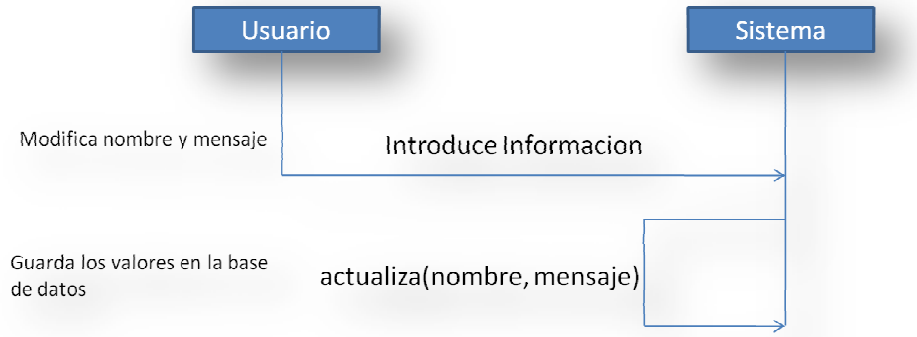
Pre-condiciones: Haber iniciado sesión como administrador del sistema.

Flujo principal:

Acción del actor	Sistema
El actor deberá hacer clic en la liga de Organización	
	El sistema desplegará un campo con el nombre de la organización y otro más en el que escribirá el mensaje que se quiera dar en la primera pagina luego del inicio de sesión.
El actor modificará los campos que desee.	

Podrá guardar los cambios o cancelar.	
	Si selecciona guardar los cambios, el sistema actualizará los datos. Si selecciona cancelar, el sistema se regresará a la página principal.

Diagrama de secuencia



Caso: 1.3.1 Agregar puesto

Descripción: Este caso de uso muestra la forma en que se va a agregar un puesto

HISTORIAL DE REVISIONES:

Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

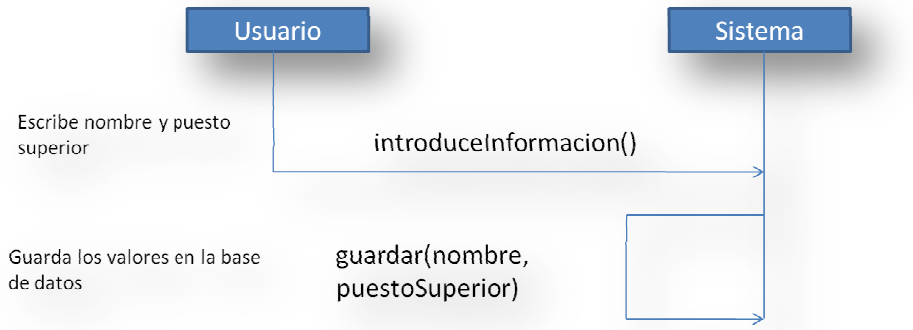
Actor que inicia el caso: **Usuario administrador** del sistema

Pre-condiciones: Haber **iniciado sesión** como administrador de sistema

Acción del actor	Sistema
	Se desplegará un **formulario** con los siguientes **campos**: • **Nombre** • **Puesto superior** (contendrá todos los puestos anteriormente creados y uno más especificando que no tiene)

Deberán llenarse los campos y seleccionar Guardar o Cancelar.	
	Si se oprime guardar, los campos serán almacenados en la base de datos. Si se cancela, se regresará a la página de Administración de puestos

Diagrama de secuencia



Caso: 1.3.2 Editar puesto

Descripción: El caso de uso muestra la forma en que se va a editar un puesto

HISTORIAL DE REVISIONES:

Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

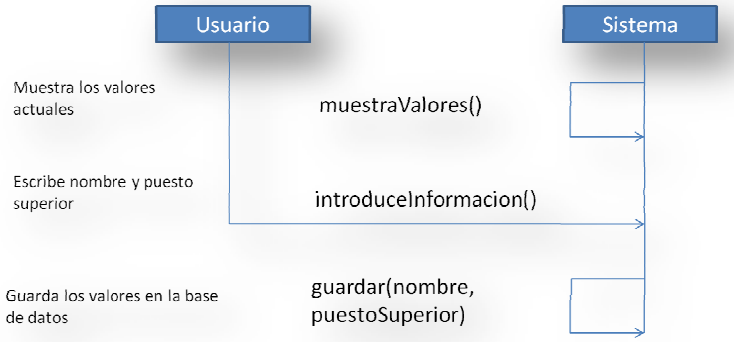
Actor que inicia el caso: Usuario administrador del sistema.

Pre-condiciones: Haber iniciado sesión como administrador del sistema.

Acción del actor	Sistema
	Se desplegará un formulario con los siguientes campos: • Nombre • Puesto superior (contendrá todos los puestos anteriormente creados y uno más especificando que no tiene)

Deberán modificarse los campos si es deseado y podrá seleccionar Guardar o Cancelar.	
	Si se oprime guardar, los campos serán actualizados en la base de datos. Si se cancela, se regresará a la página de Administración de puestos.

Diagrama de secuencia



Caso: 1.3.3 Eliminar puesto

Descripción: El caso de uso muestra cómo eliminar un puesto

HISTORIAL DE REVISIONES:

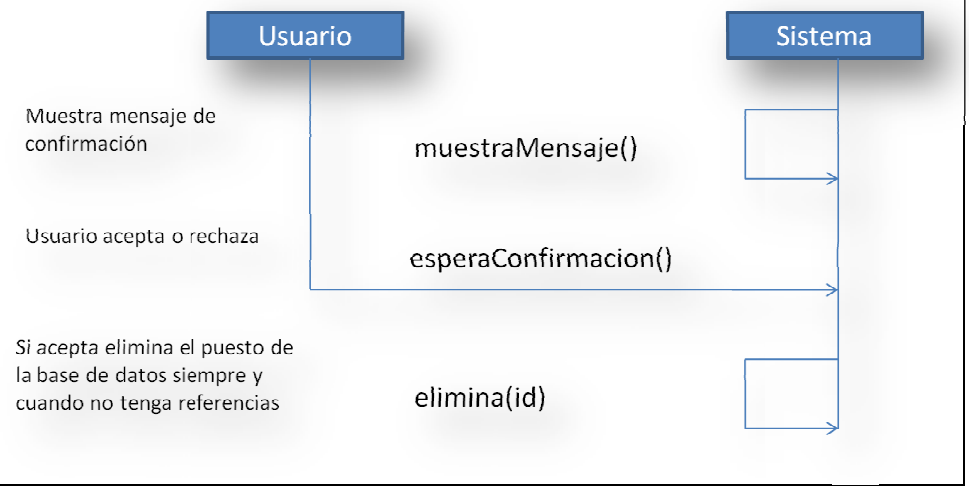
Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

Actor que inicia el caso: Usuario administrador del sistema.

Pre-condiciones: Haber iniciado sesión como administrador del sistema.

Acción del actor	Sistema
	Se desplegará un mensaje de confirmación.
o Si se acepta	Se eliminará el registro.
o Si se cancela	Se regresará a la lista.

Diagrama de secuencia:



Caso: 1.6 Ver historial

Descripción: El caso de uso muestra la forma que se podrá ver el historial de indicador. Esta es la parte final, ya que aquí todos los datos son unidos y mostrados de una forma entendible.

HISTORIAL DE REVISIONES:

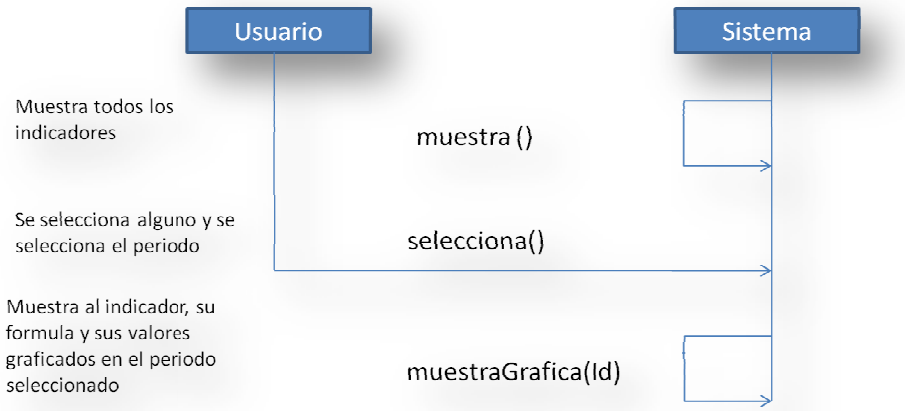
Fecha	Versión	Descripción	Autor
16-07-2009	1.0	Creación del documento	Julio Alberto Rojas

Actor que inicia el caso: Usuario registrado en el sistema.
Pre-condiciones: Haber iniciado sesión en el sistema.

