APLICACIÓN WEB PARA LA CONCEPTUALIZACIÓN DEL CUADRO DE MANDO INTEGRAL EN EL ARCHIVO DE BOGOTÁ

CRISTINA HERRERA Cod. 36031033

Trabajo de Investigación Dirigida presentado al Programa de Sistemas como requisito parcial para optar al título de
TECNOLÓGA EN SISTEMAS

CRISTIAN ANDRÉS RODRÍGUEZ MONTES
ASESOR

CORPORACIÓN UNIVERSITARIA UNITEC
FACULTAD DE SISTEMAS
BOGOTÁ, D.C.
2007

APLICACIÓN WEB PARA LA CONCEPTUALIZACIÓN DEL CUADRO DE MANDO INTEGRAL EN EL ARCHIVO DE BOGOTÁ

Todo el trabajo descrito en este documento es de la autoría de los abajo firmantes y fue realizado bajo la dirección del tutor asignado, excepto donde se han hecho referencias al trabajo de otros.

María Cristina Herrera Calderón

Certificado de aprobación:

Los abajo firmantes certificamos haber leído este Trabajo de Investigación Dirigida y que, en nuestra opinión, es totalmente adecuado, en calidad y nivel de profundidad, para optar al título de tecnólogo.

Cristian Andrés Rodríguez Montes
Coordinador Área de Software

José Eber Bonilla Olaya
Jefe de Programa

José Ignacio Duarte García
Jefe del Departamento de Promoción y Desarrollo Académico

Jamir Antonio Avila Mojica
Archivo de Bogotá
Contratista

A mis padres, Carmelina Calderón y Ricardo Herrera; y
A Jamir Antonio Avila Mojica

AGRADECIMIENTOS

Antes que nada, al Archivo de Bogotá, a su director Germán Rodrigo Mejía Pavonny, por la oportunidad y la colaboración brindada a lo largo de este proceso; a María Mercedes LadrondeGuevara amiga de siempre, por la confianza, los consejos, por ser el empujón que se necesita cuando se esta empezando, gracias porque sin su apoyo yo no habría conocido el Archivo de Bogotá y seguramente no hubiese sido tan fácil abrir las puertas del camino recorrido en estos dos últimos años; a Jamir Antonio Avila Mojica el mejor amigo, profesor, jefe, quizás el mejor profesional que he conocido en la vida, el mejor en muchas cosas más; gracias por el tiempo, la paciencia, los regaños, la incondicionalidad, la dedicación, el compromiso, la constancia y por todo su trabajo de enseñanza, gracias porque este proceso fue difícil, pero sin él lo hubiera sido mucho más, gracias porque siempre estuvo ahí; perdón, por las tantas veces que lo puse de mal genio y por todas las rabietas que le hice dar; mientras desarrolle y cuando culmine esta monografía pensé que no era suficiente decir "gracias", así que lo hice dueño de la lealtad más grande que se puede sentir y brindar a un amigo.
También, gracias a Gloria Margarita Rendón Cuartas, Subdirectora Técnica del Sistema de Archivos de la Administración Distrital del Archivo de Bogotá, quien me dio la oportunidad de poner en práctica lo que aprendí haciendo este trabajo, gracias por la confianza y por creer en mis capacidades para desarrollar Cuadro de Mando Integral en la subdirección que ella coordina.
Gracias a todos aquellos que de alguna forma desde su especialidad participaron en el desarrollo de este proyecto; gracias a: Gabriel Ibáñez, sus libros me fueron muy útiles; a Alejandro Sierra, quien resolvió las dudas con la mirada de un buen estudiante de Ingeniería de Sistemas, gracias por decir "es por tu bien", gracias por decir "sí" sin tener porque; a Erika Zamudio, por su amistad.

Gracias a la Corporación Universitaria UNITEC, al Jefe de Programa José Eber Bonilla Olaya y al Coordinador del Área de Software Cristian Andrés Rodríguez Montes de la Facultad de Sistemas, por aguantar lo dejado de un antes; gracias porque con ellos culmine satisfactoriamente este proceso.

De último en esta página, pero de primero en mi vida, ***GRACIAS*** a mis padres, por el esfuerzo, el amor y el apoyo en toda mi etapa educativa, personal y profesional; aunque nunca podría recompensar lo que han hecho y me han dado en la vida, espero que culminar esta fase los llene de alegría y orgullo. Gracias a Doris Patricia Noy Palacios por ser parte importante en mi vida.

CONTENIDO

Capítulo 3. APLICABILIDAD DEL MARCO TEÓRICO AL PROYECTO

ILUSTRACIONES

LISTA DE TABLAS

En el capítulo de aplicabilidad del marco teórico al proyecto de grado se utilizaron las siguientes tablas:

INTRODUCCIÓN

La competitividad del mundo moderno hace necesario implementar metodologías gerenciales que le permitan a la dirección de una organización conocer con veracidad el desempeño de la misma y poder tomar decisiones preventivas y correctivas; metodologías que garanticen el comportamiento estable de los factores e indicadores que intervienen en la funcionalidad de una entidad dentro de un rango previamente establecido, teniendo en cuenta los criterios de actuación de esta.

Aunque, no es suficiente tomar decisiones acertadas en el momento adecuado; también se hace necesario que la organización en su totalidad apunte al logro de unos objetivos y metas en común con el fin de alcanzar la visión y mantener la misión de la misma.

Apuntando en este sentido existe una metodología gerencial denominada *Balanced Scorecard (Cuadro de Mando Integral)* que traduce la visión y la misión de una organización en un conjunto de estrategias e indicadores enlazados mediante relaciones causa-efecto, agrupados en cuatro perspectivas: financiera, del cliente, procesos internos y aprendizaje y crecimiento.

El propósito de este Trabajo de Investigación Dirigida es brindar al Archivo de Bogotá una herramienta de control que le permita a las directivas, gestionar de manera eficaz la ejecución de los procesos y planes derivados de la misión de la entidad y la toma adecuada de decisiones; basándose en datos que brinden información constante, real y precisa, que identifique los diversos factores que se derivan del desarrollo funcional de la misma.

OBJETIVO GENERAL

Diseñar y desarrollar una aplicación, que permita conceptualizar la estructura gerencial y la definición formal de los indicadores de gestión del Archivo de Bogotá, con el fin de apoyar la toma adecuada de decisiones y hacer seguimiento al funcionamiento de la entidad.

OBJETIVOS ESPECÍFICOS

- Definir la estructura gerencial del Archivo de Bogotá
- Diseñar y desarrollar un módulo para el ingreso de los conceptos que definen la estructura gerencial del Archivo de Bogotá

- Permitir la definición de indicadores de eficacia, eficiencia y efectividad según las necesidades del Archivo de Bogotá, articulándolos en relaciones causa-efecto con la visión, misión, objetivos, metas, estrategias y perspectivas de la entidad
- Identificar las relaciones causa-efecto entre las características del Cuadro de Mando Integral

De igual manera, con el fin de aplicar los conocimientos adquiridos durante la carrera y entregar un producto a satisfacción del usuario, se tuvieron en cuenta aspectos para el desarrollo del software; como las metodologías de ingeniería de software. Sabiendo, que parte del trabajo de un desarrollador es entregar aplicaciones de alta calidad a unos costes establecidos y en un plazo determinado

Puesto que la importancia del software en los negocios aumenta, la eficacia de los desarrolladores es primordial. Por lo tanto, el activo más valioso como ingeniero es la capacidad para hacer coincidir sus compromisos con la calidad de los productos.

Así, el principal objetivo de las metodologías de desarrollo de software es proporcionar una guía al desarrollador para: Planificar, hacer, verificar y actuar de acuerdo a unos estimados de tiempo, coste y calidad.

Para el desarrollo del proyecto *APLICACIÓN WEB PARA LA CONCEPTUALIZACIÓN DEL CUADRO DE MANDO INTEGRAL EN EL ARCHIVO DE BOGOTÁ* se tomaron como base tres de las más importantes metodologías de desarrollo, que son:

- Personal Software Process (PSP)
- Rational Unified Process (RUP)
- Extreme Programming (XP)

Con el desarrollo de este proyecto a nivel profesional pretendí adquirir conocimiento sobre bases de datos relacionales, lenguajes de programación orientados a objetos, servidores de aplicaciones, construcción de interfaces gráficas para la Web, incursionar en el campo de las aplicaciones Web, así mismo el manejo de herramientas de gestión. Espero que sea para ustedes de gran utilidad, tal y como lo fue para mí.

CUADRO DE MANDO INTEGRAL

Antes de profundizar en el tema de Cuadro de Mando Integral (Balanced Scorecard BSC) es necesario saber que este es una herramienta de gestión para el control de la calidad y concretamente, orienta a una organización en la toma adecuada de decisiones y en el conocimiento real de su desempeño funcional.
De esta manera es importante conocer aspectos como: El Control de Gestión y los Indicadores de Gestión previo al Cuadro de Mando Integral.

CONTROL DE GESTIÓN

El control es garantizar el comportamiento estable de los factores e indicadores que intervienen en la funcionalidad de una organización dentro de un rango previamente establecido, teniendo en cuenta los criterios de actuación de la empresa.
Cabe resaltar que son los procesos diarios de la organización los encargados del monitoreo de sus factores e indicadores, y de éstos depende el éxito o el fracaso de los aspectos vitales de la empresa. Son las actividades cotidianas las que determinan como se hacen las cosas para el logro de un resultado positivo o negativo y en este caso realizar los ajustes necesarios cuando se presenten desviaciones frente a lo establecido o esperado.
Pero, cuando se habla de control no debemos enfocarnos sólo en acciones correctivas, también hay que observar aquellos aspectos que le permitan a la organización conocer las causas de los resultados y hacer previsiones para el futuro.

De esta manera se pueden definir dos tipos de control:

- Control de verificación: Se realiza implementando una acción correctiva, buscando que las actuaciones y resultados se ajusten a lo planeado; una vez logrado este propósito el control termina

- Control para el aprendizaje: Este control pone en conocimiento las causas que generan un resultado tanto positivo como negativo, mantiene un registro histórico de los resultados de la organización que le permita conocer la evolución de los factores claves, indicadores y procesos, y establecer la capacidad que tiene para mejorar su desempeño. Este control permite la constante confrontación de los criterios de actuación con lo hechos observados durante la ejecución, garantizándole a la organización aprender de su propia experiencia.

Ahora, la gestión se define como el conjunto de decisiones y acciones que llevan al logro de objetivos previamente establecidos. Dentro de este concepto es importante destacar aspectos como la estrategia y los indicadores de gestión; ya

que en una organización no basta con tomar decisiones acertadas sino que hay que contar con información adecuada que se constituye en el recurso clave que garantiza la calidad de las decisiones.

Esta información está contenida en los indicadores de gestión convirtiéndose en un instrumento de medición esencial en una organización que de no emplearlos le impide: Evaluar la capacidad de mejoramiento de los procesos, el conocimiento del estado y evolución de un factor clave a través de determinado período de tiempo, la comparación de los resultados obtenidos con los esperados y el entendimiento de las causas de una desviación, la relación satisfactoria con el entorno y la comparación con los resultados de otras empresas tomadas como referentes; de esta manera la organización obstaculiza la posibilidad de aprender de la experiencia.

Habiendo descompuesto estos dos términos (control y gestión) nos podemos referir al Control de Gestión como: Un sistema integro de evaluación que mediante un Cuadro de Mando Integral apoyado en estrategias, objetivos, indicadores, relaciones de causa -efecto e informes generados en forma sistemática, periódica y objetiva le permite a la organización tomar decisiones acertadas y oportunas, adoptar las medidas correctivas pertinentes y controlar la evolución en el tiempo de los factores claves, indicadores y procesos. El Control de Gestión es un sistema de mejoramiento continuo con miras al perfeccionamiento de los procesos organizacionales.

Funciones del control de gestión

El aporte que hace el control de gestión a la gerencia de una organización resulta de la ejecución de las siguientes funciones:

- Facilitar el aprendizaje organizacional, creando la memoria institucional
- Facilitar el diagnóstico permanente
- Mejorar la planeación y la programación
- Medir el perfeccionamiento
- Posibilitar la descentralización
- Evaluar el desempeño funcional de la entidad
- Mejorar la flexibilidad
- Definir niveles de exigencia

El control de gestión establece no sólo el grado en el que se alcanzan los objetivos y las metas organizacionales, sino también las causas que inciden en los resultados favorables o desfavorables de la entidad.

El control de gestión desde sus funciones permite a la organización aprender de la experiencia y planear procesos de mejoramiento, como factor de perfeccionamiento; ya que guardando datos históricos se visualiza la variación de

las variables y es posible analizar la tendencia de su comportamiento permitiendo realizar diagnósticos acertados de la situación real de la entidad mostrando los datos más significativos; de allí que el control de gestión sea considerado como un "sistema de alarmas tempranas" porque informa cada vez que se produce una situación excepcional y permite emprender acciones correctivas, de mejoramiento o preventivas contribuyendo a mejorar los tiempos de respuesta de la organización.

Dado que se esta hablando de un control integral de gestión, se debe considerar a la organización como un sistema que esta inmerso en un ambiente en el cual se encuentran elementos que si bien son externos a la entidad pueden afectar positiva o negativamente su desempeño y funcionalidad. Es así, como se deben identificar esos elementos y la relación directa o indirecta con la organización y establecer mecanismos para monitorear su comportamiento y de esa manera poder predecir su impacto y generar acciones de mitigación a posibles eventos negativos o riesgos.
Es importante tener en cuenta que las condiciones del ambiente son dinámicas, lo cual representa que la empresa debe estructurarse y adaptarse constantemente para responder con efectividad a los posibles cambios, a la vez que aprende de su propia experiencia y de la de otras entidades tomadas como referencia.

El control de gestión se apoya en el seguimiento y la medición de indicadores; este proceso proyecta el desempeño real, lo compara con un objetivo o meta y desencadena una acción correctiva en caso de ser necesario.

Indicadores de Gestión

Los indicadores de gestión son información exacta, integral y oportuna que refleja el logro de la visión, misión, objetivos, metas, factores clave de éxito, procesos y actividades de una organización. Para controlar, mejorar, comparar y justificar cualquier proceso y conocer su desempeño es necesario contar con un conjunto de guías que le sirvan a la entidad de apoyo para la gestión y el control de su funcionamiento; esas guías son los indicadores. Es muy difícil administrar lo que no se puede medir, pero algo fundamental que hay que tener en cuenta es que los indicadores son un medio y no un fin, es decir, no son la meta inmediata a alcanzar, son apoyo para el control de gestión y calidad en una organización.

Los indicadores permiten sustituir el análisis subjetivo de la gerencia de una organización, por una visualización objetiva y real del desempeño de los procesos. Esta información sirve a su vez para apreciar la evolución de las variables a través de períodos relativamente largos, para evaluar el mejoramiento continuo de los procesos y para hacer comparaciones con los resultados de otras empresas tomadas como referencia (benchmarking).

Como toda información los indicadores de gestión se componen de unos atributos que permiten el correcto planteamiento de los mismos:

- *Exactitud:* Guardan coherencia con los objetivos y metas a alcanzar
- *Deben tener una forma de representación, ya sea cuantitativa o cualitativa*
- *Periodicidad o frecuencia:* Cuán a menudo se requiere, produce o analiza
- *Origen:* Pueden ser indicadores internos o externos a la entidad
- *Los indicadores pueden ser históricos o en tiempo real*
- *Relevantes:* Sirven efectivamente para la toma de decisiones
- *Integrales:* Visualizan de manera integral una situación determinada
- *Oportunos:* La información debe esta disponible, actualizada y ser exacta y confiable cuando se la necesita
- *Excluyentes:* No debe haber indicadores redundantes, es decir, que midan lo mismo o no sean coherentes con lo que pretenden medir
- *Jerarquizados:* Pueden haber indicadores que se refieren a la organización como un todo e indicadores que controlan áreas, procesos, procedimientos, personas, etc. específicos

Especificación de los indicadores

Los indicadores se establecen partiendo de los siguientes aspectos:

1. *Composición* son las características en la definición textual del indicador

    - Nombre: Debe definir claramente su objetivo y utilidad; responde a la pregunta ¿qué se quiere medir?
    - Descripción o glosario: El indicador debe encontrarse documentado en términos de especificar de manera precisa las variables que se relacionan en su cálculo
    - Unidad de medida: Define la naturaleza de las variables que se relacionan en el cálculo del indicador, por ejemplo: pesos, metros, litros
    - Presentación: Forma en la que se expresa o presenta el valor de un indicador. Existen diversas maneras de expresar un indicador:

        - Razones: Expresan la relación entre dos datos con unidad de medida de la misma naturaleza. La relación se obtiene al dividir un dato (numerador), entre una base (denominador). Cuando las razones resultan de relacionar cantidades expresadas en unidades de medida diferentes, reciben el nombre de tasa
        - Porcentaje: Presentan el valor relativo de una cifra con respecto de un valor de referencia igual a cien
        - Promedio: Los términos más usuales para referirse al promedio son:

- Media aritmética: Se obtiene sumando un conjunto de valores y dividiendo el resultado por el número de valores sumados
- Mediana: La mediana de un conjunto de datos es el valor que ocupa la posición central cuando esos datos son ordenados del más bajo al más alto. Cuando el número de datos es par, la mediana es la media entre los datos centrales
- Moda: Es el valor que se presenta el mayor número de veces

➢ Números índices: Expresan los cambios relativos de una variable, comparada con una base a la cual se le asigna el valor de cien

❖ Descripción operacional: Identifica las variables y la manera como se relacionan para el cálculo del indicador
❖ Forma de cálculo: Fórmula matemática para establecer el valor resultado de un indicador

2. *Naturaleza o tipo*: En cuanto a la naturaleza los indicadores se clasifican según los factores claves de éxito ya que contar con un conjunto de indicadores que los abarque permite la adecuada toma de decisiones.