Acción del actor	Sistema
El actor hará clic en Ver el historial	
	Mostrará una lista con los indicadores que tiene permitido visualizar.
Seleccionará un indicador	
	Mostrará un semáforo para ver el estado del indicador así como también una gráfica en la que se podrá observar el progreso del indicador a través de los días con diferentes opciones de visualización: • Fecha de inicio • Fecha final

	• Ver metas • Ver tendencia
El usuario podrá modificar las opciones de visualización.	

Diagrama de flujo:



Requerimientos de hardware

- El sistema servidor estará formado por una máquina Windows Server con el servicio de IIS activado y el Sistema Gestor de Base de Datos Mysql. Este servidor dará el servicio Web que podrá ser visto por el navegador de internet de cada máquina conectado a éste.
- El sistema cliente es una máquina que tenga un navegador de internet y este pueda conectarse al servidor.

2.5 Modelo entidad relación

Mediante el análisis, se encontraron las siguientes entidades:

- Organización: Es una pequeña entidad en la que se va a almacenar solamente el nombre de la organización y el mensaje de inicio para los usuarios.
- Procesos: Es la entidad que da descripción a un indicador.
- Indicadores: Es donde se van a almacenar todos los indicadores.
- Variables: Los indicadores tienen una fórmula, y cada una de ellas está formada por variables, aquí se almacenan éstas.

- Valores de variables: cada variable de la formula de un indicador debe ser alimentada, aquí se almacenan sus valores conforme a fechas.
- Sucursales: La organización puede tener sucursales, y estas pueden tener varios indicadores, aquí se almacenan.
- Metas: Los indicadores tienen metas por periodos, aquí se almacenan.
- Puesto: Almacena los puestos de la organización.
- Usuarios: Almacena los usuarios que pueden utilizar el sistema.
- Usuarios que ven indicadores: Almacena todos los usuarios con la capacidad de observar el comportamiento de un indicador.

Su representación gráfica se puede ver en la figura 2.3.

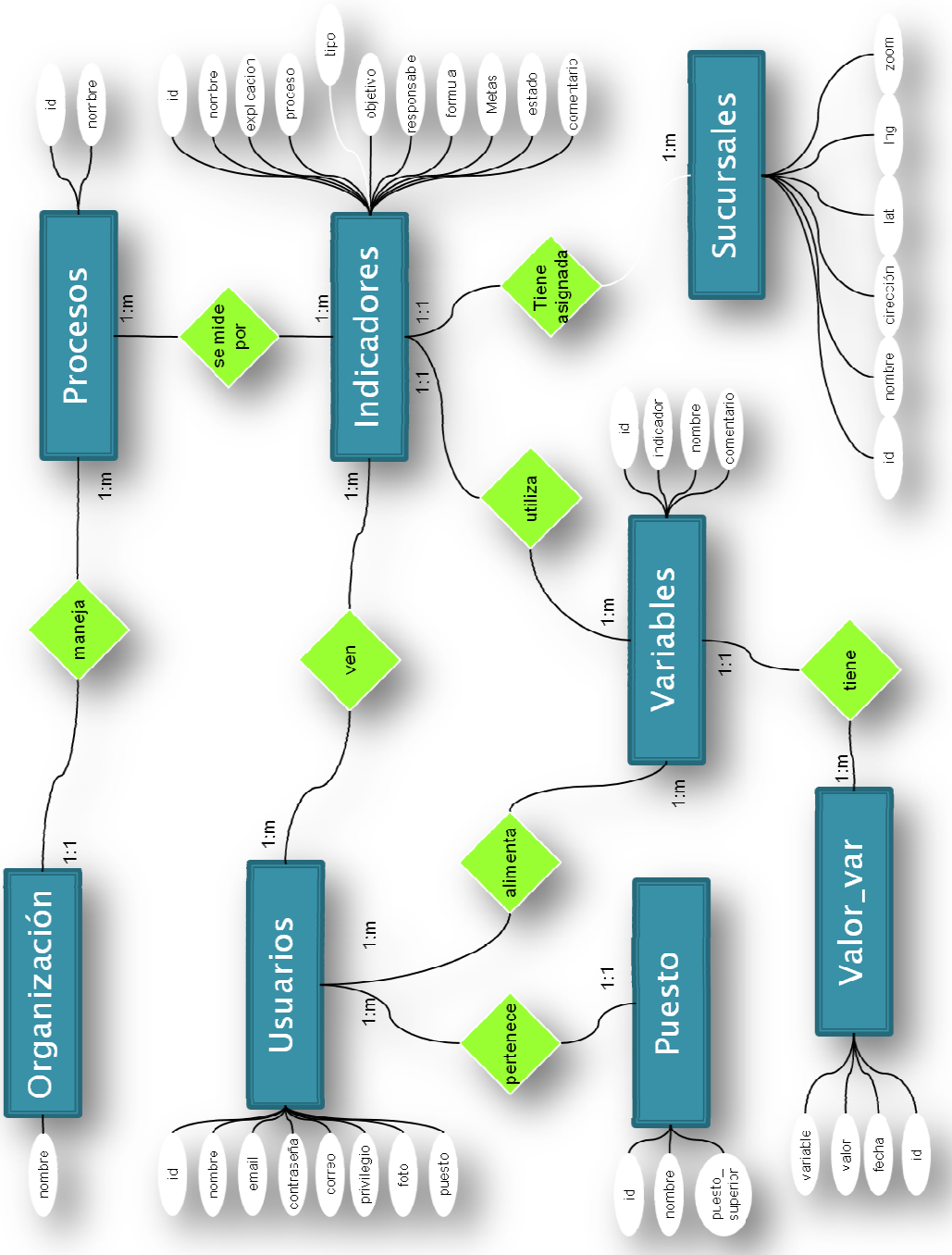


Figura 2.3 Modelo entidad relación

2.6 Diagrama de clases

2.7 Modelo relacional

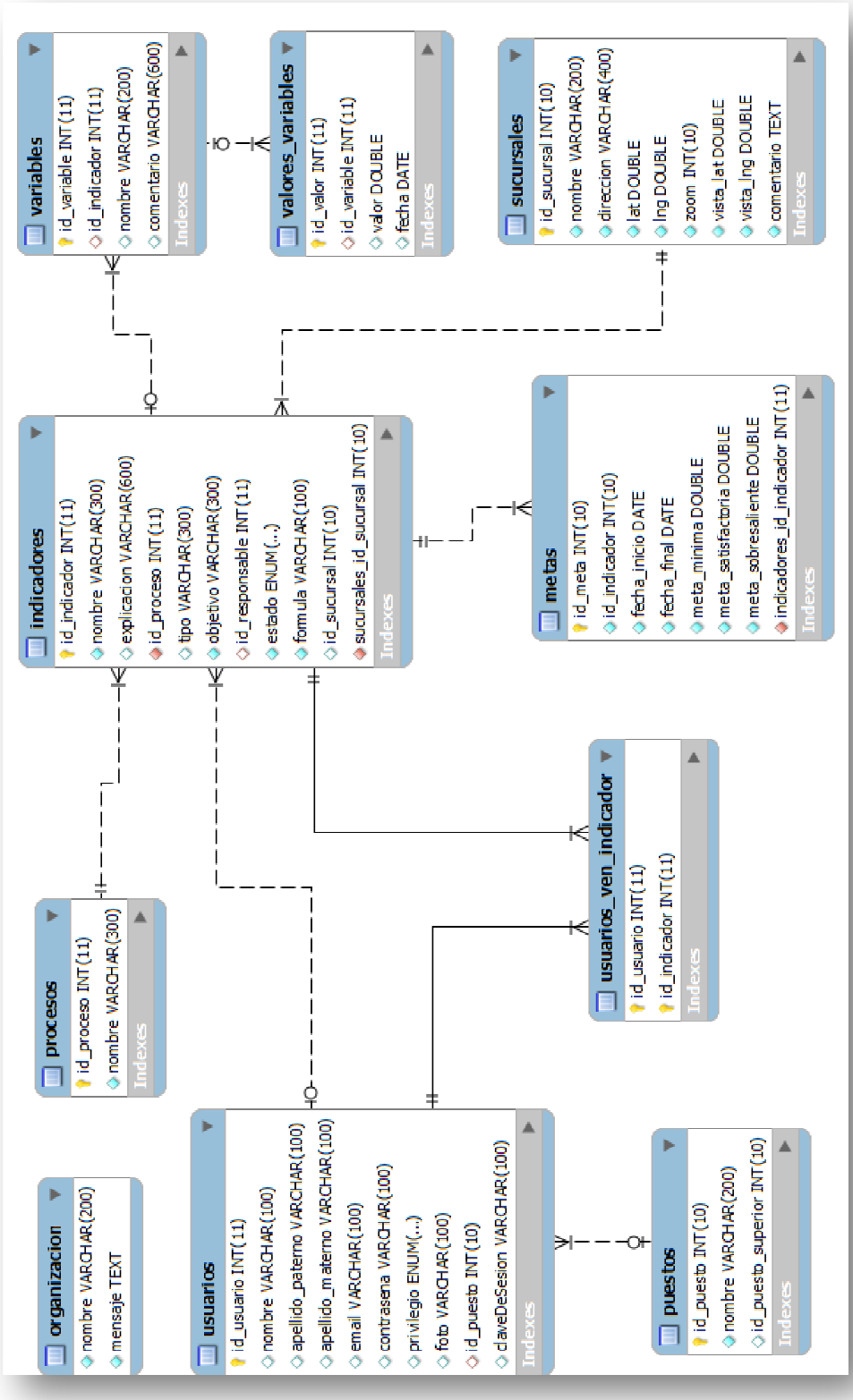


Figura 2.5 Modelo relacional

2.8 Normalización de la base de datos

Cada tabla pasó por el proceso de normalización, vamos a hacer el ejemplo de la tabla indicadores:

Primera forma normal 1FN

- Todos los atributos son atómicos, ya que ninguno de los elementos como nombre, explicación, procesos, etc. son indivisibles.
- La tabla contiene la llave primaria id_indicador
- Nunca puede existir una llave primaria con valor nulo ya que se agrega automáticamente con un valor autoincremental

Segunda forma normal 2FN

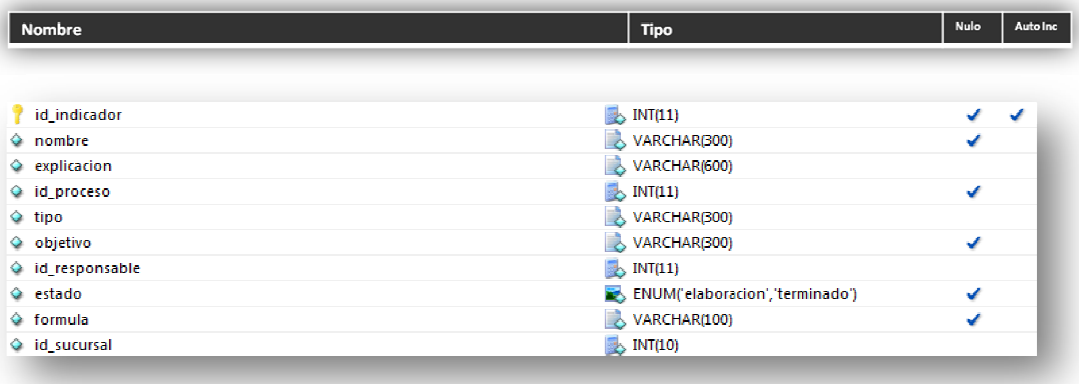
- Está en la primera forma normal
- Los atributos de la tabla nombre, explicación, tipo, objetivo, estado, formula dependen únicamente de la llave primaria id_indicador, los restantes como son id_proceso, id_responsable, id_sucursal dependen de otras llaves que están en sus respectivas tablas.

Tercera forma normal 3FN

- Esta en primera y segunda forma normal
- Ningun atributo de la tabla indicadores depende de otro atributo no llave (no existen dependencias funcionales)

2.9 Diccionario de datos

Indicadores



Nombre	Tipo	Nulo	Auto Inc
id_indicador	INT(11)	✓	✓
nombre	VARCHAR(300)	✓	
explicacion	VARCHAR(600)		
id_proceso	INT(11)	✓	
tipo	VARCHAR(300)		
objetivo	VARCHAR(300)	✓	
id_responsable	INT(11)		
estado	ENUM('elaboracion','terminado')	✓	
formula	VARCHAR(100)	✓	
id_sucursal	INT(10)		

Metas

Nombre	Tipo	Nulo	Auto Inc
id_meta	INT(10)	✓	✓
id_indicador	INT(10)	✓	
fecha_inicio	DATE	✓	
fecha_final	DATE	✓	
meta_minima	DOUBLE	✓	
meta_satisfactoria	DOUBLE	✓	
meta_sobresaliente	DOUBLE	✓	

Organización

Nombre	Tipo	Nulo	Auto Inc
nombre	VARCHAR(200)	✓	
mensaje	TEXT	✓	

Procesos

Nombre	Tipo	Nulo	Auto Inc
id_proceso	INT(11)	✓	✓
nombre	VARCHAR(300)	✓	

Puestos

Nombre	Tipo	Nulo	Auto Inc
id_puesto	INT(10)	✓	✓
nombre	VARCHAR(200)	✓	
id_puesto_superior	INT(10)		

Sucursales

Nombre	Tipo	Nulo	Auto Inc
id_sucursal	INT(10)	✓	✓
nombre	VARCHAR(200)	✓	
direccion	VARCHAR(400)	✓	
lat	DOUBLE	✓	
lng	DOUBLE	✓	
zoom	INT(10)	✓	
vista_lat	DOUBLE	✓	
vista_lng	DOUBLE	✓	
comentario	TEXT	✓	

Usuarios

Nombre	Tipo	Nulo	Auto Inc
id_usuario	INT(11)	✓	✓
nombre	VARCHAR(100)		
apellido_paterno	VARCHAR(100)		
apellido_materno	VARCHAR(100)		
email	VARCHAR(100)		
contrasena	VARCHAR(100)		
privilegio	ENUM('admin','direct','usuario')		
foto	VARCHAR(100)		
id_puesto	INT(10)		
claveDeSesion	VARCHAR(100)		

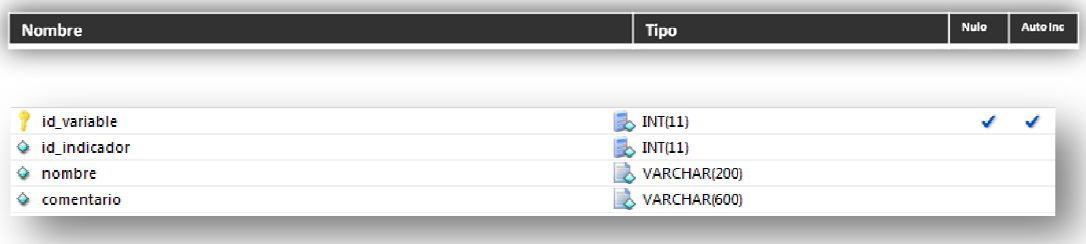
Usuarios_ven_indicador

Nombre	Tipo	Nulo	Auto Inc
id_usuario	INT(11)	✓	
id_indicador	INT(11)	✓	

Valores_variables

Nombre	Tipo	Nulo	Auto Inc
id_valor	INT(11)	✓	✓
id_variable	INT(11)		
valor	DOUBLE		
fecha	DATE		

Variables



Nombre	Tipo	Nulo	Auto Inc
id_variable	INT(11)	✓	✓
id_indicador	INT(11)		
nombre	VARCHAR(200)		
comentario	VARCHAR(600)		

C a p í t u l o 3

IMPLEMENTACIÓN

Indika es un sistema que pretende ayudar a las organizaciones a llevar un control de la misma, así mismo ayudarla en proceso de auditorías a auditores.