Los indicadores de acuerdo a los factores clave pueden ser de *eficacia, eficiencia, efectividad, productividad, calidad, economía y perfeccionamiento*

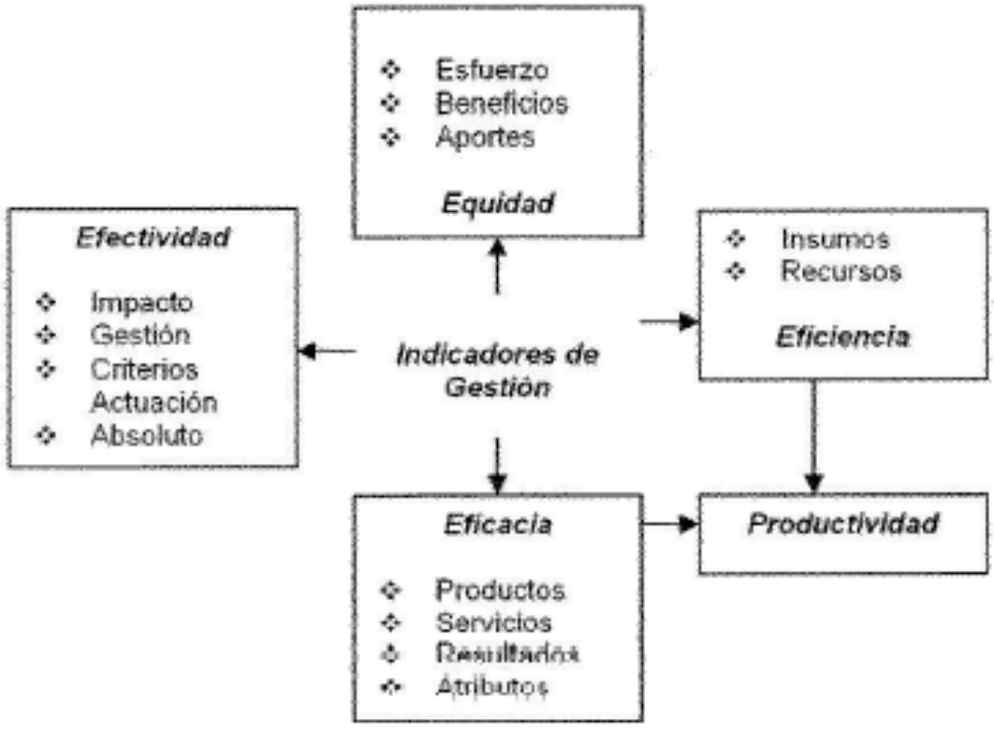


Figura No. 1 Factores clave e indicadores de gestión

Es un error ejercer control basándose únicamente en resultados, es decir, en la eficacia; se debe observar todas las dimensiones de la gestión integral.

3. *Vigencia:* Los indicadores pueden ser:

   - Temporales: Son indicadores asociados a proyectos cuya durabilidad es finita en la organización
   - Permanentes: Se asocian a los procesos de la organización, son indicadores que se establecen a largo plazo y que permiten medir el logro de los objetivos misionales y visionales de la entidad

4. *Nivel de generación:* Se refiere al nivel dentro de la organización donde se genera la información y medición del indicador; el nivel puede ser: estratégico, táctico u operativo

Para establecer indicadores es necesario conocer los resultados específicos que se esperan y sus características, lo que representaría el conjunto de factores clave para lograr la eficacia.
También es necesario tener presente los mecanismos de gestión actual, los procedimientos que permitirán el logro de los resultados y los recursos disponibles teniendo en cuenta factores óptimos de aprovechamiento, lo que representaría el conjunto de factores clave para lograr la eficiencia.

Establecer indicadores de gestión

Lo primero para establecer indicadores de gestión, es generar una medida para cada factor clave de éxito de la organización, entendiéndose por factor clave todo aquel aspecto que es necesario mantener bajo control para lograr el éxito de un proceso y el logro de las metas y objetivos organizacionales.

Luego, se determina para cada indicador: estado, umbral y rango de gestión

- *Estado:* Corresponde al valor actual del indicador

- *Umbral:* Corresponde al valor del indicador que se pretende lograr o mantener

- *Rango de gestión:* Consiste en establecer un rango de comportamiento para el indicador que permita hacerle seguimiento, este rango esta comprendido entre los valores máximo y mínimo que puede tomar el indicador: máximo, satisfactorio, aceptable, precaución y mínimo; sin olvidar la conveniencia del indicador: incremento o decremento

Indicadores de Eficacia

La eficacia se entiende como el logro de los objetivos y el cumplimiento en las especificaciones de los productos y servicios que genera la organización para la satisfacción de un cliente o usuario interno o externo a esta.
La eficacia está dada por el número de logros alcanzados, con respecto al número de logros esperados. La evaluación de la eficacia es posterior a: no se puede establecer si la hipótesis bajo las cuales se ejecuta una acción es acertada o no, hasta que no se conocen los resultados.

Indicadores de Eficiencia

La eficiencia se entiende como el uso racional de los recursos disponibles en el logro de los objetivos de la organización. Lo que se mide al ser eficientes es el manejo correcto del recurso empleado para la obtención de productos o resultados. La eficiencia está relacionada con aspectos internos de la empresa, ya que al cliente o usuario de la misma no le interesa cuanto le cuesta a la entidad poner a disposición un producto o servicio sino que el precio sea accesible, y la que la calidad satisfaga sus necesidades.
La eficiencia está dada por el número de productos o servicios generados por la cantidad de recurso utilizado, con respecto al número de productos o servicios esperados por la cantidad de recurso disponible.
Las variables que se miden en la eficiencia son: materia prima, talento humano, recurso tecnológico, logístico, metodológico y monetario.

Indicadores de Efectividad

La efectividad se entiende como el impacto que genera los resultados de la organización en sus clientes o usuarios, es decir, como el impacto de ser eficaces; pero, también se puede entender la efectividad como el cumplimiento de los criterios de actuación de la entidad, es decir, la efectividad es:"la capacidad para ejecutar las operaciones administrativas de manera que satisfagan los criterios de actuación previamente establecidos"[1]

La importancia de los indicadores de gestión recae en que son muchos los factores que afectan el desempeño de la organización, las áreas que la componen y las personas que las conforman. El comportamiento de estos factores es probabilístico y no determinístico, y la manera más efectiva de reducir la incertidumbre es contando con información administrable.

Tomar decisiones es elegir entre un conjunto de posibles opciones; la información es el recurso clave que garantiza la calidad de las decisiones, por tanto, el proceso

---

[1] PACHECO Juan Carlos, CASTAÑEDA Widberto y CAICEDO Carlos Hernán. Indicadores integrales de gestión. Bogotá: Mc Graw Hill, 2002. pg 58

de toma de decisiones debe reconocer la incertidumbre, la imperfección de los sistemas de evaluación y la complejidad interna y externa que afecta a la organización.

CUADRO DE MANDO INTEGRAL - BALANCED SCORECARD

Antes de la introducción del Cuadro de Mando Integral (CMI) a principios de la década de los noventa, en el campo empresarial ya existían herramientas de control que le permitían a una organización la adecuada toma de decisiones.

El *Tableau de Bord* que durante los años sesenta se convirtió en una de las herramientas más utilizadas en empresas, especialmente francesas es un antecedente del CMI, que incorporaba en un documento diversos indicadores para el control financiero de una empresa. Con el paso del tiempo esta herramienta ha evolucionado combinando no solo indicadores financieros, sino también indicadores no financieros que permiten controlar los diferentes procesos de la organización.

Durante esta misma década en Estados Unidos, General Electric desarrolló un tablero de control para hacer el seguimiento de los procesos de la empresa, a partir de ocho áreas clave de resultados, que incluían temas como rentabilidad, cuota de mercado, formación o responsabilidad pública; General Electric definía indicadores para hacer el seguimiento y controlar el logro de objetivos tanto a corto como a largo plazo.

En el año 1907 la empresa Du Pont utilizó un modelo con el fin de proporcionar una imagen de la situación económica de la misma; siendo este un modelo limitado porque sólo incluía datos de carácter financiero y se encontraba estrictamente orientado a las utilidades: era una presentación gráfica de los datos del estado de pérdidas y ganancias, teniendo el beneficio a corto plazo como principal criterio de evaluación.

De igual manera el concepto de Cuadro de Mando Integral ha sufrido grandes modificaciones desde su primera formulación en 1992, cuando se definía como: "Un conjunto de indicadores que proporcionan a la alta dirección una visión comprensiva de la empresa", para ser "Una herramienta de gestión que traduce la visión y la misión de una organización en un conjunto de estrategias e indicadores conectados entre sí mediante relaciones causa-efecto, agrupados en cuatro perspectivas: financiera, cliente, procesos internos y aprendizaje y crecimiento"

Se podría señalar, que el CMI actual recoge aspectos que ya existían alrededor del concepto de tablero de control; y aunque la idea siempre giraba en torno a seleccionar un conjunto de indicadores que pudieran ser construidos para apoyar la gestión, en los antecesores del Cuadro de Mando Integral esta selección correspondía a cada directivo quien escogía los que considerara más convenientes según su propia intuición y experiencia.

Es aquí donde se refleja la principal característica y aporte del CMI; este ofrece un método más estructurado de selección de indicadores y esto le concede más versatilidad, permitiendo una gestión proactiva de la empresa; y se escogen porque son relevantes en algún punto para la implantación de la estrategia o el seguimiento de la misma.

En el CMI es indispensable primero definir y entender la visión y misión organizacional, es decir, el modelo de negocio, que reflejará las interrelaciones entre los diferentes componentes de la empresa. Una vez construido, la organización utiliza este modelo para definir los objetivos y metas a lograr; esta es la base para diseñar las estrategias e indicadores, representados en un mapa de relaciones causa-efecto.
El CMI es una herramienta que permite establecer las brechas entre los objetivos, las metas y la realidad ejecutada, para impulsar los cambios en la estrategia y en éstos mismos de ser necesario.

Así pues, un Cuadro de Mando Integral adecuadamente construido debe contar la historia de la estrategia de la unidad de negocio. Debe identificar y ahcer que sea explícita la secuencia de hipótesis respecto a las relaciones de causa-efecto, entre las medidas de los resultados y los inductores de la actuación de esos resultados. Cada una de las medidas seleccionadas para un CMI debe ser un elemento en una cadena de relaciones causa-efecto, que comunique el significado de la estrategia a toda la organización.[2]

El énfasis sobre la construcción de relaciones de causa-efecto en el Balanced Scorecard permite visualizar a la entidad como un sistema dinámico. Permite que todos los que participan en el funcionamiento de la organización comprendan la forma en la que su desempeño afecta el de los demás.

En 1992, Kaplan y Norton de Harvard University presentaron al mundo empresarial el ***Balanced Scorecard*** que nace como una herramienta para relacionar de manera definitiva la estrategia y su ejecución dentro de una empresa, empleando indicadores y objetivos en torno a cuatro perspectivas.

Lo que pretende Balanced Scorecard es, que toda la unidad de negocio este encaminada hacia la estrategia; es relacionar el direccionamiento estratégico con los planes y estos con el desempeño diario de la organización; permitiendo generar soluciones integrales desde el nivel operativo, con el fin de lograr los objetivos a corto, mediano y largo plazo.

---

[2]Robert Kaplan, David Norton, *Cuadro de Mando Integral*. Barcelona: Gestión 2000, 2004, p. 44.

Los beneficios de implementar el BSC se pueden integrar en cuatro conceptos:

- Relacionar la estrategia con su ejecución definiendo objetivos en el corto, mediano y largo plazo.
- Tener una herramienta de control que permita la toma de decisiones de manera ágil.
- Comunicar la estrategia a todos los niveles de la organización consiguiendo así alinear a las personas con la estrategia.
- Tener una clara visión de las relaciones causa-efecto de la estrategia.

El Cuadro de Mando Integral contempla la actuación de la empresa desde cuatro perspectivas complementarias que se integran en la visión y la estrategia de la organización: la financiera, la del cliente, la del proceso interno y la de formación y crecimiento.

Uno de los aportes que ha convertido al CMI en una de las herramientas de control de gestión más utilizada en el mundo es que se basa en un modelo de negocio el cual a su vez se apoya en las relaciones causa-efecto; de donde se concluye que para dirigir de forma proactiva hay que actuar sobre las causas y no sobre las consecuencias.

Para tener un buen CMI, el modelo de negocio es crítico. Cada empresa tiene su propio modelo, que depende de su sector y de su estrategia.

A continuación, y con los cimientos de un buen modelo, los indicadores del CMI facilitan los puntos de referencia que se necesitan para evaluar el progreso en el desarrollo de la estrategia. Sin los datos que facilitan los indicadores, puede ocurrir que estrategias bien formuladas fracasen por falta de información actualizada acerca del proceso de implantación.

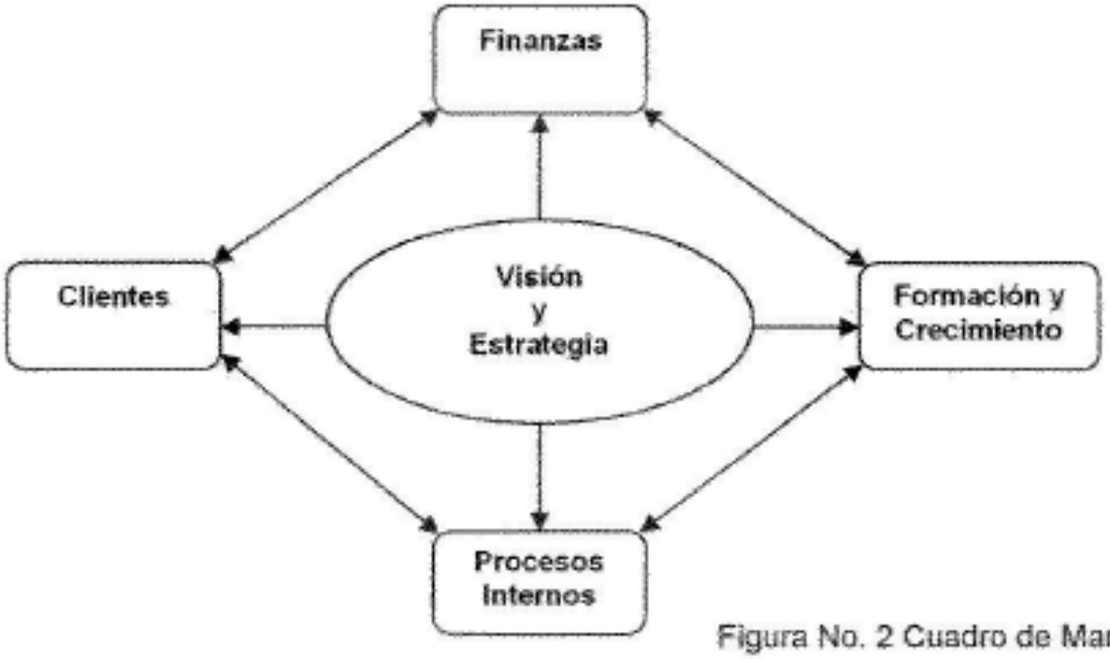


Figura No. 2 Cuadro de Mando Integral

Perspectivas del Cuadro de Mando Integral

- Perspectiva financiera: Incorpora la visión de los accionistas y mide la creación de valor de la empresa. Responde a la pregunta: ¿Qué indicadores tienen que ir bien para que los esfuerzos de la empresa realmente se transformen en valor?

- Perspectiva del cliente: Refleja el posicionamiento de la empresa en el mercado, es decir, en los segmentos de mercado donde quiere competir.

- Perspectiva interna: Recoge indicadores de procesos internos que son críticos para el posicionamiento en el mercado y para llevar la estrategia a buen fin.
  Los objetivos y los indicadores basados en este enfoque permiten a los directivos saber como está funcionando la entidad, y si sus productos o servicios están cumpliendo con los requerimientos del cliente.

  Los indicadores de esta perspectiva deben ser representativos del proceso a medir. Ejemplos de indicadores asociados a procesos: Tiempo de ciclo del proceso (Cycle Time), costo unitario por actividad, niveles de producción, costos de falla, costos de trabajo, desperdicio (costos de calidad), beneficios derivados del mejoramiento continuo, reingeniería, eficiencia en uso de los activos.