Debe ser un sistema que esté al alcance de todos, por lo que se planeó hacerlo en un ambiente web.

3.1 Herramientas de implementación

Las herramientas que se utilizaron para la implementación del sistema fueron:

- Microsoft® Visual C#® 2008: es la herramienta de desarrollo que llevo el control del proyecto y donde se codificó la mayor parte del sistema.
- Microsoft .NET Framework 3.5 SP1: componente de Windows que gestiona la ejecución de programas y de esta forma sea seguro.
- Microsoft Chart Controls para .NET Framework 3.5: serie de controles para ser integrados en aplicaciones ASP.NET y genere distintos tipos de gráficas.
- ASP.NET Ajax Library: Es la biblioteca que permite trabajar con Ajax dentro de ASP.NET
- GoogleMaps.Subgurim.NET: Es una librería que permite trabajar con los mapas de google maps
- TSHAK.Components: es una librería que permite encriptar cadenas
- jQuery: es una librería de javascript, el cual permite manejar varios componentes de forma dinámica y vistosa
- Sexy Lightbox 2: es una librería de javascript el cual hace efectos de popups dentro de la página, necesita de jQuery
- Mysql Connector/Net: es el conector entre la base de datos de Mysql y ASP:NET

3.2 Herramientas de hardware

Para el desarrollo del sistema se utilizó una máquina con las siguientes características:

- Windows Xp
- 1Gb de memoria Ram
- Disco duro de 50 Gb
- Procesador Amd Athlon 64 X2 a 1.8 GHz

Para su funcionamiento, se necesita de un servidor con las siguientes características:

- Windows Xp profesional o Windows server 2003 / 2008
- Internet Information Server 6 / 7
- Microsoft .Net Framework 3.5 sp1
- 1 GB de espacio en disco
- 512 Mb de memoria Ram

Así mismo, se necesita de una máquina cliente con las siguientes características:

- Windows Xp o mayor
- Internet Explorer 7 o mayor, o cualquier otro navegador con compatibilidad para framework .NET
- 512 Mb de memoria Ram
- Conexión a internet o a la red de tal forma que pueda conectarse al servidor

3.3 Funciones del sistema

A continuación se listan todas las funciones del sistema:

- Inicio de sesión
- Cambiar contraseña
- Administrar información de la organización
- Puestos
    - Agregar puestos
    - Editar puestos
    - Eliminar puestos
- Usuarios
    - Agregar usuarios
    - Editar usuarios
    - Eliminar usuarios

- Procesos
    - Agregar procesos
    - Editar procesos
    - Eliminar procesos
- Sucursales
    - Agregar sucursales
    - Editar sucursales
    - Eliminar sucursales
- Indicadores
    - Agregar indicadores
    - Editar indicadores
    - Eliminar indicadores
    - Agregar metas de indicadores
    - Editar metas de indicadores
    - Eliminar metas de indicadores
    - Alimentar indicadores
    - Visualizar el estado de los indicadores

La lógica y la representación de las páginas están en archivos independientes. Así tenemos dos archivos diferentes para cada página web, uno con extensión .aspx que es la representación o vista al usuario y otro con extensión .aspx.cs que es la parte lógica o comportamiento de la página.
Así mismo se tiene la capa de negocios y la capa de datos.

De esta forma veremos el código de algunas funciones del sistema:

En la capa de datos tenemos una clase genérica que se encarga de las transacciones de la base de datos, conexionMySQL.cs:

```
// Agregamos referencias a tipos en un espacio de nombres
using System;
using System.Collections.Generic;
using System.Data;
using MySql.Data.MySqlClient;
using System.Configuration;
using System.Web;

public class conexionMySQL
{
    String cadenaConexion;
    MySqlConnection conexion; // Creamos una conexión con MySql
    public List<string> consultas; // Creamos una lista donde se guardan
las consultas a la base de datos

    public conexionMySQL()
    {
        // Asignamos null a la cadena de conexión y la conexión
         cadenaConexion = null; >();
         conexion = null;
```

```
        // Creamos el objeto consulta que es una lista de strings
         consultas = new List<string


        // Obtenemos la cadena de conexión de la configuracion del
       sistema.
         cadenaConexion =
System.Web.Configuration.WebConfigurationManager.ConnectionStrings["cadena
DeConexion"].ConnectionString;



    }

    // Creamos el metodo conectar que hara la conexión con la base de
   datos

    public void conectar()
    {
        if (cadenaConexion == null)
        {
            conexion = null;
            throw new ErrorBaseDatos("Error en la cadena de conexion");
        }

        if (conexion == null)
        {
            try
            {
                conexion = new MySqlConnection(cadenaConexion);
                conexion.Open();
            }
            catch (Exception)
            {
                conexion = null;
                throw new ErrorBaseDatos("No se Puede Conectar a la Base
de Datos");
            }
        }
        else
        {
            if (conexion.State == ConnectionState.Closed || conexion.State
== ConnectionState.Broken)
            {
                try
                {
                    conexion.Open();
                }
                catch (Exception)
                {
                    conexion = null;
                    throw new ErrorBaseDatos("No se Puede Conectar a la
Base de Datos");
                }
            }
        }
    }


    // Creamos la función que ejecuta todas las cadenas almacenadas en la
   variable consultas

    public string ejecutaTransaccion()
    {
        string cad;
        int i;

        conectar();
        if (conexion == null)
        {
            return "Error al tratar de conectarse a la base de Datos.";
```

```
        }
        MySqlCommand myCommand = conexion.CreateCommand();
        MySqlTransaction myTrans;

        myTrans = conexion.BeginTransaction();
        myCommand.Connection = conexion;
        myCommand.Transaction = myTrans;

        try
        {
            for (i = 0; i < consultas.Count; i++)
            {
                cad = consultas[i];
                myCommand.CommandText = cad;
                myCommand.ExecuteNonQuery();
            }
            myTrans.Commit();
        }
        catch (Exception e)
        {
            try
            {
                myTrans.Rollback();
                return e.ToString();
            }
            catch (MySqlException ex)
            {
                if (myTrans.Connection != null)
                {
                    return ex.ToString();
                }
            }
        }
        finally
        {
            cerarConexion();
        }
        limpiarConsultas();
        return "OK";
    }

    // Agregamos una consulta a la lista consultas

    public void insertarConsulta(string consulta)
    {
        consultas.Add(consulta);
    }

    // Borramos todo el contenido de la lista consultas

    public void limpiarConsultas()
    {
        consultas.Clear();
    }

    // Obtenemos la respuesta de la base de datos de acuerdo a la consulta

    public MySqlDataReader obtenerResultados(string consulta)
    {
        MySqlDataReader reader = null;
        MySqlCommand myCommand;
        conectar();
        if (conexion == null)
        {
            return null;
        }
        myCommand = new MySqlCommand(consulta, conexion);
        reader = myCommand.ExecuteReader();
        return reader;
    }
```

```
    // Cerramos la conexión con la base de datos

    public void cerarConexion()
    {
        try
        {
            conexion.Close();
        }
        catch
        {

        }
        conexion = null;
    }

}
```

Agregar Indicador

En la capa de negocios tenemos la clase indicador con el siguiente método:

```
public String agregar(String nombre, String explicacion, String procesoId,
String tipo, String objetivo, String responsableId, String estado, String
sucursalId)
        {
            conexionMySQL conn = new conexionMySQL();

            conn.insertarConsulta("INSERT INTO indicadores
(id_indicador,nombre,explicacion,id_proceso,tipo,objetivo,id_responsable,e
stado, id_sucursal) VALUES(" +
                    "default,'" +
                    nombre + "','" +
                    explicacion + "','" +
                    procesoId + "','" +
                    tipo + "','" +
                    objetivo + "','" +
                    responsableId + "','" +
                    "elaboracion','" +
                    sucursalId + "'" +
                    ")");

            String resultado = conn.ejecutaTransaccion();
            conn.cerarConexion();
            if (resultado == "OK")
            {
                return "OK";
            }
            else
            {
                return "error " + resultado;
            }
        }
```

Este trozo de código realiza lo siguiente:

- Crea el objeto conn de la clase conexionMysql
- Manda a llamar al método insertarConsulta de la clase conexionMysql con sus respectivos parámetros
- Si la transacción fue exitosa, devolvemos un mensaje de confirmación, si no, un mensaje de error.

Así mismo tenemos en la capa de presentación el archivo agregarIndicador.aspx que contiene todos los elementos que interactúan con el usuario:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="agregarIndicador.aspx.cs" Inherits="sistema_agregarIndicador"
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div id="mensaje" runat="server" visible="true">
    </div>
    <div style="text-align:center">
    <p>
        <label>Nombre: </label><asp:TextBox ID="nombre"
runat="server"></asp:TextBox>
    </p>
    <p>
        <label>Explicacion: </label><asp:TextBox ID="explicacion"
runat="server"></asp:TextBox>
    </p>
    <p>
        <label>Proceso: </label>
        <asp:DropDownList ID="procesos" runat="server"
DataSourceID="SqlDataSource2"
            DataTextField="nombre" DataValueField="id_proceso">
        </asp:DropDownList>

        <asp:SqlDataSource ID="SqlDataSource2" runat="server"
            ConnectionString="<%$ ConnectionStrings:cadenaDeConexion %>"
            ProviderName="<%$
ConnectionStrings:cadenaDeConexion.ProviderName %>"
            SelectCommand="SELECT id_proceso, nombre FROM procesos ORDER
BY nombre">
        </asp:SqlDataSource>

    </p>
    <p>
        <label>Tipo: </label><asp:TextBox ID="tipo"
runat="server"></asp:TextBox>
    </p>
    <p>
        <label>Objetivo: </label><asp:TextBox ID="objetivo"
runat="server"></asp:TextBox>
    </p>
    <p>
        <label>Responsable: </label>
        <asp:DropDownList ID="responsable" runat="server"
DataSourceID="SqlDataSource3"
            DataTextField="nombre" DataValueField="id_usuario">
        </asp:DropDownList>
        <asp:SqlDataSource ID="SqlDataSource3" runat="server"
            ConnectionString="<%$ ConnectionStrings:cadenaDeConexion %>"
            ProviderName="<%$
ConnectionStrings:cadenaDeConexion.ProviderName %>"
            SelectCommand="SELECT id_usuario, CONCAT(nombre, ' ',
apellido_paterno, ' ', apellido_materno) AS nombre FROM usuarios ORDER BY
nombre">
        </asp:SqlDataSource>
    </p>
```

```
    <p>
        <label>Sucursal: </label>
        <asp:DropDownList ID="sucursal" runat="server"
DataSourceID="SqlDataSource1"
            DataTextField="nombre" DataValueField="id_sucursal">
        </asp:DropDownList>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:cadenaDeConexion %>"
            ProviderName="<%$
ConnectionStrings:cadenaDeConexion.ProviderName %>"
            SelectCommand="SELECT id_sucursal, CONCAT(nombre, ' (',
direccion, ')') AS nombre FROM sucursales ORDER BY nombre">
        </asp:SqlDataSource>
    </p>
    <br />
    <asp:Button ID="guardar" runat="server" Text="Guardar y continuar"
        onclick="guardar_Click" />

    </div>
    </form>
</body>
</html>
```

Y cada uno de los elementos tiene asignada una acción en el archivo agregarIndicador.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MySql.Data.MySqlClient;

public partial class sistema_agregarIndicador : System.Web.UI.Page
{
    String indicadorId = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        indicadorId = datos.obtenIdPagina(Request,Response,"id");
        if (indicadorId != "null" && !Page.IsPostBack)
        {
            negocios.indicadores i = new negocios.indicadores();
            i.obtenPorId(indicadorId);

            nombre.Text = i.nombre;
            explicacion.Text = i.explicacion;
            procesos.SelectedValue = i.procesoId;
            tipo.Text = i.tipo;
            objetivo.Text = i.objetivo;
            responsable.SelectedValue = i.responsableId;

        }
    }
    protected void guardar_Click(object sender, EventArgs e)
    {

        if (nombre.Text == "" || explicacion.Text == "" || tipo.Text ==
"" || objetivo.Text == "")
        {
            mensaje.InnerText = "Debes llenar todos los campos";
        }
        else
        {
            conexionMySQL conn = new conexionMySQL();
            negocios.indicadores i = new negocios.indicadores();
            if (indicadorId == "null")
            {
```

```
                i.agregar(nombre.Text, explicacion.Text,
procesos.SelectedValue, tipo.Text, objetivo.Text,
responsable.SelectedValue,"elaboracion",sucursal.SelectedValue);
                indicadorId =
i.obtenPorDatos(nombre.Text,explicacion.Text,objetivo.Text);
            }
            else
            {
                i.editar(indicadorId, nombre.Text, explicacion.Text,
procesos.SelectedValue, tipo.Text, objetivo.Text, sucursal.SelectedValue,
responsable.SelectedValue);
            }

            String resultado = conn.ejecutaTransaccion();
            if (resultado == "OK")
            {
                mensaje.Visible = true;
                mensaje.InnerText = "Registro modidicado";


Response.Redirect("generarFormula.aspx?id="+datos.encripta(indicadorId));
            }
            else
            {
                mensaje.Visible = true;
                mensaje.InnerText = "Error" + resultado;
            }
        }
    }
}
```

Como podemos observar en el código del método guardar_Click que es invocado por un botón, hace los siguientes pasos:

- Verificamos que la información haya sido introducida correctamente.
- Creamos el objeto i de la clase indicadores localizada en la capa de negocios.
- Mandamos a llamar al método i.agregar de la clase indicadores en el caso de que no exista el indicador, si ya existe, entonces manda a llamar al método i.edita de la clase indicadores y mandamos como parámetros la información escrita por el usuario.
- Si se realizó con éxito la inserción o edición del indicador, mostramos un mensaje de éxito, si no, mostramos un mensaje de error.

Capítulo 4

PRUEBAS DEL SISTEMA

Para finalizar el desarrollo del sistema, se hicieron una serie de pruebas para ver un correcto funcionamiento, basados en los objetivos planteados en el análisis y diseño del sistema.

A continuación se mostrarán algunas pantallas del sistema en funcionamiento.

4.1 Inicio de sesión

Esta es la parte por la que deben pasar todos los usuarios, ya que aquí se autentican y obtiene los permisos correspondientes.

El usuario deberá colocar su nombre de usuario (correo electrónico) y contraseña asignada para poder entrar al sistema.

Si digitalizó un nombre de usuario o contraseña invalida, el sistema le notificará que existe un error.

En caso de que se haya olvidado la contraseña, el sistema generará una nueva y la enviará al correo electrónico del usuario.

Figura 4.1 Pantalla de inicio de sesión del sistema

4.2 Página maestra

Una página maestra es como una plantilla que van a tomar todas las páginas que se quieran, es un marco de diseño.

El sistema tiene creada una página maestra que tiene los siguientes elementos:

1. Titulo de la aplicación
2. Imagen de un semáforo del lado izquierdo
3. Menú de links
4. Texto que muestra el nombre del usuario autenticado
5. Botón de cierre de sesión
6. Espacio para el contenido

Figura 4.2 Pagina maestra

4.3 Pantalla de inicio

Esta es la primera página que muestra el sistema luego de haber iniciado sesión.

Aquí solo mostramos un titulo en el espacio de contenido y un mensaje configurable que verán todos los usuarios.

Figura 4.3 Pantalla de inicio

4.4 Crear indicador

La creación de un indicador está compuesta por pasos:

4.4.1 Almacenamiento de datos generales

Aquí ponemos la descripción del indicador, como su nombre, explicación, procesos que maneja, responsable, sucursal, etc.