- Perspectiva de formación y crecimiento: Es la perspectiva donde más tiene que ponerse atención, sobre todo si piensan obtenerse resultados constantes a largo plazo. Aquí se identifican la infraestructura necesaria para crear valor a largo plazo. Hay que lograr formación y crecimiento en 3 áreas: personas, sistemas y ambiente organizacional. Normalmente son intangibles, pues son indicadores relacionados con capacitación a personas, software o desarrollos, máquinas e instalaciones, tecnología y todo lo que hay que potenciar para alcanzar los objetivos de las perspectivas anteriores.

¿Cómo se usa el Cuadro de Mando Integral?

Todo lo que pasa en cualquier empresa es un conjunto de hipótesis sobre la causa y efecto entre indicadores. Cualquier acción que se ejecute, tendrá un impacto directo sobre otra variable, es por eso que la perspectiva de Formación y Crecimiento es la base que permite crear la infraestructura necesaria para crecer en las otras perspectivas. Lo importante es saber que ninguna perspectiva funciona en forma independiente, sino que puede iniciarse una acción con alguna de ellas y repercutirá sobre todas las demás.

- Definición de Visión y Estrategias: Llegar a consensos sobre las estrategias lleva a establecer tanto objetivos como indicadores que los midan.

- Definición de Indicadores: Entendida la visión y estrategias de la empresa es posible determinar los objetivos que hay que cumplir para lograr la estrategia y representarlos en indicadores. Es importante que los indicadores no controlen la actividad pasada solamente, los indicadores deben reflejar los resultados concretos de los objetivos, pero también deberán informar sobre el avance para alcanzar esos objetivos. La combinación de indicadores de resultados e indicadores de actuación es lo que permitirá comunicar la forma de conseguir los resultados y, al mismo tiempo, el camino para lograrlo.

"Resultados son los indicadores históricos, indicadores de la actuación son indicadores previsionales." [3]

Los indicadores deben estar conectados a los objetivos y metas no sólo de las áreas ni de las personas, sino de la organización en su totalidad, deben permitir lograr los objetivos estratégicos y los de corto plazo (el direccionamiento estratégico y el día a día); es así, como los indicadores son un instrumento para el control y el despliegue de las estrategias que le permitan alcanzar la misión y la visión a la organización.

Otro aspecto que hay que resaltar es el número de indicadores; según Kaplan y Norton, un número adecuado es de 7 indicadores por perspectiva y si son menos, mejor. Se parte de la idea de que un tablero con más de 28 indicadores es difícil de evaluar totalmente, además de que se pueden dispersar los esfuerzos intentando perseguir demasiados objetivos al mismo tiempo. Puede ser recomendable durante el diseño empezar con una lista más extensa de indicadores. Pero es necesario un proceso de síntesis ya que los seleccionados serán los que se consulten frecuentemente y los que indiquen verdaderamente el estado actual de la empresa.

De igual manera es importante a la hora de seleccionar los indicadores, que en la medida de lo posible, sean cuantificables y objetivos.

Implementación del Cuadro de Mando Integral

Una vez definido el modelo de negocio y los indicadores de acción y resultados, es posible implementar el CMI de dos formas:

- Modelo de control y seguimiento (Sistema de control por excepción)
- Modelo de aprendizaje organizativo y comunicación

---

[3] Robert Kaplan, David Norton, *Cuadro de Mando Integral*. Barcelona: Gestión 2000, 2004.

Figura No. 3 Implementación del Cuadro de Mando Integral

En el primer caso el CMI tendrá unos objetivos para cada indicador y existirá un seguimiento de las medidas reales frente a los objetivos preestablecidos. Cuando haya una discrepancia importante entre la realidad y el pre-supuesto, se analizará el porqué de la diferencia. La atención se centra en procesos que sólo requieren tiempo en casos excepcionales.

En el segundo caso los resultados que recogen los indicadores sirven para evaluar si hay que cambiar el modelo de negocio o incluso la estrategia. La comparación entre lo que se esperaba y lo que ocurre realmente es una fuente de información útil para ajustar la forma de competir de la empresa. En este caso, el CMI no sirve para centrar atención directiva en procesos específicos; al contrario, sirve para enfocarla en aprender sobre la evolución del entorno y de la empresa.

Feedback y aprendizaje estratégico

El CMI no termina con el análisis de los indicadores. Es un proceso permanente en el que puede haber feedback de un bucle, que consiste en corregir las desviaciones para alcanzar los objetivos fijos definidos y feedback de doble bucle, donde las directivas reflexionan sobre la vigencia y actualidad de la teoría planteada en un inicio, y su posible adecuación. El feedback sugiere aprendizaje estratégico, que es la capacidad de adaptación de la organización frente a la desviación de los objetivos y metas establecidas, es el poder *"aprender a utilizar el Balanced Scorecard como un sistema de gestión estratégica."*

El feedback permite comparar la situación presente con la pasada y anticiparse posiblemente a una futura; indaga en las causas de los riesgos organizacionales

para evitar que vuelvan a presentarse, no se limita a introducir acciones correctivas.

El feedback tiene como fin, el proporcionar información precisa y real del desempeño de la organización, para definir los objetivos y establecer las estrategias, corregir su actuación y lograr la eficacia, eficiencia y efectividad.

Figura No. 4 Feedback

Dentro de las principales características que tiene el Cuadro de Mando Integral, se pueden listar las siguientes:

- Es una herramienta gerencial que se constituye como un apoyo eficaz para la adecuada toma de decisiones, con el fin de ejercer un buen control de calidad
- No sólo se concentra en los resultados, sino en la manera como éstos se obtuvieron, es decir, apunta tanto a la causa como al efecto o resultado
- Proyecta el futuro de la organización
- Es integrador; alinea todas las áreas de la organización hacia el logro de la misión y visión articulándolas en relaciones causa-efecto en el desarrollo de los objetivos, metas y estrategias, monitoreando con indicadores de causa y efecto
- Maneja los resultados como indicadores históricos y las causas como indicadores previsionales
- No sólo cuenta con indicadores financieros, sino también con indicadores para los intangibles organizacionales

- Permite mejorar la planificación futura de la organización
- Permite la mejora continua en la organización

BALANCED SCORECARD COLLABORATIVE - REQUERIMIENTOS FUNCIONALES DE LAS APLICACIONES BSC[4]

BSCol facilita el conocimiento mundial, el uso, el aumento e integridad del balanced scorecard como valor agregado en el manejo de procesos. Collaborative ofrece educación, recursos y desarrollos, servicios diseñados como parte de las mejores prácticas para lograr los mejores resultados. Fundado y manejado por los creadores de Cuadro de Mando Integral - Kaplan y Norton - provee a las organizaciones con un centro a nivel mundial de excelencia y experiencia en el manejo de esta herramienta gerencial de calidad.

Requerimientos funcionales para las aplicaciones Scorecard

Las aplicaciones deben facilitar el manejo de la organización a través de la metodología BSC. Los estándares representan las funcionalidades mínimas para los sistemas scorecard.
BSCol desarrolló estándares funcionales de Balanced Scorecard en 1999, para preservar la integridad del software de gestión en el mercado y proporcionar los medios por los cuales se certifican los desarrollos realizados, con base en la medida de funcionamiento.

Hoy, BSCol es la fuente de conocimiento para las últimas tecnologías Balanced Scorecard; tanto para los desarrolladores como para el gerente que busca seleccionar e implementar el mejor software.

Las aplicaciones que adoptan los estándares Kaplan y Norton obtienen la marca certificada de Collaborative, lo que se convierte en una ventaja para el usuario final ya que le permite la gerencia equilibrada de la entidad mediante Scorecard.

Los estándares están divididos en cuatro secciones:

- Diseño: La aplicación debe ser flexible a los elementos básicos de un Cuadro de Mando Integral, es decir, debe permitir:
  - Visualizar la estrategia de las cuatro perspectivas: financiera, cliente, procesos internos, aprendizaje y crecimiento.
  - Identificar objetivos estratégicos para cada perspectiva
  - Asociar indicadores para los objetivos estratégicos
  - Enlazar los objetivos estratégicos en relaciones causa-efecto
  - Asignar metas (umbral) a los indicadores

[4] Traducción Libre de: Balanced Scorecard Functional Standards ™ Release 1.0a

- Crear iniciativas o planes de acción estratégicos

- Estrategia de educación y comunicación: Una de las razones principales para implementar un software de Cuadro de Mando Integral es que permita la fácil educación y comunicación sobre las estrategias de la organización. Por lo tanto, una aplicación certificada debe habilitar al usuario para documentar y comunicar descripciones de: objetivos, indicadores y planes de acción alineados con la estrategia

- Ejecución de los procesos: La aplicación debe permitir alinear los procesos de la entidad a la estructura estratégica de la misma

- Feedback y aprendizaje: Los tiempos de los ciclos de retroalimentación pueden reducirse significativamente. El análisis de los resultados de los indicadores contra los resultados esperados (umbrales) permitiría a los gerentes entender cuales áreas de la organización requieren atención en el futuro. Sin embargo, el sistema no debe anular el juicio del gerente; el software debe relacionar ambos juicios tanto el objetivo como el subjetivo, mediante la representación gráfica de los indicadores y un reporte en tiempo real específico de un objetivo, una estrategia, un indicador y sus tendencias.

Diseño Balanced Scorecard

1. Perspectivas: Una perspectiva es un componente, en el cual se descompone la estrategia organizacional para el manejo de la implementación del Cuadro de Mando Integral. Existen cuatro perspectivas, pero estas no son camisa de fuerza; otras pueden ser agregadas o se puede reemplazar según las necesidades de la organización.
   Las aplicaciones certificadas deben tener por lo menos las cuatro perspectivas y habilitar el renombre y la creación de nuevas según las opciones del usuario.

2. Objetivos: Un objetivo es la razón por la cual una estrategia es operacional, acertada y funcional. Generalmente, los objetivos forman la base para la estrategia de la organización. Las aplicaciones certificadas permitirán alinear los objetivos estratégicos a cada una de las perspectivas.

3. Indicadores: Un indicador refleja la ejecución y progreso de un objetivo; deben ser cuantificables. Los indicadores comunican el comportamiento específico que requiere lograr el objetivo y llega a ser una declaración de cómo alcanzarlo. Las aplicaciones certificadas deben tener un número razonable de indicadores e identificar las relaciones causa-efecto entre estos y los objetivos a los que miden.

4. Umbrales: El umbral es la meta cuantificable de un indicador. Las aplicaciones certificadas tendrán umbrales con un especifico timeframe (periodicidad).

5. Relaciones causa-efecto: Los objetivos, estrategias e indicadores están enlazados entre sí mediante relaciones causa-efecto. Estas relaciones son similares a la expresión "si-entonces" y deben ser explicitas en el Cuadro de Mando Integral. Las aplicaciones certificadas permitirán generar mapas estratégicos, que son representaciones gráficas de las relaciones causa-efecto entre los elementos de un BSC; estos mapas deben ser fáciles de editar para el usuario.

6. Iniciativas estratégicas: Son los planes de acción que gestionan la ejecución de la estrategia organizacional. Son el grupo de actividades que aseguraran la realización de las estrategias y el logro de los resultados a los que éstas apuntan.

Educación y comunicación de la estrategia

Las aplicaciones certificadas deben tener las siguientes funcionalidades:

1. Descripción de los elementos básicos del Cuadro de Mando Integral: Las aplicaciones certificadas deberán generar la documentación de la descripción cualitativa de cada uno de los elementos del Cuadro de Mando Integral

    - Fórmulas
    - Unidades de medida
    - Frecuencia de reportes
    - Responsables
    - Orígenes de datos
    - Fechas de efectividad
    - Datos históricos
    - Líneas de tiempo
    - Recursos

Dentro de la descripción de los elementos del BSC es necesario que las aplicaciones certificadas permitan visualizar la información gerencial de la organización de manera gráfica y entendible para el usuario.

A continuación la lista de los desarrolladores y proveedores de software certificados por BSCol:

ActiveStrategy Inc.	CorVu	Microsoft	Prodacapo
Bitam	Extensity	Oracle	QPR
Business Objects	Hyperion	PeopleSoft	Rocket Software
Cognos	Information Builders	Performancesoft	SAP
Consist FlesSI	InPhase	Pilot Software	SAS
Corporater	Intalev	Procos	Vision Grupo Consultores

Tabla No. 1 Software certificado por BSCol

Este es el diseño básico de un Scorecard:

Perspective	Cause & Effect Linkage	Objectives	Measures	Targets	Initiatives
Financial	Profitability; Revenue Growth	• Profitable Business Growth	• Operating Income • Sales vs. Last Yr	• 20% increase • 12% increase	• Likes Program
Customer	Product Quality → Shopping Experience	• Quality Product from a knowledge-able Associate	• Return Rate • Customer Loyalty – Ever Active % – # units	• Reduce by 50% each yr • 80% • 2.4 units	• Quality management program • Customer loyalty program
Internal Process	"A" Class Factories ↔ Line Plan Management	• Improve factory quality	• % of Merchandise from 'A' factories • Items in-Stock vs. Plan	• 70% by year 3 • 85%	• Corporate Factory Development Program
Learning & Growth	Factory Relationship Skills; Merchandise Buying / Planning Skills	• Train & equip the workforce	• % of Strategic Skills Available	• yr 1 50% yr 3 75% yr 5 90%	• Strategic Skills Plan • Merchants Desktop

Figura No. 5 Diseño Básico de un Scorecard

METODOLOGÍAS DE DESARROLLO

- **PROCESO SOFTWARE PERSONAL (PSP)**

- **DISEÑO ORIENTADO A OBJETOS**
  - Modelos de Diseño - UML
  - Casos de Uso
  - Requerimientos Funcionales y No Funcionales
  - El Proceso Unificado de Rational (RUP)

- **PROCESO DE DISEÑO DE LA INTERFAZ DE USUARIO**

- **METODOLOGÍAS ÁGILES**
  - Programación Extrema (XP)

PROCESO SOFTWARE PERSONAL (PSP)

El Proceso Software Personal (PSP) fue diseñado para ayudar a los desarrolladores a hacer bien su trabajo. Muestra como aplicar métodos avanzados de ingeniería a sus tareas diarias. Proporciona métodos detallados de planificación y estimación, muestra a los ingenieros como controlar su rendimiento frente a estos planes y explica como los procesos definidos guían su trabajo.

Características más relevantes del Proceso de Software Personal:

Cuando un proceso está totalmente descrito, se denomina *proceso definido*. Los procesos definidos están compuestos normalmente de guiones, tablas, plantillas y estándares. Un guión del proceso es un conjunto de pasos escritos, que los usuarios o agentes del mismo siguen cuando lo utilizan. Distintas tablas, tales como registros y resúmenes, se emplean para registrar y almacenar los datos del proyecto.