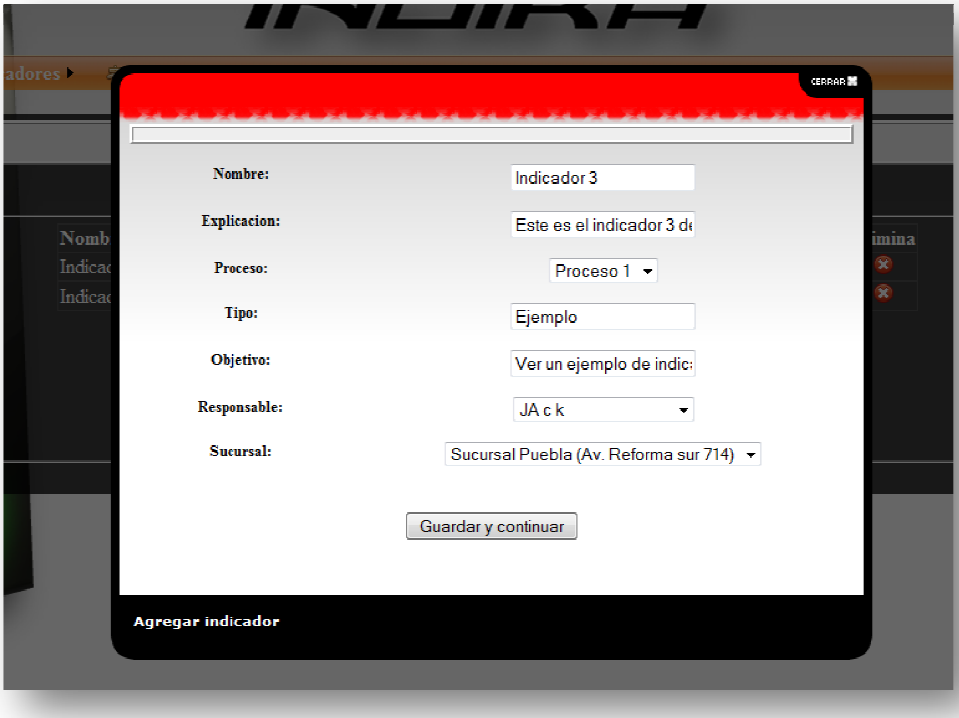


Figura 4.4 Pantalla de almacenamiento de datos generales en la creación de un indicador

4.4.2 Creación de la fórmula

Cada indicador debe tener una fórmula, esta la creamos en este formulario. Primero creamos todas las variables que vamos a utilizar y después armamos la fórmula de acuerdo a las variables creadas.

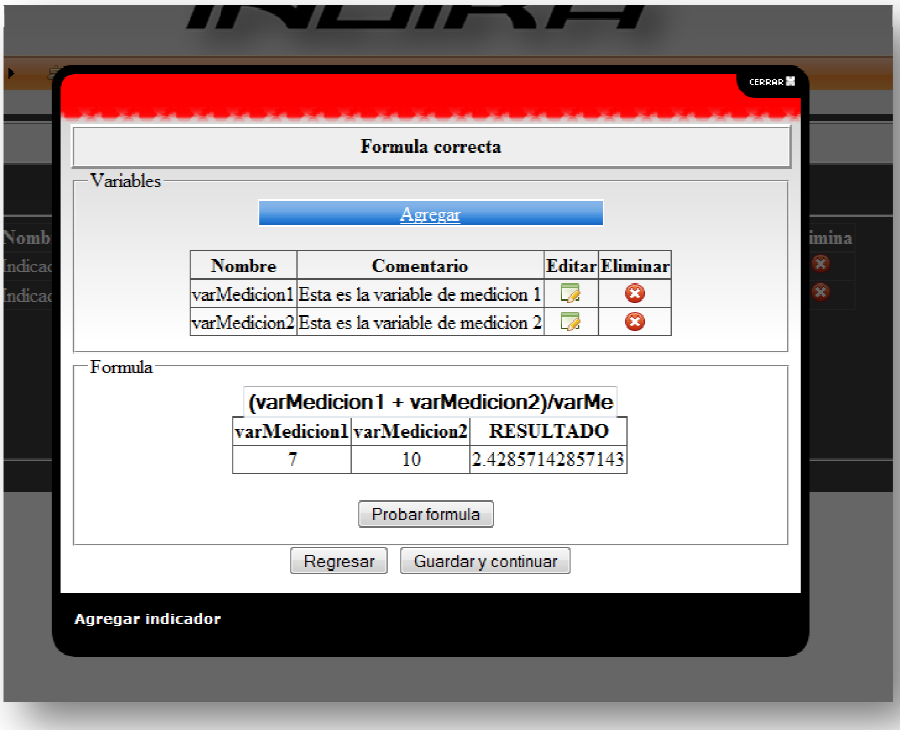


Figura 4.5 Pantalla de la creación de la formula de un indicador

4.4.3 Generación de metas

Los indicadores deben tener una meta que alcanzar para ver su progreso, aquí se configuran las metas de acuerdo a periodos.

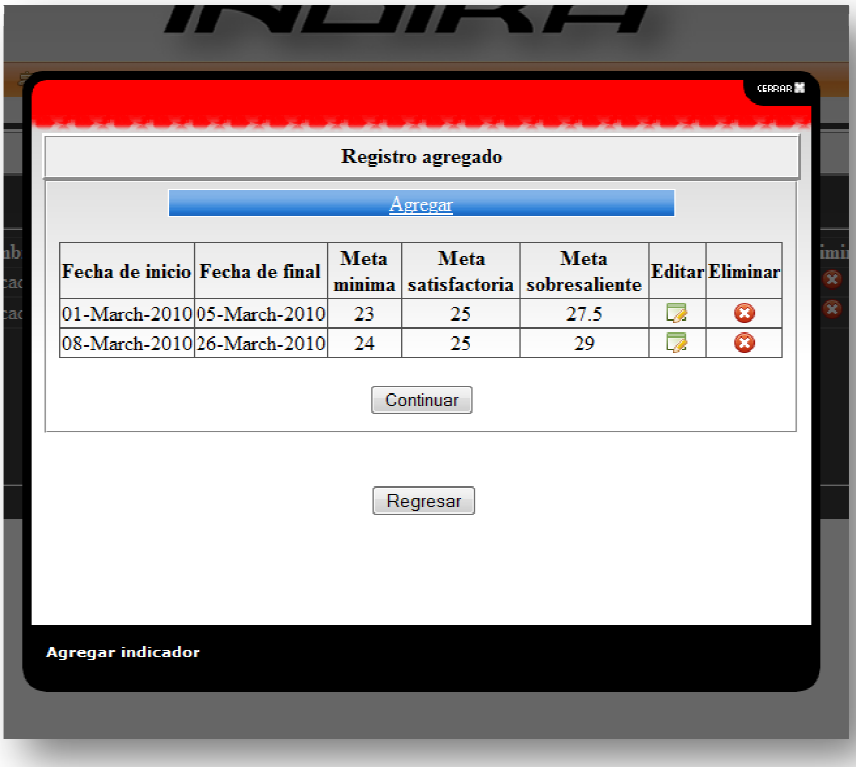


Figura 4.6 Pantalla de la creación de metas de un indicador

4.4.4 Selección de usuarios que podrán visualizar el indicador y su progreso.

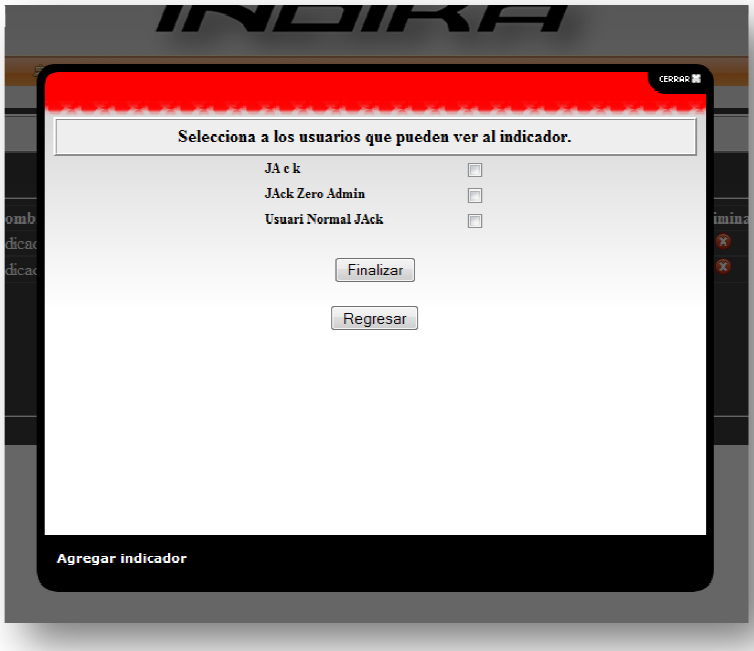


Figura 4.7 Pantalla de selección de usuarios que podrán ver un indicador

Una vez finalizada la creación del indicador, el sistema mandara un correo electrónico a todos los interesados del indicador.

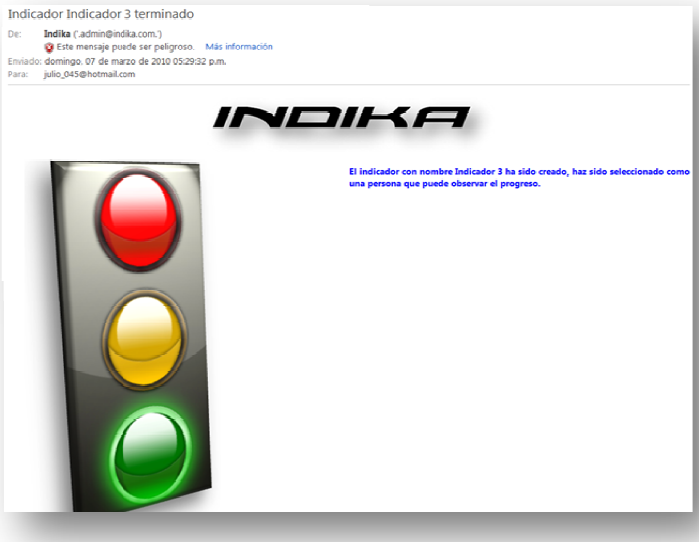


Figura 4.8 Mensajes de correo electrónico que manda el sistema una vez finalizada la creación de un indicador

4.5 Alimentar indicador.

La persona seleccionada como quien será el responsable de su seguimiento, va a ser quien lo alimente, para esto debe seleccionar el día que será alimentado y tener el valor de todas las variables para ese día.

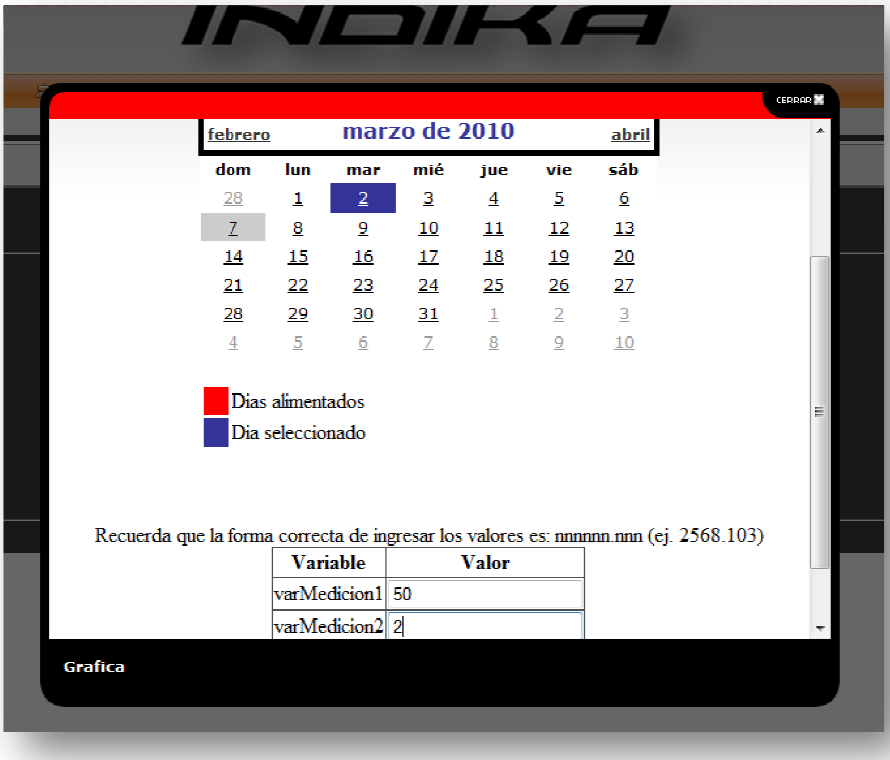
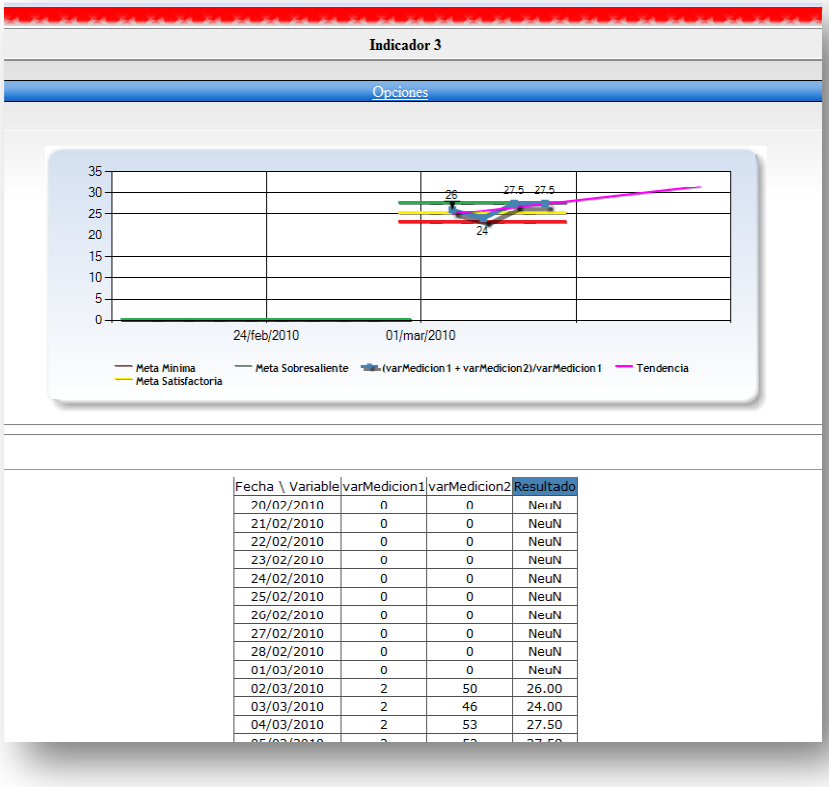


Figura 4.9 Pantalla en la cual se alimentan los valores de un indicador en una fecha

4.6 Ver el estado de un indicador

Esta página muestra una gráfica de un periodo dado, aquí, todos los datos almacenados y todas las operaciones relacionadas con el indicador se pueden visualizar.



Fecha \ Variable	varMedicion1	varMedicion2	Resultado
20/02/2010	0	0	NeuN
21/02/2010	0	0	NeuN
22/02/2010	0	0	NeuN
23/02/2010	0	0	NeuN
24/02/2010	0	0	NeuN
25/02/2010	0	0	NeuN
26/02/2010	0	0	NeuN
27/02/2010	0	0	NeuN
28/02/2010	0	0	NeuN
01/03/2010	0	0	NeuN
02/03/2010	2	50	26.00
03/03/2010	2	46	24.00
04/03/2010	2	53	27.50

Figura 4.10 Pantalla que muestra el estado de un indicador por medio de una gráfica

4.7 Gestión de errores

El sistema tiene una función global que en el caso de que exista algún error, (ya sea que el usuario lo provocó, falla en las conexiones o por falla del sistema), mostrará una pantalla de error al usuario (Figura 4.11) e inmediatamente enviará un correo electrónico al programador indicándole la dirección ip donde se encuentra el sistema y un mensaje con datos técnicos del error(Figura 4.12).