- Planificar: Elaborar una descripción de las funciones del programa y el tiempo estimado para el desarrollo de cada una
- Diseñar: Estructurar la lógica del programa antes de comenzar a escribir el código; levantar el diseño en un diagrama de flujo, seudocódigo o en cualquier formato especificado
- Codificar: Implementar el diseño codificándolo en el lenguaje de programación seleccionado
- Compilar: Compilar el programa y corregir todos los defectos que se encuentren. Continúe compilando y corrigiendo los defectos hasta que compile el programa sin ningún mensaje de error
- Pruebas: Hacer pruebas para asegurarse que los programas cumplen todos los requisitos y superan un conjunto adecuado de pruebas sin error

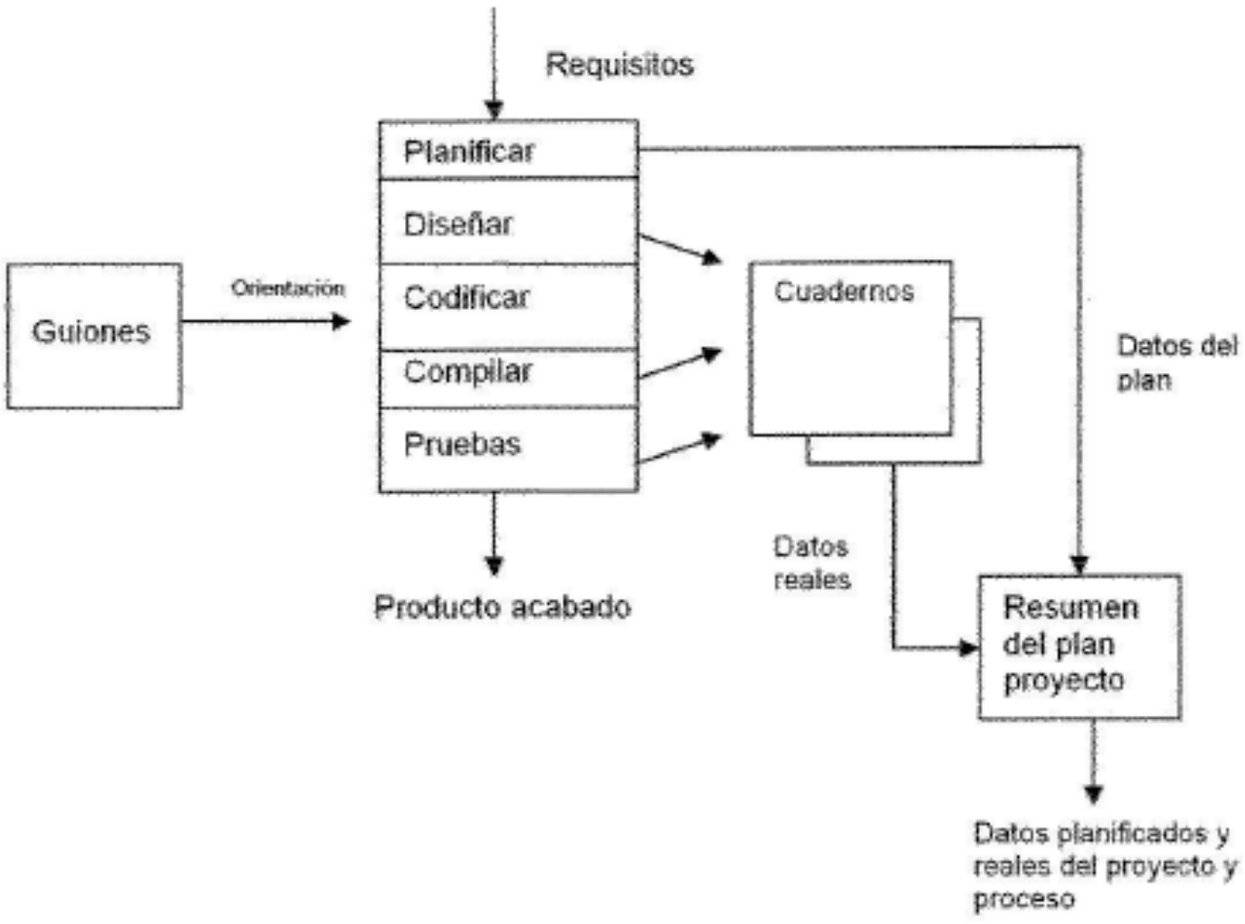


Figura No.6 Flujo del proceso del PSP

La disciplina del trabajo de alta calidad

La disciplina se define como una actividad o ejercicio que desarrolla o mejora habilidades. PSP proporciona un marco de trabajo estructurado para desarrollar las habilidades personales y los métodos que se necesitan como desarrollador de software, sin olvidar que el poco entrenamiento en esta metodología implica costos, consume tiempo y aumenta el riesgo durante el trabajo de desarrollo.

La importancia del trabajo de alta calidad

Los sistemas informáticos modernos pueden ejecutar millones de instrucciones por segundo. Así, un defecto poco probable que pueda suceder solamente una vez de entre un billón puede ocurrir varias veces en un día.

Con el software, las condiciones inusuales se presentan todas las veces, condiciones que parecen imposibles ocurren en poco tiempo. Los defectos en los elementos más pequeños de un gran sistema pueden causar serios problemas de forma impredecible y ocasional. Si se comete un error trivial que deja un defecto en el producto, el resultado puede causar un gran inconveniente o consecuencias para el usuario.

¿Cómo mejorar la calidad del trabajo de desarrollo?
Al igual que en un proceso administrativo, cualquier actividad que se ejecute se debe someter al *mejoramiento continuo*, de ahí la necesidad de contar con indicadores, estrategias y objetivos para establecer, si es necesario, cambios. El proceso de mejora es difícil porque las personas son reacias a intentar cosas nuevas. Sus hábitos parecen tan naturales que pueden pensar que el cambio no les ayudará.

Una vez se han definido las medidas, se deben reunir y analizar los datos. Si es necesario mejorar, a continuación se analiza el proceso para ver dónde se tiene que hacer cambios. Finalmente, para mejorar, debe cambiar lo que se hace normalmente.

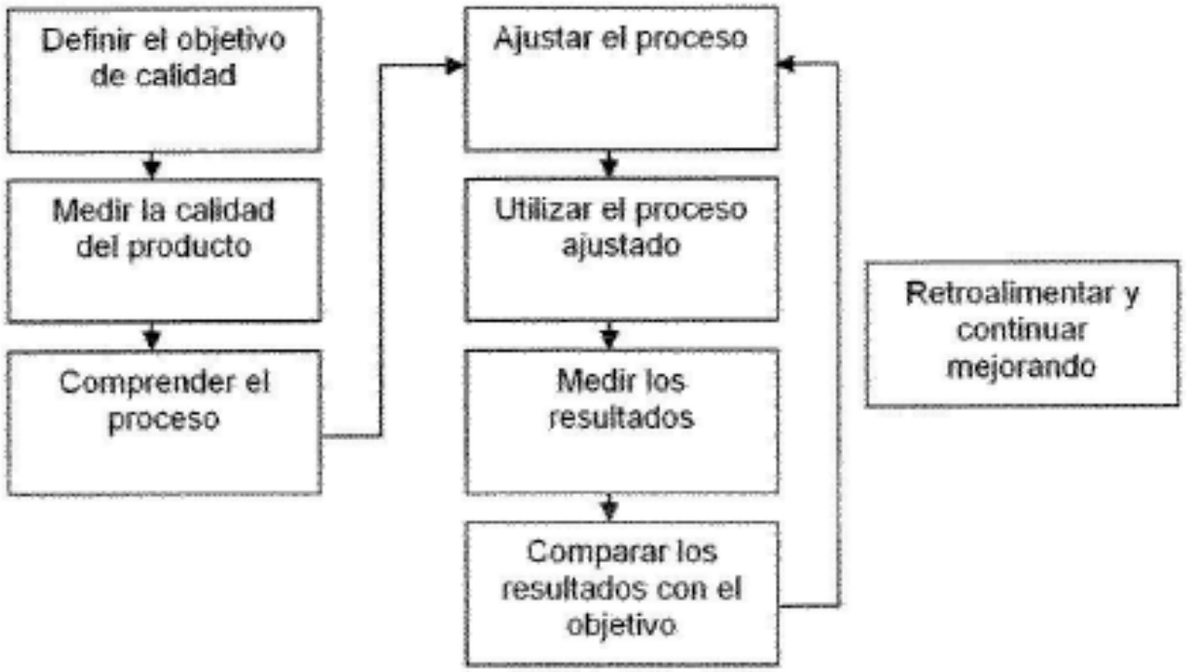


Figura No.7 Proceso de mejoramiento

Proceso de planificación

El primer paso para entender el proceso es identificar lo que se hace dentro de este. Aquí se definen las principales actividades, se estima la frecuencia en que se ejecutan y el tiempo dedicado a cada una; en este primer paso no se requiere que se midan esos tiempos, una estimación es suficiente. Esta información se escribe en el siguiente formato[5]:

[5] Ver Capítulo Aplicabilidad en el Proyecto

Desarrollador: _____ *Fecha:* _____

ACTIVIDAD	FRECUENCIA	TIEMPO

Tabla No.2 Planeación de Actividades[6]

La gestión del tiempo

Los fundamentos para gestionar el tiempo son:

- Probablemente harás esta semana lo mismo que hiciste la semana pasada
- Para hacer un plan realista, tienes que controlar la forma de gastar tu tiempo
- Para comprobar la exactitud de tus estimaciones de tiempo y planes, debes documentarlas y posteriormente compararlas con la que realmente haces
- Para hacer más precisos tus planes, determina las equivocaciones de los planes anteriores, y qué podrías haber hecho para mejorar
- Para gestionar el tiempo, planifica tu tiempo y sigue el plan

La planificación es una habilidad que pocas personas han aprendido. Hay, sin embargo, métodos conocidos que se pueden aprender y practicar. El primer paso para aprender a hacer buenos planes, es hacer planes. Así que, escriba su plan para que posteriormente tenga algo con lo que pueda comparar sus datos actuales. Una vez hecho esto registre el tiempo que utiliza; cuando tenga documentado el plan compare los resultados reales con el plan original.
La clave para planificar con exactitud es hacer planes consistentes y compararlos con los resultados posteriores. Luego, siga el plan.
Para practicar la gestión del tiempo, el primer paso es entender cómo utiliza el tiempo actualmente. Esto se hace en varios pasos:

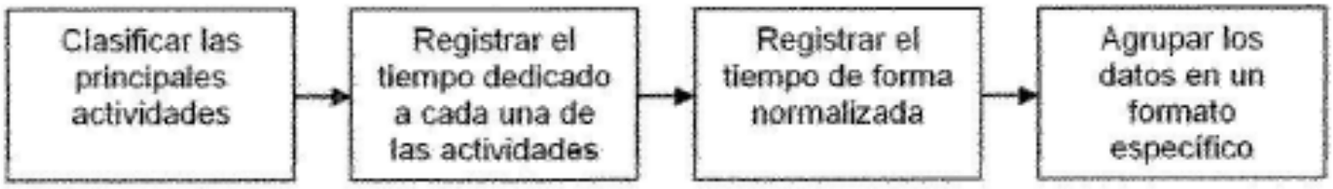


Figura No.8 Gestión del tiempo

El control del tiempo

Cuando se lleva un registro del tiempo, el objetivo es obtener datos de cómo se trabaja realmente. La forma y el procedimiento utilizado para reunir los datos no es

[6] Tomado de Introducción al Proceso Software Personal PSP

La tabla utilizada para registrar el tiempo es la siguiente[7]:

Desarrollador: _____ *Fecha:* _____

FECHA	COMIENZO	FIN	TIEMPO DE INTERRUPCIÓN	ΔTIEMPO	ACTIVIDAD	COMENTARIOS	C	U

Tabla No.3 Registro de tiempos

- Fecha: La fecha de realización de la actividad
- Comienzo: La hora de comienzo de la actividad
- Fin: La hora de finalización de la actividad
- Interrupción: Cualquier pérdida de tiempo debida a interrupciones. Los datos de tiempo registrados pueden utilizarse para comprender con qué frecuencia se interrumpe nuestro trabajo
- ΔTiempo: El tiempo dedicado a cada actividad, entre los tiempos de comienzo y fin menos el tiempo de interrupción
- Actividad: Nombre descriptivo para la actividad
- Comentarios: Una descripción más completa de lo que está haciendo, el tipo de interrupción o cualquier cosa que podría ser útil cuando posteriormente se analicen los datos de tiempo
- C (Completado): Rellenamos esta columna cuando terminamos una actividad
- U (Unidades de trabajo): El número de unidades de una actividad terminada

Planificación de períodos y productos

Planes de períodos y productos

Hay dos clases de planificación. La primera está basada en un período de tiempo. Puede ser cualquier intervalo del calendario: un día, semana, mes o año. Un plan de período hace referencia a la forma de planificar la utilización del tiempo durante ese período.

La segunda clase de plan está basada en la actividad, como desarrollar un programa o escribir un informe. Los productos pueden ser tangibles como los programas o informes, o intangibles como el conocimiento adquirido o el un servicio que se proporciona.

---

[7] Ver Capitulo Aplicabilidad en el Proyecto
Tomado de Introducción al Proceso Software Personal PSP

Aunque los planes del período y del producto deben estar relacionados, son diferentes. El trabajo de planificación, los ingresos y otras actividades diarias están controlados por períodos de tiempo. Los gastos e ingresos están gobernados por planificaciones semanales, mensuales y anuales. De esta manera vivimos en un mundo con períodos de actividades, reglas y limitaciones. El principal propósito del trabajo de desarrollo de software es producir productos y servicios de valor para otros. El coste, la planificación y la calidad de esos bienes y servicios son lo más importante. No podemos hacer un plan competente de uno de ellos sin planificar también el otro.

Resumen semanal de actividades

La Tabla 4 muestra los formatos de tiempo que son más adecuados para la planificación del período[8]

Desarrollador: _____ *Fecha:* _____

Tarea									**Total**
Fecha									
Dom									
Lun									
Mar									
Mier									
Jue									
Vie									
Sab									
Totales									

Tiempos y medias del período *Número de semanas (número anterior + 1):*

Resumen de las semanas anteriores

Totales									
Media									
Máximo									
Mínimo									

Resumen incluyendo la última semana

Totales									
Media									
Máximo									
Mínimo									

Tabla No.4 Resumen semanal de actividades

[8] Ver Capitulo Aplicabilidad en el Proyecto
Tomado de Introducción al Proceso Software Personal PSP

- Tareas
- Fecha: La fecha de realización de la tarea
- Para cada día, se suman todos lo minutos dedicados a cada tarea basándose en el registro de tiempos
- Después de completar todas las casillas para cada día, se obtiene el total para cada día
- Luego, se calcula los totales semanales para cada tarea y,
- se comprueban los totales

Cálculo de los tiempos y medias del período

- Anotar el total de semanas transcurridas
- Resumen de las semanas anteriores: Valores mínimos y máximos del número de semanas transcurridas menos 1
- Sumar los tiempos dedicados a cada tarea incluyendo la semana actual
- Calcular el tiempo medio dedicado semanalmente a cada tarea durante el período: Total de tiempo / número de semanas
- Calcular el tiempo máximo dedicado a una tarea durante una semana: Comparar los valores máximo del resumen de las semanas anteriores con los valores totales de la semana actual y tomar el valor mayor
- Calcular el tiempo mínimo dedicado a una tarea durante una semana: Comparar los valores mínimo del resumen de las semanas anteriores con los valores totales de la semana actual y tomar el valor menor

La planificación del producto

Los planes del producto se utilizan para planificar y gestionar el trabajo. Un plan bien hecho incluye una estimación del coste del proyecto. Las estimaciones son esenciales para contratar desarrollos, ya que los clientes necesitan saber el precio por adelantado. Las estimaciones también son necesarias cuando se desarrollan productos. El coste del proyecto es la parte más importante del precio de un producto y debe ser lo bastante bajo para que el precio sea competitivo en el mercado.

El primer paso para hacer un plan de producto, es tener una definición clara de lo que se quiere producir; un adecuado plan requiere tres cosas:

- El tamaño y las características más importantes del producto a desarrollar
- Una estimación del tiempo requerido para hacer el trabajo
- Una previsión de la planificación

Un plan del producto identifica el desarrollo a hacer y contiene estimaciones del tamaño de la aplicación, las horas de trabajo y la programación. Productos más complejos requieren una planificación más sofisticada y más tipos de información, tales como asignación de responsabilidades, planes de personal, especificaciones

de procesos o productos, dependencias con otros grupos, pruebas especiales y cuestiones de calidad.

La Tabla 5 muestra un formato de trabajo para elaborar un plan de producto[9]

Desarrollador: _____ *Fecha:* _____

Trabajo	Fecha	Estimado		Real			Hasta la fecha				
		Tiempo	Unidades	Tiempo	Unidades	Velocidad	Tiempo	Unidades	Velo	Máx	Min
	Descripción:										

Tabla No.5 Plan de producto

La gestión de las programaciones[10]

Una programación es una lista ordenada por tiempos de eventos planificados. Para un proyecto de cualquier tamaño, el primer paso para hacer una programación es analizar el trabajo con bastante detalle para identificar las distintas tareas que lo componen. A continuación, estimar el tamaño para cada una de estas pequeñas tareas y determinar la cantidad de trabajo que probablemente necesitarán.

Puntos de control

En la planificación de cualquier proyecto, por pequeño que sea, es importante dividir el trabajo en varias partes que puedan ser estimadas y planificadas. Cada una de estas partes se puede tratar como un elemento de la programación. Es decir, cuando se completa cada parte, se ha realizado un determinado grado de progreso. Estos puntos de la programación que son medibles se llaman puntos de control o hitos. Los hitos son una parte importante de la planificación y gestión de proyectos.

Un hito es un punto que, objetivamente, se puede identificar es un proyecto. Cuando un plan incluye varios hitos, cada uno con una fecha de terminación planificada, se puede ver fácilmente si se está dentro de lo programado o se está retrasado.

Manejo de excepciones

Durante la ejecución del programa, los errores o eventos no esperados ocurren inevitablemente. Esto puede darse debido a un defecto en el programa o puede ser el resultado de circunstancias externas no predecibles. Un error o un evento

---

[9] Tomado de Introducción al Proceso Software Personal PSP
[10] Ver Anexo A

inesperado que ocurra durante la ejecución de un programa se denomina una *excepción*. Las excepciones pueden ser provocadas por condiciones de hardware o software.

Cuando ocurre una excepción, ésta debe ser manejada por el sistema. Esto puede hacerse dentro del mismo programa o puede implicar la transferencia de control a un mecanismo de manejo de excepciones del sistema. Normalmente, éste simplemente informa del error y abandona la ejecución. Por lo tanto, para asegurar que las excepciones del programa no provocan fallos de ejecución del sistema, debería definirse un manejador de excepciones para todas las posibles que puedan ocurrir y asegurarse de que todas se manejen de forma explícita.

Lenguajes de programación como *Java*, incluyen construcciones que soportan el manejo de excepciones de forma que no se necesitan secuencias condicionales adicionales para comprobar las excepciones. Este tipo de construcción se denomina ***Exception*** y diferentes excepciones pueden declararse con éste. Cuando ocurre una situación excepcional, la excepción es capturada y el soporte de ejecución del lenguaje transfiere el control a un manejador de excepciones. Éste es una sección de código que declara los nombres de las excepciones y las acciones adecuadas para manejar cada una.