Figura 4.11 Pantalla mostrada al usuario en caso de que ocurra cualquier error no controlado

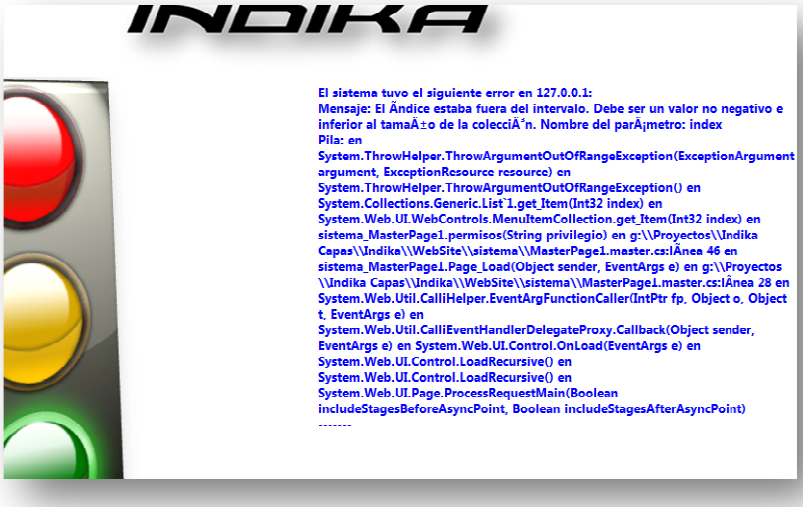


Figura 4.12 Muestra del correo electrónico que es mandado en caso de que ocurra algún error en el sistema.

CONCLUSIONES

En el desarrollo de esta tesis se logró cumplir con los objetivos generales y específicos planteados, ya que mediante la correcta creación de indicadores y una alimentación constante de este, se podrán llevar datos reales y actualizados y se podrá visualizar el desempeño y evolución de cualquier empresa.

Los requerimientos se obtuvieron de una organización, creo que hay una oportunidad de perfeccionamiento al ampliar el sistema y agregarle mejoras obteniendo requerimientos de varias organizaciones. Esto enriquecería más al sistema y las organizaciones tendrán una herramienta mucho más potente que ayude a visualizar el estado de la empresa y que ayude a auditores en los procesos de certificación.

Todo el sistema está basado en una arquitectura cliente servidor, inclusive en la programación que fue por capas. Este estilo de programación por capas permite una fácil actualización y adaptación a cambios de arquitectura ya que todo está dividido en módulos independientes.

La ventaja de utilizar ASP.NET para la creación de aplicaciones web es que dentro de su framework tiene integradas varias funciones para una fácil interacción entre el sistema operativo y el sistema que se está desarrollando, de esta forma, se pueden eliminar varios errores, se tiene más seguridad y el código no es tan extenso.

El sistema actualmente está siendo utilizado por una empresa dedicada a micro finanzas para llevar el control de indicadores requerido por normas que lleva.

De forma personal, el desarrollo de este trabajo de tesis me introdujo más en las necesidades de una organización, ya que en la toma de requerimientos surgieron muchas oportunidades de otro tipo de sistemas que podrían ayudar en las organizaciones.

De igual forma, el utilizar la ingeniería de software hace que el proyecto sea muy fácil de llevar y de programar. Este proceso lleva de requerimientos de usuarios comunes a requerimientos de software y así mediante distintas técnicas, se puede llevar a un sistema funcional y de gran ayuda al usuario final.

BIBLIOGRAFÍA

Libros

[1]

Autor: Comité Técnico de Normalización Nacional de Sistemas de Calidad *COTENNSISCAL*

Titulo: "NMX-CC-9001-IMNC-2000 Norma Mexicana para los sistemas de gestión de calidad"

Revisión 2003

[2]

Autor: Instituto Nacional de la Mujer *INMUJERES*

Titulo: "MEG:2003 Modelo de equidad de Género"

Revisión 2003

[3]

Autor: Organización Internacional para la Estandarización (ISO) y Comisión Electrotécnica Internacional (IEC)

Titulo: "ISO/IEC 27000 Sistemas de Gestión para la seguridad de la información"

Revisión 2005

[4]

Autor: Pressman S. Roger

Titulo: "Ingeniería del software. Un enfoque practico"

Editorial: MacGraw Hill 2002

País: España

Año: 2005

[5]

Autor: Raúl Monferrer Agut

Titulo: "Especificación de Requisitos Software según el estándar de IEEE 830"

Editorial: Departament d'Informàtica, Universitat Jaume

País: España

Año: 2000

[6]

Autor: Juan Antonio López Quesada

Titulo: “Fundamentos de ingeniería de software”

Editorial: Facultad de Informática Campus Universitario de Espinardo Murcia

País España

Año 2006

Referencias en internet

[A] http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-software.php

[B] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos

[C] http://www.lsi.us.es/docencia/get.php?id=28022

[D] http://es.wikipedia.org/wiki/Arquitectura_de_software

[E] http://migueljaque.com/index.php/metodologias/up/30-up/58-procesounificado

[F] http://ocw.uoc.edu/informatica-tecnologia-y-multimedia/bases-de-datos/Course_listing

[G] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelad

[H] http://es.wikipedia.org/wiki/Object_Linking_and_Embedding

[I] http://es.wikipedia.org/wiki/Active_Server_Pages

[J] http://es.wikipedia.org/wiki/HyperText_Markup_Language

[K] http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[L] http://es.wikipedia.org/wiki/Internet_Information_Services

[M] http://es.wikipedia.org/wiki/Internet

[N] http://es.wikipedia.org/wiki/Javascript

[O] http://es.wikipedia.org/wiki/AJAX

[P] http://es.wikipedia.org/wiki/Mysql

[Q] http://es.wikipedia.org/wiki/Microsoft_.NET

Referencias de figuras

Figura 0.2 http://www.camaracervecera.com.ar/proceso-de-fabricacion-de-la-cerveza.php

Figura 0.3 http://www.negociosgt.com/main.php?id=160&show_item=1&id_area=86

Figura 0.4 http://www.distriluz.com.pe/hidrandina/04_cliente/info11.html

Figura 1.1 http://www.standishgroup.com/chaos/toc.php

Figura 1.2 http://wapedia.mobi/es/Software

Figura 1.3 http://scruz334.blogspot.es/1193169600

GLOSARIO

Activex: Es un sistema de objeto distribuido y un protocolo desarrollado por Microsoft y usado por diseñadores web para utilizar archivos multimedia a paginas [H].

AJAX: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications) [O].

ASP (Active Server Pages): Es una tecnología que desarrolló Microsoft que trabaja del lado del servidor para páginas web generadas dinámicamente. Esta tecnología combina código HTML, secuencias de comando y componentes ActiveX[I].

HTML: siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web [J].

HTTP: El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW) [K]

IEC: International Electrotechnical Commission (Comisión Electrotécnica Internacional)

IEEE: Instituto de Ingenieros Electricistas y Electrónicos

IIS: Internet Information Services , IIS, es una serie de servicios para los ordenadores que funcionan con Windows, que ejecuta paginas ASP [L].

IMNC: Instituto Mexicano de Normalización y Certificación A. C.

ISO: International Organization for Standardization (Organización Internacional para la Estandarización)

INDIKA: Sistema de administración de “indicadores de resultados” para la medición de los procesos de una organización.

Internet: Es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP [M].

JAD: Joint Application Design (Diseño de Aplicación Conjunta)

Javascript: Es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones [N].

MEG: Modelo de Equidad de Género

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario [P].

.NET: Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones [Q].

WWW: World Wide Web (*Red Global Mundial*)

WYSIWYG: What You See Is What You Get (Lo que ve es lo que obtiene)

1FN: Primera forma normal.

2FN: Segunda forma normal.

3FN: Tercera forma normal.