El manejo de excepciones también puede utilizarse para simplificar los programas y hacerlos más fáciles de leer y entender. Esto reduce la probabilidad de error del programador e incrementa la posibilidad de que otro encuentre cualquier problema que exista. El manejo de excepciones se utiliza para separar el código de manejo de error del código que maneja el procesamiento normal. Por lo tanto, es posible leer y comprender cada una de estas secciones del código por separado.

DISEÑO ORIENTADO A OBJETOS

El Diseño orientado a objetos modela el software en términos similares a los que utilizan las personas para describir objetos del mundo real. Este diseño aprovecha las relaciones entre las *clases*, en donde los objetos de cierta clase tienen las mismas características. También aprovecha las relaciones de herencia, en donde las nuevas clases de objetos se derivan absorbiendo las características de las clases existentes y agregando sus propias características únicas.

El Diseño orientado a objetos (DOO) ofrece una manera más natural e intuitiva de ver el proceso de diseño: a saber, modelando los componentes de software de igual forma que como se describen los objetos del mundo real (por sus atributos y comportamientos). El DOO también modela la comunicación entre los objetos. Así como las personas se envían mensajes unas a otras, los objetos también se comunican mediante mensajes.

El DOO *encapsula* los atributos y las *operaciones* (comportamiento) en los *objetos;* los atributos y las operaciones de un objeto se enlazan íntimamente entre sí. Los objetos tienen una propiedad que se conoce como *ocultamiento de información*. Esto significa que, aunque los objetos pueden saber como comunicarse entre sí a través de *interfaces* bien definidas, generalmente no se les permite saber como se implementan otros objetos; los detalles de la implementación se ocultan dentro de los mismos objetos.

Los lenguajes como Java son orientados a objetos. La programación en dichos lenguajes se llama *programación orientada a objetos (POO)*, y permite a los diseñadores implementar un diseño orientado a objetos como un sistema funcional. En Java, la unidad de programación es la clase a partir de la cual se obtienen (crean) las instancias; es decir, los objetos. Las clases en Java contienen *métodos* (que implementan operaciones) y campos (que implementan atributos).

A/DOO es el término genérico para el proceso de analizar un problema y desarrollar un método para resolverlo. Los pequeños problemas no requieren de un proceso exhaustivo, podría ser suficiente con escribir *seudocódigo* antes de empezar a escribir el código. El seudocódigo es un medio informal de expresar el código de un programa. En realidad no es un lenguaje de programación, pero se puede usar como un tipo de "bosquejo" guía a medida que se escribe el programa.

El seudocódigo puede ser suficiente para los problemas pequeños, pero a medida que éstos y los grupos de personas que los resuelven se incrementan en tamaño, los métodos de A/DOO se vuelven más necesarios. Idealmente, un grupo debería acordar un proceso estrictamente definido para resolver su problema, y acordar

también una manera uniforme para que los miembros del grupo se comuniquen los resultados de ese proceso entre sí. Aunque existen muchos procesos de A/DOO distintos, existe un lenguaje gráfico para comunicar los resultados de cualquier proceso A/DOO que se ha vuelto muy popular. Este lenguaje se conoce como *Lenguaje Unificado de Modelado (UML)*. UML se desarrolló a mediados de la década de los noventa, bajo la dirección inicial de tres metodologistas de software: Grady Booch, James Rumaugh e Ivar Jacobson.

Booch, Rumaugh y Jacobson exponen la necesidad de un proceso de software "guiado por los *casos de uso*, *iterativo e incremental*", éste es El Proceso Unificado, el cual reconoce la importancia de comunicación con el cliente y los métodos encaminados a describir el punto de vista del cliente con respecto a un sistema y proporciona el sentido evolutivo esencial en el desarrollo del software moderno.

MODELOS DE DISEÑO - UML

El Lenguaje Unificado de Modelado (UML) es una notación que se produjo como resultado de la unificación de la técnica de modelado de objetos (OMT) propuesta por James Rumbaugh y Grady Booch y la ingeniería de software orientada a objetos (OOSE) de Ivar Jacobson.
UML es el lenguaje gráfico más utilizado para modelar sistemas orientados a objetos. Una de sus características fundamentales es la flexibilidad. UML es extensible e independiente de los diversos procesos de A/DOO. Los modeladores tienen la libertad de diseñar sistemas utilizando varios procesos, pero todos los desarrolladores pueden ahora expresar esos diseños con un conjunto de notaciones estándar.

El desarrollo de sistemas se enfoca en tres modelos diferentes del sistema:

- El modelo funcional, representado en UML con diagramas de caso de uso, describe la funcionalidad del sistema desde el punto de vista del usuario

- El modelo de objetos, representado en UML con diagramas de clase, describe la estructura de un sistema desde el punto de vista de objetos, atributos, asociaciones y operaciones

- El modelo dinámico, representado en UML con diagramas de secuencia, diagramas de estado y diagramas de actividad, describe el comportamiento interno del sistema.
  Los diagramas de secuencia describen el comportamiento como una secuencia de mensajes intercambiados entre un conjunto de objetos, mientras que los diagramas de estado describen el comportamiento desde le punto de vista de estados de un objeto individual y las transiciones posibles entre estados.

UML especifica nueve tipos de diagramas para modelar sistemas. Cada diagrama modela una característica distinta de la estructura o comportamiento de un sistema; los primeros cuatro diagramas se relacionan con la estructura del sistema, es decir, describen sus objetos y sus relaciones; los otros cinco se relacionan con el comportamiento del sistema, es decir, describen como cambia éste, a medida que sus objetos interactúan entre sí. Todos los sistemas tienen tanto estructura como comportamiento.

1. Los *diagramas de clases*, modelan la estructura del sistema. Las clases son abstracciones que especifican la estructura y el comportamiento común de un conjunto de objetos. Los objetos son instancias de las clases que se crean, modifican y destruyen durante la ejecución del sistema. Los objetos

2. tienen estados que incluyen los valores de sus atributos y sus relaciones con otros objetos.
   Los diagramas de clases describen el sistema desde el punto de vista de objetos, clases, atributos, operaciones y asociaciones.

3. Los *diagramas de objetos,* modelan una "instantánea" del sistema, al modelar sus objetos y las relaciones de éstos en un punto específico en el tiempo. Cada objeto representa la instancia de una clase del diagrama de clases, y puede haber varios objetos creados a partir de una clase.

Las ***clases*** son abstracciones que especifican los atributos y comportamientos de un conjunto de objetos. Los ***objetos*** son entidades que encapsulan estado y comportamiento. Cada objeto tiene una identidad: se puede hacer referencia a él de manera individual y es distinguible con respecto a otros objetos.

En UML las clases y objetos se muestran mediante cuadros que influyen tres compartimientos. El compartimiento superior muestra el nombre de la clase u objeto. El compartimiento central muestra sus atributos y el compartimiento inferior muestra sus operaciones. Por convención, los nombres de clase comienzan con minúscula. El tipo de un atributo se usa para especificar el rango válido de valores que puede tener un atributo.

Asociaciones y vínculos

Un **vínculo** representa una conexión entre dos objetos. Las **asociaciones** son relaciones entre clases y representan grupos de vínculos. Cada extremo de una asociación puede etiquetarse con un texto llamado **papel**. El etiquetado de los extremos de la asociación con papeles permite distinguir entre varias asociaciones que se originan en una clase. Además, los papeles aclaran el propósito de la asociación.

Multiplicidad

Cada extremo de una asociación puede ser etiquetado con un conjunto de enteros que indica la cantidad de vínculos que se pueden originar a partir de una instancia de la clase conectada al extremo de la asociación. Cuando la multiplicidad es "muchos" se representa con un asterisco, esta multiplicidad es una abreviatura de 0...n.

Agregación

Las asociaciones se usan para representar un amplio rango de conexiones entre un conjunto de objetos. En la práctica sucede a menudo un caso especial de asociación: *la composición*; tales relaciones podrían modelarse usando una asociación de uno a muchos. En vez de ello, UML proporciona el concepto de

agregación para indicar la composición. Una *agregación* se indica mediante una línea simple con un rombo en el extremo del contenedor de la asociación. Aunque las asociaciones de uno a muchos y las agregaciones pueden usarse en forma alterna, se prefieren las agregaciones debido a que enfatizan los aspectos jerárquicos de la relación.

Generalización

La **generalización** es la relación entre una clase general y una o más clases más especializadas. La generalización permite describir todos los atributos y operaciones que son comunes para un conjunto de clases. A la generalización, se le llama **superclase** y a las especializaciones, se les llama **subclases**; estas *heredan* los atributos y operaciones de su clase de origen. Las clases **abstractas** se usan en el modelado orientado a objetos para clasificar conceptos relacionados, reduciendo, por lo tanto, la complejidad general del modelo.
El comportamiento de los objetos se especifica mediante **operaciones**. Un conjunto de operaciones representa un **servicio** proporcionado por una clase particular. Un objeto solicita la ejecución de una operación de otro objeto enviándole un **mensaje**. El mensaje concuerda con un **método** definido por la clase a la que pertenece el objeto receptor o por cualquiera de sus superclases. Las operaciones de una clase son los servicios públicos que proporciona la clase. Los métodos de su clase son las implementaciones de estas operaciones.

4. Los *diagramas de componentes,* modelan los artefactos y componentes de software como: componentes de código, componentes binarios que son los generados por la compilación de los componentes de código y los componentes ejecutables, en este diagrama también se pueden modelar los paquetes (que son grupos de clases) que conforman el sistema.

   Los componentes de software se representan con un rectángulo que contiene dos pequeños rectángulos en su extremo izquierdo. Cada paquete debe tener un diagrama de componentes para representar las clases que contiene internamente.

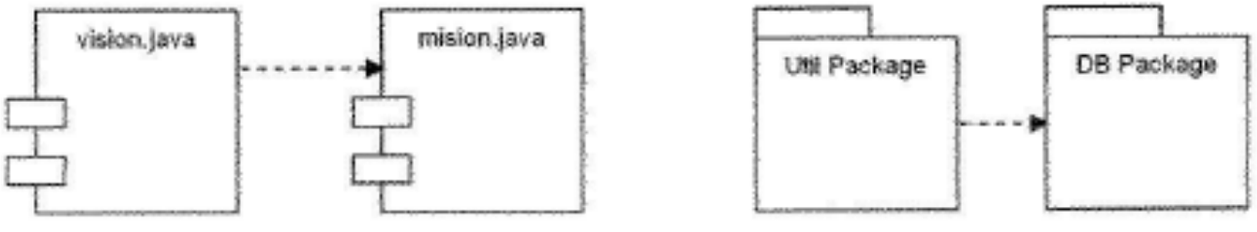


Figura No.9 Diagrama de componentes - Paquetes

5. Los *diagramas de distribución* modelan los requerimientos en tiempo de ejecución del sistema (tales como el o los computadores en los que residirá el sistema), los requerimientos de memoria para el sistema u otros dispositivos que éste requiera durante la ejecución.
   Es decir se sitúa el software en el hardware que lo contiene. Cada Hardware se representa como un nodo. Un nodo se representa como un cubo, un nodo es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes.

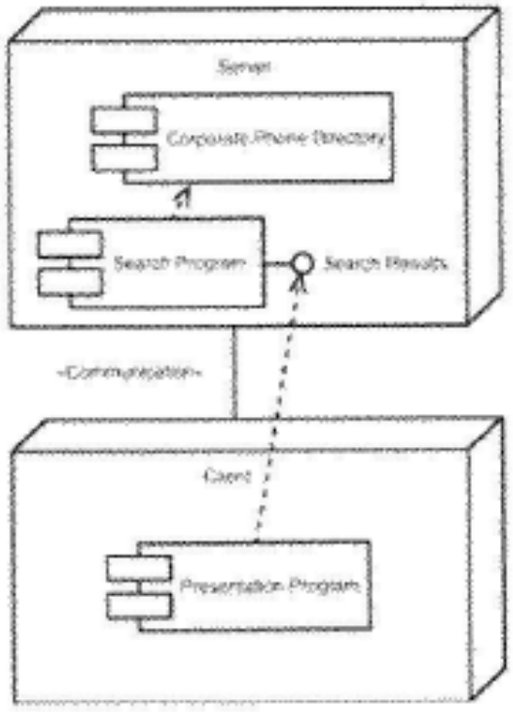


Figura No.10 Diagrama de despliegue

6. Los *diagramas de estados,* modelan cómo un objeto cambia de *estado* es decir, la condición de un objeto en un tiempo específico. Cuando un objeto cambia de estado, ese objeto puede comportarse de manera distinta en el sistema.

7. Los *diagramas de actividad,* modelan la actividad de un objeto: el flujo de trabajo del objeto durante la ejecución del programa. Un diagrama de actividad es un diagrama de flujo que modela las acciones que el objeto realizará, y en qué orden.

Diagramas de interacción

8. Los *diagramas de colaboración* modelan las interacciones entre objetos en un sistema, con un énfasis acerca de qué interacciones ocurren. Modelan los aspectos de un sistema relacionados con los comportamientos, al

proporcionar información acerca de cómo interactúan los objetos. Los diagramas de colaboración enfatizan qué objetos participan en las interacciones. Los objetos se modelan con nombres de la forma **nombreObjeto:NombreClase**

9. Los *diagramas de secuencia* también modelan las interacciones entre los objetos en un sistema, pero a diferencia de los diagramas de colaboración, enfatizan cuándo ocurren las interacciones. Un objeto interactúa con otro objeto enviando **mensajes**. La recepción de un mensaje por parte de un objeto activa la ejecución de una operación, la cual, a su vez, puede enviar mensajes a otros objetos. Se pueden pasar **argumentos** junto con un mensaje y se asocian a los parámetros de la operación que se va a ejecutar en el objeto que los recibe.

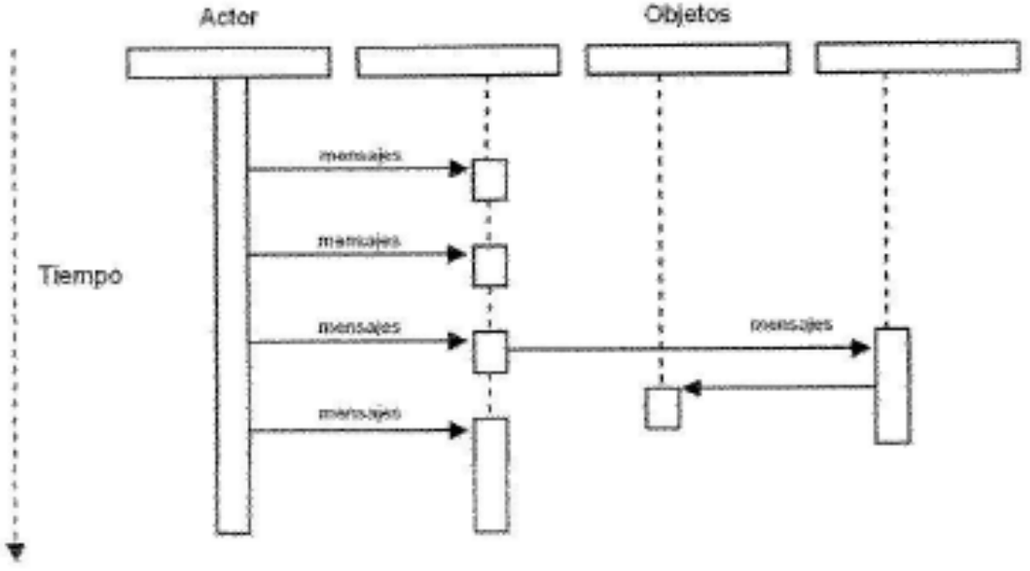


Figura No.11 Diagrama de secuencia

Cada columna representa un objeto que participa en la interacción. El eje vertical representa el tiempo de arriba hacia abajo. Los mensajes se muestran con flechas. Los rótulos de las flechas representan nombres de mensajes y pueden contener argumentos. Las activaciones (es decir, la ejecución de métodos) se muestran con rectángulos verticales. Los actores se muestran en la columna de la extrema izquierda.

Los diagramas de secuencia pueden usarse para describir una secuencia abstracta (es decir, todas las interacciones posibles) o secuencias concretas (es decir, una interacción posible). Cuando se describen todas las interacciones posibles, los diagramas de secuencia también proporcionan notaciones para condiciones e iteradores. Una condición en

un mensaje se indica por una expresión entre corchetes antes del nombre del mensaje. Si la expresión es cierta se envía el mensaje. La invocación repetitiva de un mensaje se indica con un "*" antes del nombre del mensaje.

10. Los *diagramas de caso-uso* representan la interacción entre el usuario y el sistema, es decir, todas las acciones que el usuario puede realizar en el sistema.

CASOS DE USO

Para desarrollar un sistema aplicando un enfoque orientado a objetos, es sumamente útil extraer y describir los requerimientos mediante una técnica conocida como *casos de uso*. Un caso de uso describe una funcionalidad particular que se supone que el sistema debe realizar o exhibir, modelando el diálogo que un usuario, un sistema externo u otra entidad tendrán con el sistema a ser desarrollado.
La entidad que interactúa con el sistema se denomina ***actor*** y puede ser un usuario, un dispositivo u otro sistema.

Cada caso de uso describe un posible escenario de la interacción de la entidad externa con el sistema. A menudo el caso de uso se representa como un dibujo de los objetos involucrados, acompañado de una breve descripción textual (el guión del escenario) que indica como se realiza la función. Para cada escenario, el caso de uso identifica todos los eventos que pueden ocurrir, así como las respuestas del sistema. Los casos de uso en su totalidad constituyen una descripción completa de todas las formas posibles de utilización del sistema por todas las posibles entidades.

En general, un caso de uso describe una secuencia de acciones que desarrolla un *actor* cuando éste interactúa con el software. Los casos de uso ayudan a identificar el ámbito del proyecto y proporcionan una base para la planeación de éste.

Para identificar los actores dentro de un sistema, se pueden utilizar las siguientes preguntas:

- ¿Qué usuarios o grupos realizan sus tareas usando el sistema?
- ¿Qué usuarios o grupos son indispensables para que el sistema complete sus funciones?
- ¿Qué sistemas externos usan el sistema para realizar una tarea?
- ¿Qué sistemas externos, usuarios o grupos envían información al sistema?
- ¿Qué sistemas externos, usuarios o grupos reciben información del sistema?

Una manera de incrementar la comprensión consiste en documentar la relación entre casos de uso. Es decir, se pone de manifiesto cuáles casos de uso son dependientes de otros. Se buscan aquellos actores que están involucrados en cada caso de uso y se asegura que cada caso tenga todos los actores que necesita. De manera similar se examinan las dependencias entre éstos: ¿Cuáles

deben finalizar antes de que un caso particular pueda empezar?, ¿cuáles pueden comenzar cuando uno particular termina?, ¿existen conflictos o inconsistencias?

UML proporciona el *diagrama de caso-uso* para facilitar la recopilación de los requerimientos. Este diagrama modela las interacciones entre los clientes externos del sistema y los casos de uso del mismo. Cada caso de uso representa una herramienta distinta que el sistema proporciona a sus clientes. Por ejemplo, las máquinas de cajero automático generalmente tienen varios casos de uso, incluyendo "Depositar dinero", "Retirar dinero" y "Transferir fondos".

En sistemas grandes, los diagramas de caso-uso son herramientas indispensables que ayudan a los desarrolladores de software a mantenerse enfocados en satisfacer las necesidades de los usuarios. El objetivo de éste diagrama es mostrar los tipos de interacciones que tienen los usuarios con un sistema, sin tener que proporcionar los detalles de esas interacciones; desde luego que esos detalles se proporcionan en otros diagramas de UML. Los diagramas de caso-uso vienen acompañados de texto informal que describe las interacciones con más detalle.

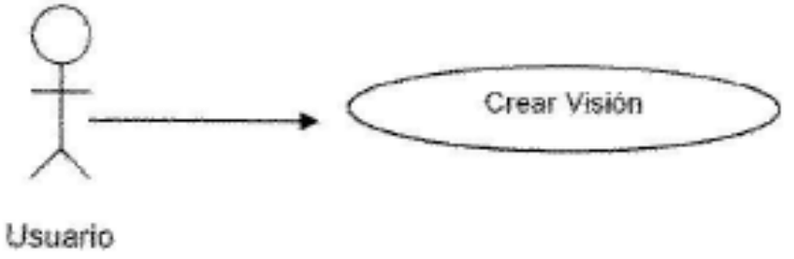


Figura No.12 Casos de Uso

En la figura 12 se muestra un caso de uso. La figura lineal representa a un actor, el cual a su vez representa a un conjunto de roles que puede desempeñar una entidad externa (como un a persona u otro sistema). UML modela cada caso de uso como un óvalo. Debemos asegurarnos que nuestros casos de uso no modelen interacciones que sean demasiado específicas entre el cliente externo y le sistema. Una subdivisión incorrecta y repetitiva de los casos de uso podría crear problemas durante la implementación.

Escenarios

Un caso de uso es una abstracción que describe todos los escenarios posibles que involucran la funcionalidad que se describe. Un **escenario** es una instancia de un caso de uso que representa un conjunto de acciones concretas. Los escenarios se usan como ejemplos para ilustrar casos comunes. Su enfoque es la comprensión.

Los diagramas de caso de uso pueden incluir cuatro tipos de relaciones: comunicación, inclusión, extensión y generalización.

Relaciones de comunicación

Los actores y los casos de uso se **comunican** cuando intercambian información entre ellos. Las relaciones de comunicación se muestran con una línea continua entre los símbolos de actor y caso de uso. Pueden emplearse para indicar acceso a funcionalidad.

REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

Los requerimientos de software se clasifican en funcionales y no funcionales:

- Requerimientos funcionales: Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente los que el sistema no debe hacer
- Requerimientos no funcionales: Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.

Requerimientos funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del mismo y del enfoque general tomado por la organización al redactar requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo, los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

Estos requerimientos funcionales del usuario definen los recursos especificos que el sistema debe proporcionar e ilustran los diferentes niveles de detalle den que se pueden redactar los requerimientos funcionales.

La impresión en la especificación de requerimientos es la causa de muchos de los problemas de la ingeniería del software. Para un desarrollador es natural dar interpretaciones de un requerimiento ambiguo con el fin de simplificar su implementación. Sin embargo, a menudo no es lo que el cliente desea. Se deben establecer nuevos requerimientos y hacer cambios en el sistema, lo que retrasa la entrega e incrementa los costos.

En principio, la especificación de requerimientos funcionales de un sistema debe estar completa y ser consistente. La completitud significa que todos los servicios solicitados por el usuario deben estar definidos. La consistencia significa que los requerimientos no deben tener definiciones contradictorias. En la práctica, para

sistemas grandes y complejos, es casi imposible alcanzar los requerimientos de consistencia y completitud.

Una razón de esto es que es fácil cometer errores y omisiones cuando se redactan especificaciones para sistemas grandes y complejos. Otra razón es que los usuarios del sistema tienen necesidades diferentes, y a menudo contradictorias. Estas contradicciones pueden no ser obvias cuando los requerimientos se especifican por primera vez, es posible que los problemas surjan solamente después de un análisis más profundo o, a veces, después de que se termine el desarrollo y el sistema se entregue al cliente.

Requerimientos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Los requerimientos no funcionales surgen de las necesidades del usuario, debida a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware, o factores externos como regulaciones de seguridad o legislaciones sobre privacidad.

Los tipos de requerimientos no funcionales son:

- Requerimientos del producto: Especifican el comportamiento del producto. Algunos ejemplos son los requerimientos de rendimiento en la rapidez de ejecución del sistema y cuanta memoria se requiere; los requerimientos de fiabilidad que fijan la tasa de fallos para que el sistema sea aceptable; los requerimientos de portabilidad, y los de usabilidad

- Requerimientos organizacionales: Se derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador. Algunos ejemplos son los estándares en los procesos que deben utilizarse; los requerimientos de implementación, como los lenguajes de programación o el método de diseño a utilizar, y los requerimientos de entrega que especifican cuando se entregará el producto y su documentación

- Requerimientos externos: Incluye todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo. Éstos pueden incluir los requerimientos de interoperabilidad que definen la manera en que el sistema interactúa con sistemas de otras organizaciones; los requerimientos legislativos que deben seguirse para asegurar que el

sistema funcione dentro de la ley, y los requerimientos éticos. Estos requerimientos son puestos en un sistema para asegurar que será aceptado por sus usuarios y por el público en general.

Un problema común con los requerimientos no funcionales es que pueden ser difíciles de verificar. Los usuarios o clientes declaran a menudo estos requerimientos como metas generales tales como la facilidad de uso, la capacidad del sistema para recuperarse de los fallos o la respuesta rápida al usuario. Estas metas imprecisas causan problemas a los desarrolladores del sistema puesto que dejan abierta la posibilidad a interpretación, lo que provoca discusiones subsecuentes una vez que el sistema se entrega. Siempre que sea posible, se deben redactar los requerimientos no funcionales de manera cuantitativa para que se puedan probar de un modo objetivo.

A menudo, los requerimientos no funcionales entran en conflicto e interactúan con otros requerimientos funcionales o no funcionales. Es de utilidad diferenciar los requerimientos; en la práctica esto es difícil ya que si se declaran de forma separada ver la relación entre ellos y distinguir los que se refieren al sistema como un todo se complica.

El PROCESO UNIFICADO DE RATIONAL (RUP)

El RUP es un ejemplo de un modelo de proceso moderno que proviene del trabajo en UML y el asociado Proceso Unificado de Desarrollo de Software y que se describe normalmente desde tres perspectivas:

- Una perspectiva dinámica que muestra las fases del modelo sobre el tiempo
- Una perspectiva estática que muestra las actividades del proceso que se representan
- Una perspectiva práctica que sugiere buenas prácticas a utilizar durante el proceso

El RUP es un modelo en fases que identifica cinco diferentes en el proceso de desarrollo de software que son:

1. Inicio: Abarca la comunicación con el cliente y las actividades de planeación. Al colaborar con los clientes y usuarios finales se identifican los requisitos para el software, se propone una arquitectura aproximada para el sistema, y se desarrolla un plan para la naturaleza iterativa e incremental del sistema subsiguiente. Los requisitos fundamentales del software se describen a través de un conjunto preliminar de *casos de uso*.

2. Elaboración: Refina y expande los casos de uso preliminares que se desarrollaron como una parte de la fase de inicio. Además, el plan se revisa de manera cuidadosa al término de la fase de elaboración para asegurar que el ámbito, los riesgos y los datos entregados aún son razonables. Las modificaciones al plan se deben hacer en este momento.

3. Construcción: Desarrolla o adquiere los componentes del software que harán que cada caso de uso sea operativo para los usuarios finales. Todas las características y funciones necesarias y requeridas del incremento del software se implementan en código fuente. Conforme los componentes están en proceso de implementación, se diseñan y ejecutan pruebas de unidad para cada uno de ellos. Además, se realizan las actividades de integración (ensamblaje de componentes y pruebas de integración). Los casos de uso se utilizan para derivar un conjunto de pruebas de aceptación que se ejecutan antes del inicio de la siguiente fase del RUP.

4. Transición: Abarca las últimas etapas de la actividad de construcción y la primera parte de la de despliegue o puesta en marcha. El software se entrega a los usuarios finales para realizar pruebas beta, y la retroalimentación del usuario reporta tanto defectos como cambios necesarios. Además, el equipo

5. de desarrollo crea la información de soporte necesaria (manuales de usuario, guías de resolución de problemas, procedimientos de instalación) para el lanzamiento. Al final de esta fase, el incremento de software se convierte en un lanzamiento del software utilizable

6. Producción: Coincide con la actividad de puesta en marcha. Durante esta fase se monitorea el uso subsiguiente del software, se proporciona el soporte para el ambiente operativo (infraestructura), y se reciben y evalúan los informes de defectos y los requerimientos de cambios.

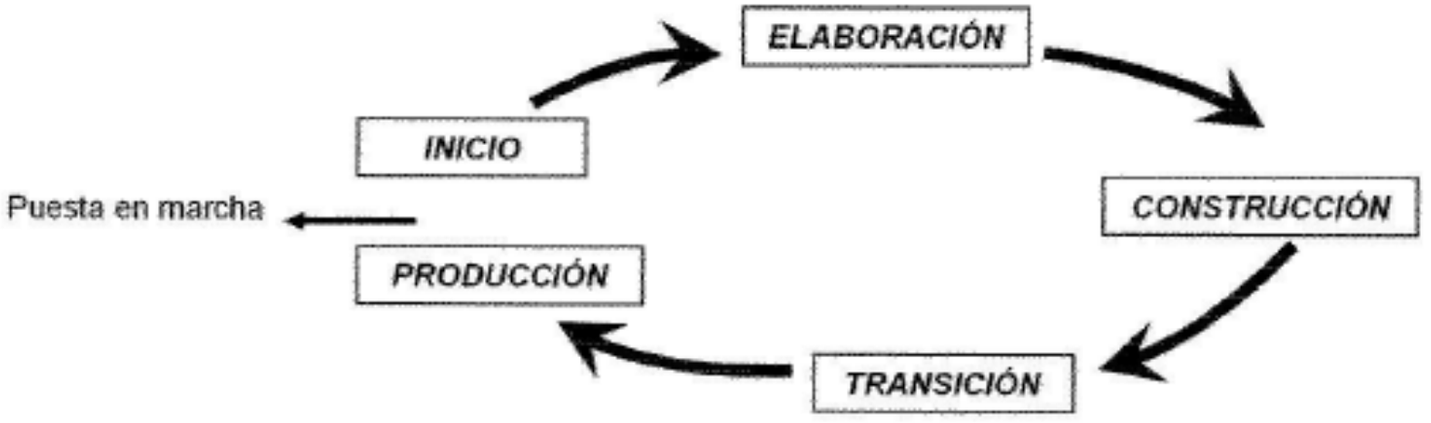


Figura No.13 Etapas del RUP

Es probable que mientras se realizan las fases de construcción, transición y producción ya se hayan iniciado los trabajos para el siguiente incremento del software; esto significa que las cinco fases del RUP no suceden en una secuencia, sino en una concurrencia por etapas.

A lo largo de las fases del RUP se distribuye un flujo de trabajo de ingeniería del software. Esto es, un flujo de trabajo identifica las tareas necesarias para completar una acción importante de ingeniería y los productos de trabajo que se producen como consecuencia de la realización exitosa de tareas. Se debe destacar que no todas las tareas identificadas para un flujo de trabajo se realizan para cualquier proyecto de desarrollo; el equipo debe adaptar el proceso (acciones, tareas, actividades y productos) para satisfacer sus necesidades.

Productos de trabajo del RUP

Desde el punto de vista del ingeniero, el producto más importante generado durante la fase de inicio es el *modelo de casos de uso*: que describen la forma en que actores externos interactúan con el sistema y obtienen valor a partir de éste.

La fase de elaboración produce un conjunto de productos de trabajo que elabora requisitos (funcionales y no funcionales). Cuando el ingeniero inicia con el análisis

orientado a objetos, el objetivo primordial es definir un conjunto de clases que describan en forma adecuada el comportamiento del sistema. Además, en la fase de elaboración se revisan los riesgos y el plan de proyecto para asegurar que cada uno de ellos conserve su validez.

La fase de construcción produce un modelo de implementación que traduce las clases de diseño en componentes del software que se construirán para ejecutar el sistema y un modelo de prueba que describe las que se emplean para asegurar que los casos de uso se reflejen de manera apropiada en el software que se ha desarrollado.

La fase de transición entrega el incremento del software y evalúa los productos de trabajo elaborados durante la etapa en que los usuarios finales trabajan con la aplicación. En esta etapa se produce la retroalimentación proveniente de las pruebas beta y en la fase de producción se producen los requerimientos cualitativos de cambio.

Figura No.14 Productos de trabajo del RUP

PROCESO DE DISEÑO DE LA INTERFAZ DE USUARIO

El diseño de la interfaz de usuario (UI, por sus siglas en ingles) es un proceso iterativo donde los usuarios interactúan con los diseñadores y prototipos de la interfaz para decidir las características, organización, apariencia y funcionamiento de la interfaz de usuario del sistema. A veces, se construye el prototipo de la interfaz por separado en paralelo con otras actividades de la ingeniería del software. Más comúnmente, en especial cuando se utiliza un desarrollo iterativo, el diseño de UI se lleva a cabo de forma incremental conforme se desarrolla el software. En ambos casos, sin embargo, antes de que empiece la programación, debe haber desarrollado e, idealmente, probado algunos diseños en papel.

En la figura 15 se ilustra el proceso de diseño general de la UI. Existen tres actividades esenciales en este proceso:

1. Análisis del usuario: Se desarrolla una compresión de las tareas que éste realiza, su entorno de trabajo, los otros sistemas que utiliza, como interactúan con el resto de las personas en su trabajo, etc. Para productos con una diversa variedad de usuarios, se debe intentar desarrollar esta comprensión a través de grupos de discusión, pruebas con usuarios potenciales y ejercicios similares.

2. Desarrollo del prototipo de la UI: Idealmente cuando se está construyendo el prototipo de una interfaz de usuario, se debe adoptar un proceso en dos etapas:

    - Al principio del proceso, hay que desarrollar prototipos en papel - maquetas de los diseños de las pantallas - y mostrárselos a los usuarios finales
    - Entonces, se perfecciona el diseño y se desarrollan prototipos automatizados cada vez más sofisticados, y se ponen a disposición de los usuarios para realizar pruebas y simulación de actividades.

    La construcción de prototipos en papel es un enfoque poco costoso y sorprendentemente efectivo para el desarrollo de prototipos. No se necesita desarrollar ningún software ejecutable y los diseños no tienen porque hacerse conforme a estándares profesionales. Se pueden hacer versiones en papel de las pantallas del sistema con las que interactuaran los usuarios y se puede diseñar un conjunto de escenarios que describan cómo podría utilizar el sistema. Conforme se desarrolla un escenario, hay que esbozar la información que mostrará y las opciones disponibles a los usuarios

3. Evaluación de la interfaz: Aunque obviamente se habrá hablado con los usuarios durante el proceso de desarrollo del prototipo, también se debe tener una actividad de evaluación más formalizada donde se recopile información sobre las experiencias reales de los usuarios con la interfaz.

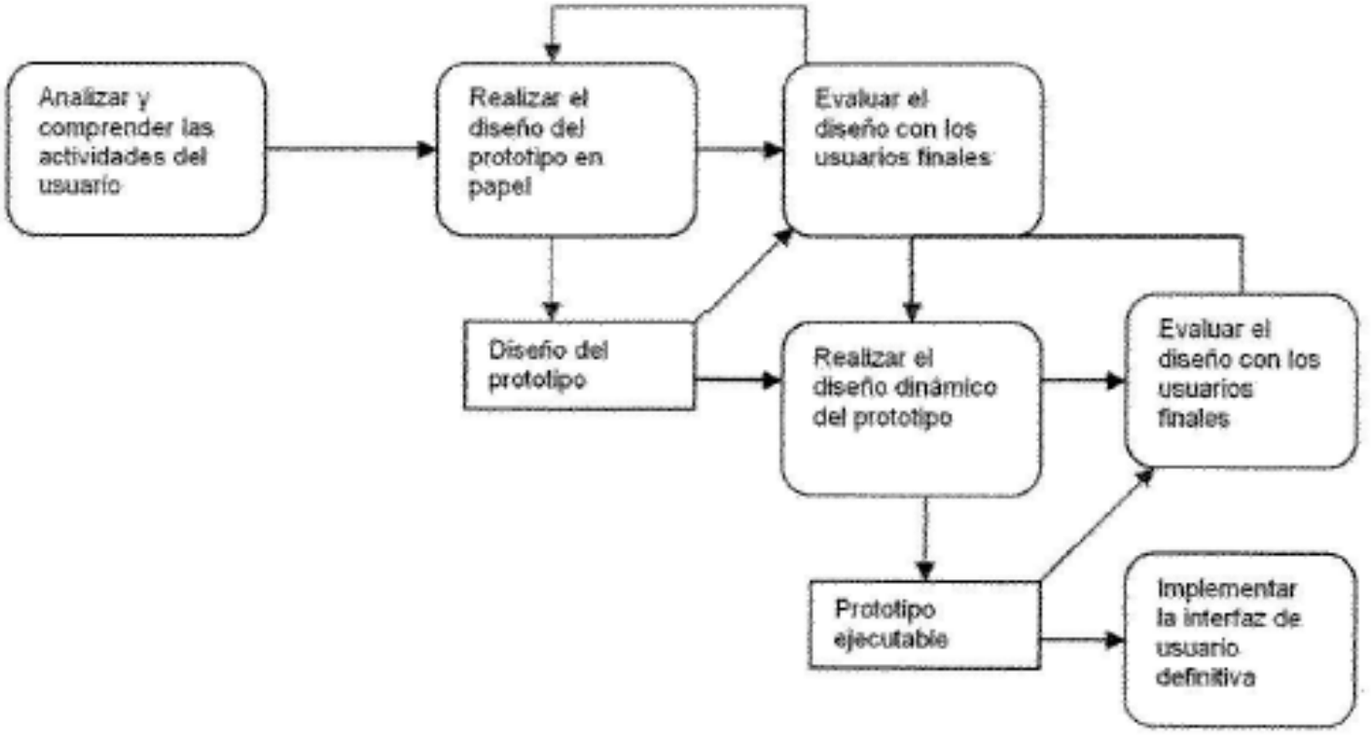


Figura No.15 El proceso de diseño de la UI

METODOLOGÍAS ÁGILES

En febrero de 2001, tras una reunión celebrada en Utah - Estados Unidos, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participó un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software; su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente respondiendo a los cambios que puedan surgir a lo largo del proyecto.
Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.
Tras esta reunión se creó The Agile Alliance[11], una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume esta filosofía.

El Manifiesto Ágil

Según el manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo con base a sus necesidades.

- Desarrollar software que funcione más que conseguir una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

- La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

[11] www.agilealliance.com

- Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

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

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. El usuario y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI. El trabajo en parejas es el método más eficiente, eficaz y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.

VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener un buen clima organizacional.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y que exigen reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

Los **procesos ágiles** de desarrollo de software, conocidos anteriormente como *metodologías livianas*, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

PROGRAMACIÓN EXTREMA (XP)

La Programación Extrema (XP, por sus siglas en ingles) es posiblemente el método ágil más conocido y ampliamente utilizado. Desarrollado por Kent Beck propone potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y adaptación a los cambios. XP es adecuada para proyectos con requisitos imprecisos, cambiantes y donde existe un alto riesgo técnico.

Las características esenciales de XP son: historias de usuario, roles, proceso y prácticas.

Las historias de usuario

En XP todos los requerimientos se expresan como escenarios (llamados historias de usuario), los cuales se implementan directamente como una serie de tareas.
Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Estas tarjetas contienen la siguiente información:

Fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado de cosas por terminar y comentarios.

A efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración). Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Roles en XP

Los roles de acuerdo con la propuesta original de Beck son:

- Programador: El programador escribe las pruebas unitarias y produce el código del sistema.

- Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

- Encargado de pruebas: Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

- Encargado de seguimiento: Proporciona retroalimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

- Entrenador: Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

- Gestor: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso XP

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de cuatro fases: *Planeación, Diseño, Desarrollo (codificación) y Pruebas.*

Planeación

En un proceso de XP, los clientes están fuertemente implicados en la especificación y establecimiento de prioridades de los requerimientos del sistema. Desarrollan conjuntamente con los desarrolladores una "tarjeta de historias" que recoge las necesidades del cliente.
Una vez que se han desarrollado las tarjetas, el equipo de desarrollo las divide en tareas y estima el esfuerzo y recursos requeridos para su implementación. El cliente establece entonces la prioridad de las historias a implementar, eligiendo aquellas que pueden ser utilizadas inmediatamente para entregar un apoyo útil al negocio. Por supuesto, cuando los requerimientos cambian, las historias sin implementar también cambian o se pueden descartar. Si se requieren cambios en un sistema que ya se ha entregado, se desarrollan nuevas tarjetas y, de nuevo, el cliente decide si estos cambios tienen prioridad sobre nuevas funcionalidades.

Diseño

El diseño en XP sigue de manera rigurosa el principio de "mantenerlo simple", es decir:

- Funciona con todas las pruebas.
- No tiene lógica duplicada.
- Manifiesta cada intención importante para los programadores
- Tiene el menor número de clases y métodos.

La programación extrema apoya la refabricación, una técnica de construcción que también lo es de diseño: "Refabricación es el proceso de cambiar un sistema de software de tal manera que no altere el comportamiento externo del código y que mejore la estructura interna. Es una manera disciplinada de limpiar el código, lo que minimiza las oportunidades de introducir errores. En esencia, al refabricar, se mejora el diseño del código después de que se ha escrito".

Desarrollo (codificación)

XP recomienda que después de diseñar las historias y realizar el trabajo de diseño preliminar el equipo no debe iniciar la codificación, sino que debe desarrollar una serie de pruebas de unidad que ejerciten cada una de las historias que vayan a incluirse en el lanzamiento actual (incremento del software).

Una vez creada la prueba de unidad, el desarrollador es más capaz de centrarse en lo que debe implementarse para pasar la prueba. No se agrega nada extraño (mantenerlo simple). Una vez que el código esta completo, la unidad puede probarse de inmediato, y así proporcionar una retroalimentación instantánea a los desarrolladores.

Un concepto clave durante la actividad de codificación es la *programación en pareja*. XP recomienda que dos personas trabajen juntas en una estación de trabajo para crear el código de una historia. Esto proporciona un mecanismo para la resolución de problemas en tiempo real y el aseguramiento de la calidad en las mismas condiciones. En la práctica, cada persona tiene un papel sutilmente diferente. Por ejemplo, una persona puede pensar en los detalles de codificación de una porción particular del diseño, mientras que la otra se asegura de que se sigan los estándares de codificación y que el código que se genera "coincida" con el diseño más amplio de la historia.

Cuando los programadores completan su trabajo el código que desarrollaron se integra con el trabajo de otros. En algunos casos esto lo lleva a cabo diariamente el equipo de integración; en otros casos, la pareja de programadores es la responsable de la integración. Esta estrategia de "integración continua" ayuda a evitar problemas de compatibilidad e interfaz y ayuda a descubrir los errores desde el principio.

Pruebas

Las pruebas de unidad que se crean deben implementarse con un marco de trabajo que permita automatizarlas (por lo tanto, puedan ejecutarse de manera fácil y repetida). Esto apoya una estrategia de regresión de prueba cuando el código se modifica, al cual a menudo se le confiere la filosofía de XP de refabricar.

Cuando las unidades individuales de prueba se organizan en un conjunto universal de pruebas, las pruebas de integración y validación del sistema pueden realizarse a diario. Esto proporciona al equipo una indicación continua del progreso y también puede encender luces de emergencia previas si las cosas salen mal.

Las *pruebas de aceptación*, las especifica el cliente y se enfocan en las características generales y la funcionalidad del sistema, elementos visibles y revisables por el cliente. Las pruebas de aceptación se derivan de las historias del usuario que se han implementado como parte de un lanzamiento de software.

APLICABILIDAD DEL MARCO TEÓRICO AL PROYECTO

Antes que nada, este trabajo tiene un por qué; personalmente, esta fue la idea que más me gustó con respecto a los demás temas que se plantearon junto con el Archivo de Bogotá y me gustó por aquello del Balanced Scorecard. Pero, este proyecto también tiene una justificación gerencial y legal[12].

La justificación gerencial, como se dijo en la introducción a este trabajo escrito, es brindarle al Archivo de Bogotá una herramienta que le permita conocer acerca de una metodología administrativa que facilite el seguimiento a su desempeño funcional; como todo proyecto este tiene un alcance y junto con la entidad se acordó que se entregaría una aplicación que permitiera conceptualizar los términos propios de un Cuadro de Mando Integral y que sirviera de inicio y apoyo para el trabajo nada fácil de gerenciar; tanto el Archivo de Bogotá como yo sabemos que para hablar de un Balanced Scorecard hace falta incorporarle muchas más funcionalidades a la aplicación; pero sin duda con este proyecto logramos que la entidad conociera ésta metodología gerencial e implementará algo que notablemente hacía falta.

En esta parte del trabajo, me parece oportuno hablar sobre el Archivo de Bogotá:

El 6 de agosto de 2003, se inauguró el Archivo de Bogotá como centro para la conservación de la memoria de la ciudad, garante de la transparencia y de los derechos ciudadanos y ente rector del Sistema Distrital de Archivos. El proyecto se planteó en 1997, durante la primera administración del alcalde Antanas Mockus y fue continuado por el alcalde Enrique Peñaloza, hasta su realización en la actual administración; con el propósito de recuperar, conservar, divulgar y servir, memoria documental para promover el conocimiento y la imagen colectiva e histórica de Bogotá.

Con la construcción del Archivo y su puesta en servicio, la administración de la ciudad cumple con lo que manda la Constitución Política de Colombia, la Ley General de Archivos, y el Plan de Desarrollo en sus conceptos esenciales de fortalecer la transparencia administrativa, sustentar la rendición de cuentas y garantizar la conservación y difusión de la memoria institucional.[13]

El Archivo de Bogotá, general e histórico de la ciudad, tiene como misión la protección de los recursos documentales del Distrito Capital, con el propósito de garantizar la transparencia, accesibilidad y conservación de la información de interés para el gobierno y el estudio de la ciudad. En este sentido, el Archivo de Bogotá rige el Sistema de Archivos de la Administración Distrital; acopia,

---

[12] Para ver la justificación legal consultar el Anexo A

[13] Tomado de la página Web: www.alcaldiabogota.gov.co/archivo

conserva, organiza y sirve fondos y colecciones con valor patrimonial, y difunde la memoria contenida en el acervo documental en beneficio de las entidades de la administración distrital, investigadores y estudiosos de la ciudad y toda otra persona interesada en conocer la historia de Bogotá.

El Archivo de Bogotá es una dirección de la Secretaría General de la Alcaldía Mayor de Bogotá y como tal se ve inmersa en una serie de actividades tales, como la de implementar el Sistema Integrado de Gestión - SIG - dentro de ese sistema, actualmente la Secretaría General avanza en un proyecto referente a Cuadro de Mando Integral y aunque aún se desconoce cuando se pondrá en marcha, el Archivo desde inicios de este año está comprometido en alcanzar las metas que como dirección tiene con respecto al SIG; una de las estrategias para este fin, fue la de contratar una persona para desarrollar Balanced Scorecard para la Subdirección del Sistema de Archivos de la Administración Distrital.

Durante los primeros meses de investigación sobre Cuadro de Mando Integral se hizo una serie de presentaciones a la entidad exponiendo el tema, una primera presentación teórica fue al personal del Archivo, en una segunda se expuso el tema a las directivas y en una tercera presentación se enseñó los avances hasta ese punto de la aplicación, a las directivas y a la oficina de Control Interno quienes se mostraron interesados en el proyecto.

En las siguientes páginas trataré de plasmar todo el trabajo realizado a lo largo de este proceso.

1. Etapa de Planeación

Para este proyecto se estimó, que la aplicación no tendría más de 3000 líneas de código, que se ejecutarían pruebas inmediatas al culminar una instrucción y que existiría una programación previa de diseño y desarrollo; es decir que se estimó que la aplicación sería desarrollada en un lapso de 15 semanas. En este proyecto los puntos de control se ubicaron al culminar cada etapa del desarrollo.

Aunque es un proyecto pequeño, inicialmente se planearon las actividades que se iban a desarrollar con el Ingeniero de Sistemas del Archivo de Bogotá; para esto utilizamos la planificación de actividades, el registro de tiempos y el resumen semanal de actividades.

Pienso, que cuando se planea una serie de actividades en las cuales manejas la incertidumbre en cuanto a lo que vas a ejecutar, muchas de las tareas no se desarrollarán dentro del tiempo establecido; esto porque desconocía el contexto tanto de la parte administrativa como de la parte de desarrollo de software; me tomo más tiempo del inicialmente planeado en adquirir el conocimiento sobre Cuadro de Mando Integral y mucho más fue el tiempo que necesite para manejar

básicamente el lenguaje de programación Java y entender como funcionan las aplicaciones Web.

En general, aunque notoria durante el tiempo en el que trabaje este proyecto, la desviación entre lo planeado y lo ejecutado no impidió que cumpliese el compromiso con el Archivo de Bogotá.

Este es el cuadro de la planificación de mis actividades:

Desarrollador: María Cristina Herrera Calderón Fecha: 26/02/2007

ACTIVIDAD	FRECUENCIA	TIEMPO (Minutos)
Presentar la propuesta del TID a la universidad	1 día	60 min.
Elaborar el anteproyecto del TID	Lunes a Viernes (durante 10 días)	2400 min.
Proceso de investigación sobre Cuadro de Mando Integral y desarrollo de aplicaciones Web	Lunes a Domingo (durante 31 días)	14880 min.
Presentación sobre Cuadro de Mando Integral al Archivo de Bogotá	1 día	120 min.
Levantamiento de requerimientos para el desarrollo de la aplicación Web	Lunes a Viernes (durante 5 días)	1200 min.
Investigación y decisión sobre la metodología de desarrollo de software a utilizar en el proyecto	Lunes a Domingo (durante 7 días)	3360 min.
Fase de diseño de la base de datos	Lunes a Domingo (durante 9 días)	4320 min.
Fase de diseño de la interfaz gráfica de usuario	Lunes a Domingo (durante 9 días)	4320 min.
Fase de diseño de la aplicación	Lunes a Domingo (durante 9 días)	4320 min.
Validación de restricciones y de la iteración de la aplicación	1 día	240 min.
Fase de desarrollo de la aplicación - Escritura del código	Lunes a Domingo (durante 65 días)	39000 min.
Pruebas individuales de la aplicación	Lunes a Domingo (durante 65 días)	39000 min.
Elaboración del trabajo escrito del TID	Lunes a Domingo (durante 65 días)	7800 min.
Entrega del TID	1 día	

Tabla No. 6 Planeación de Actividades

De igual manera como lo especifica la metodología de desarrollo PSP, sobre cada una de las actividades registre unos tiempos, quedando como resultado la siguiente tabla:

Desarrollador: María Cristina Herrera Calderón Fecha: 05/03/2007

FECHA	COMIENZO	FIN	TIEMPO DE INTERRUPCIÓN	ΔTIEMPO	ACTIVIDAD	COMENTARIOS	C	U
26/02/07	10:00	11:00		60	Propuesta TID		x	1
05/03/07 a 16/03/07	8:00	12:00	50+50+50+50	2200	Anteproyecto		x	1
17/03/07 a 16/04/07	8:00	12:00	360+360+360+360	13440	Balanced Scorecard		x	1
17/03/07 a 16/04/07	2:00	6:00	360+360+360+360	13440	Desarrollo aplicaciones Web		x	1
17/04/07	9:00	11:00		120	Presentación AB		x	1
17/04/07 a 23/04/07	2:00	6:00		1200	Levantar requerimiento		x	1
23/04/07 a 29/04/07	8:00	6:00	60+60	3360	Metodología de desarrollo		x	1
30/04/07 a 08/05/07	8:00	6:00	60+60	4320	Diseño base de datos		x	1
09/05/07 a 17/05/07	8:00	6:00	60+60	4320	Diseño interfaz gráfica		x	1
18/05/07 a 26/05/07	8:00	6:00	60+60	4320	Diseño de la aplicación		x	1
28/05/07	8:00	12:00		240	Validación		x	1
28/05/07 a 31/07/07	7:00	7:00	60+60+60+60	31200	Escritura del código	Interrupciones: descanso y trabajo escrito	x	1
28/05/07 a 31/07/07	7:00	7:00	60+60+60+60	31200	Pruebas de la aplicación	Interrupciones: descanso y trabajo escrito. Las pruebas se realizan paralelamente a la escritura del código	x	1
28/05/07 a 31/07/07	5:00	7:00		7800	Trabajo escrito		x	1
13/08/07					Entrega del TID		x	1

Tabla No. 7 Registro de Tiempos

Y, genere un resumen semanal de actividades:

Desarrollador: María Cristina Herrera Calderón Fecha: 05/03/2007- 31/07/2007

Tarea	Levantar requerimientos	Metodología de desarrollo	Diseño base de datos	Diseño interfaz gráfica	Diseño de la aplicación	Validación	Escritura del código	Pruebas de la aplicación	Trabajo escrito	Total en minuto
Fecha	17/04/07 a 23/04/07	23/04/07 a 29/04/07	30/04/07 a 08/05/07	09/05/07 a 17/05/07	18/05/07 a 26/05/07	28/05/07	28/05/07 a 31/07/07	28/05/07 a 31/07/07	28/05/07 a 31/07/07	
Dom		600	480	480	480		480	480	120	3120
Lun	240	240	480	480	480	240	480	480	120	3240
Mar	240	480	480	480	480		480	480	120	3240
Mier	240	480	480	480	480		480	480	120	3240
Jue	240	480	480	480	480		480	480	120	3240
Vie	240	480	480	480	480		480	480	120	3240
Sab		600	480	480	480		480	480	120	3120
Totales	**1200**	**3360**	**3360**	**3360**	**3360**	**240**	**3360**	**3360**	**840**	**22440**

Tiempos y medias del período *Número de semanas: 15*

Resumen de las semanas anteriores

Totales			**960**	**960**	**960**		**27840**	**27840**	**6960**	65520
Media										0
Máximo			960	960	960		26880	26880	6720	63360
Mínimo							960	960	240	2160

Resumen incluyendo la última semana

Totales	**1200**	**3360**	**4320**	**4320**	**4320**	**240**	**31200**	**31200**	**7800**	87960
Media	80	224	288	288	288	16	2080	2080	520	5864
Máximo	1200	3360	3360	3360	3360	240	26880	26880	6720	75360
Mínimo	0	0	0	0	0	0	960	960	240	2160

Tabla No. 8 Resumen Semanal de Actividades

Mientras duro este proceso de desarrollo, tuve constantes reuniones de asesoría tanto con el profesor de la universidad como con el Ingeniero de Sistemas del Archivo de Bogotá, con el fin de llevar un control del trabajo que se iba desarrollando, generé un documento de gestión de programaciones donde ubique unos puntos de control.[14] Aunque hubo desviaciones entre lo planeado y lo

[14] Ver Anexo B

ejecutado, la situación nunca llegó a ser crítica, los pequeños errores se iban corrigiendo en el camino.

2. Etapa de Análisis y Diseño

2.1 Levantamiento de Información

Ya, en la parte de diseño, se siguieron algunos pasos de la metodología investigativa en cuanto al levantamiento de información:

Se recogió información primaria:

- Investigar sobre el Archivo de Bogotá.
  Método de recolección: Reuniones periódicas con el asesor designado por el Archivo de Bogotá
- Consulta con expertos sobre los temas tratados, tanto en Balanced Scorecard como en programación orientada a objetos

Se recogió información secundaria:

- Investigación conceptual sobre los temas a tratar en el proyecto: Cuadro de Mando Integral (BSC), indicadores de gestión, control de gestión y gerencia de procesos.
- Adquirir conocimientos en el manejo de las herramientas empleadas para el desarrollo de la aplicación Web: Java, Java Server Pages (JSP), JDeveloper, Oracle XE.

Análisis de la información:

A medida que se fue levantando la información se realizaba reuniones periódicas con los colaboradores del Archivo de Bogotá y se validaban o rechazaban los datos que surgían de la investigación.

De igual manera se definieron los requerimientos funcionales y no funcionales de la aplicación; los requerimientos funcionales están plasmados en el documento y los diagramas de caso de uso.

Ambiente de implementación y restricciones consideradas (requerimientos no funcionales):

Como lenguaje de programación se utilizará Java 1.5, J2EE, ya que es una aplicación Web se manejarán Java Server Pages (JSP) y HTML 4.0, como herramienta de desarrollo se utilizará JDeveloper 10g versión 10.1.3.1 de Oracle; la base de datos con la que se comunicará la aplicación será Oracle y en la fase

2.2 Levantamiento de los Casos de Uso

Requerimientos funcionales - Casos de Uso

Los diagramas de caso de uso me permitieron identificar los requerimientos del usuario con respecto al sistema, este es el documento y los diagramas que generé:

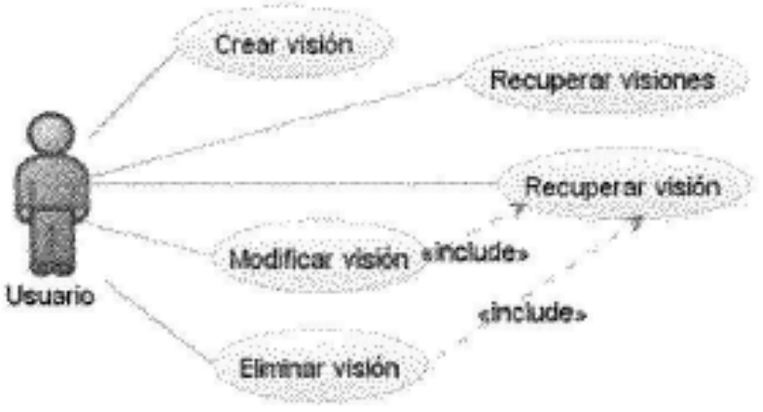


Figura No. 16 Caso de Uso Visión

Identificador y nombre	CU-01 Crear visión
Resumen	Ingresar un nuevo registro visión
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre y descripción para ingresar los datos El usuario ingresa el nombre y la descripción de la visión El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem visión
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-02 Modificar visión
Resumen	Modificar un registro visión
Esfuerzo	Dificil

Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre y la descripción de la visión El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro visión
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-03 Eliminar Visión
Resumen	Eliminar un registro visión
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro visión
Postcondiciones	El sistema elimina el registro
Notas y preguntas	

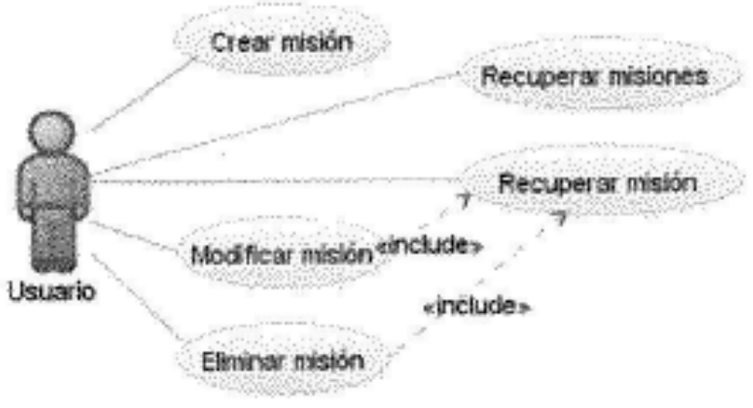


Figura No. 17 Caso de Uso Misión

Identificador y nombre	CU-04 Crear Misión
Resumen	Ingresar un nuevo registro misión
Esfuerzo	Medio

Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre y descripción de la misión para ingresar los datos El usuario ingresa el nombre y la descripción de la misión El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem misión
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-05 Modificar misión
Resumen	Modificar un registro misión
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre y la descripción de la misión El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro misión
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-06 Eliminar Misión
Resumen	Eliminar un registro misión
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro misión
Postcondiciones	El sistema elimina el registro
Notas y preguntas	

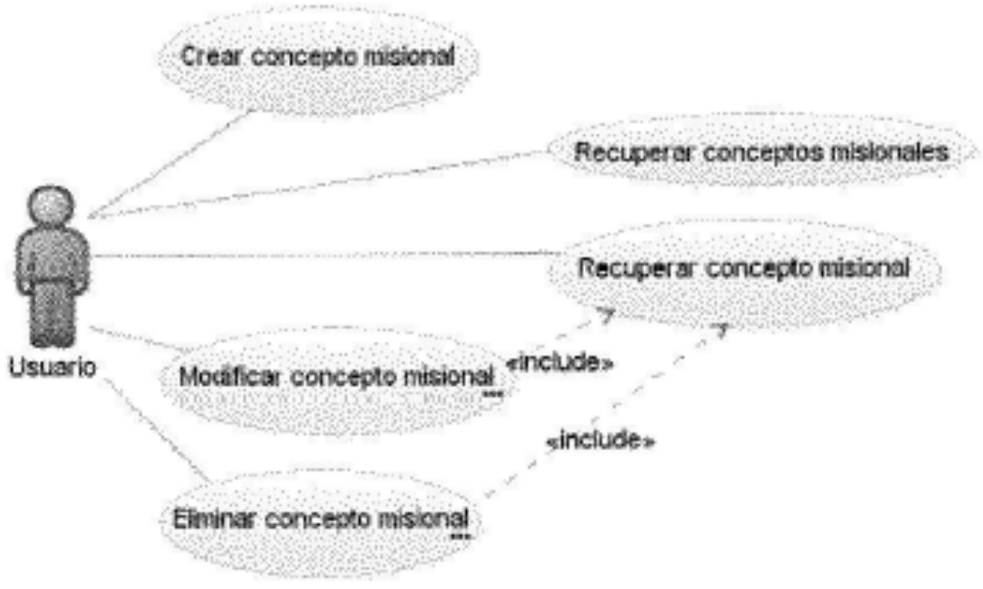


Figura No. 18 Caso de Uso Conceptos Misionales

Identificador y nombre	CU-07 Crear Conceptos misionales
Resumen	Ingresar un nuevo registro conceptos misionales
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre y la descripción del concepto misional para ingresar los datos El usuario ingresa el nombre y la descripción del concepto misional El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem de conceptos misionales
Postcondiciones	El sistema agrega el nuevo registro El sistema crea la perspectiva de Procesos Internos, cuando el Concepto Misional creado es el primero
Notas y preguntas	

Identificador y nombre	CU-08 Modificar Conceptos misionales
Resumen	Modificar un registro concepto misional
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro

	El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre y la descripción del concepto misional El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro concepto misional
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-09 Eliminar Conceptos Misionales
Resumen	Eliminar un registro concepto misional
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro concepto misional
Postcondiciones	El sistema elimina el registro El sistema elimina la perspectiva de Procesos Internos, cuando el Concepto Misional eliminado es el único en la tabla
Notas y preguntas	

Identificador y nombre	CU-10 Crear perspectivas
Resumen	Habilitar el registro de la perspectiva de Procesos Internos
Esfuerzo	Medio
Prioridad	5
Rol	Sistema
Frecuencia de uso	Bajo
Flujo de eventos	El sistema crea la perspectiva
Flujo alterno	
Precondiciones	Debe existir un registro Concepto Misional
Postcondiciones	El sistema habilita el ítem Objetivo
Notas y preguntas	

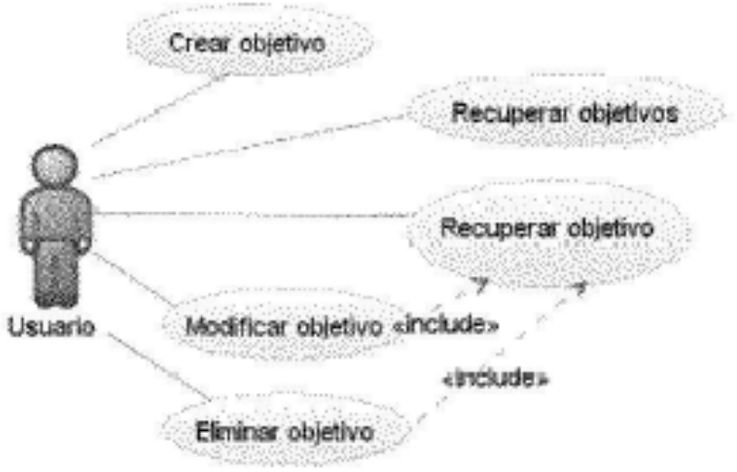


Figura No. 19 Caso de Uso Objetivo

Identificador y nombre	CU-11 Crear objetivo
Resumen	Ingresar un nuevo registro objetivo
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre, descripción, concepto misional y perspectiva para ingresar los datos El usuario ingresa el nombre, la descripción, el concepto misional y la perspectiva del objetivo El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem objetivo
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-11 Modificar Objetivo
Resumen	Modificar un registro objetivo
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre, la descripción, el concepto misional y la perspectiva del objetivo

	El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro objetivo
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-12 Eliminar Objetivo
Resumen	Eliminar un registro objetivo
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro objetivo
Postcondiciones	El sistema elimina el registro
Notas y preguntas	

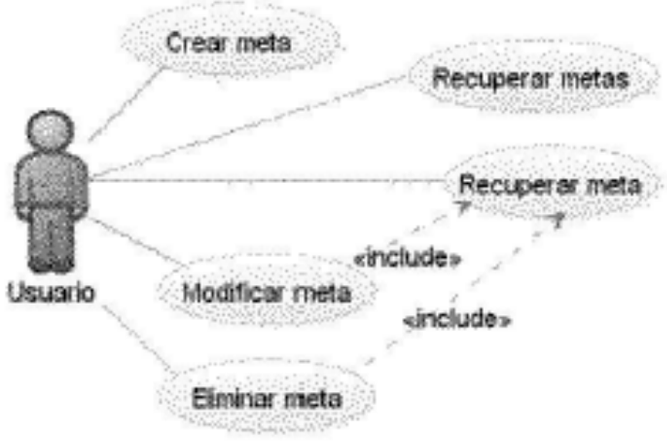


Figura No. 20 Caso de Uso Meta

Identificador y nombre	CU-13 Crear meta
Resumen	Ingresar un nuevo registro meta
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos:

	nombre, descripción y objetivo para ingresar los datos El usuario ingresa el nombre, la descripción y el objetivo de la meta El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem meta
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-14 Modificar Meta
Resumen	Modificar un registro Meta
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre, la descripción y el objetivo de la meta El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro meta
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-15 Eliminar Meta
Resumen	Eliminar un registro meta
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro meta
Postcondiciones	El sistema elimina el registro
Notas y preguntas	

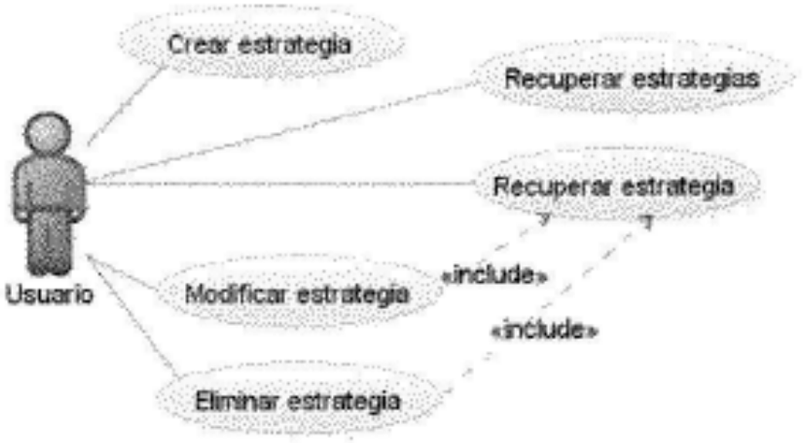


Figura No. 21 Caso de Uso Estrategia

Identificador y nombre	CU-16 Crear estrategia
Resumen	Ingresar un nuevo registro estrategia
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre y descripción para ingresar los datos El usuario ingresa el nombre y la descripción de la estrategia El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el ítem estrategia
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-17 Modificar Estrategia
Resumen	Modificar un registro Estrategia
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre y la descripción de la estrategia El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro estrategia

Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-18 Eliminar estrategia
Resumen	Eliminar un registro estrategia
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro estrategia
Postcondiciones	El sistema elimina el registro
Notas y preguntas	

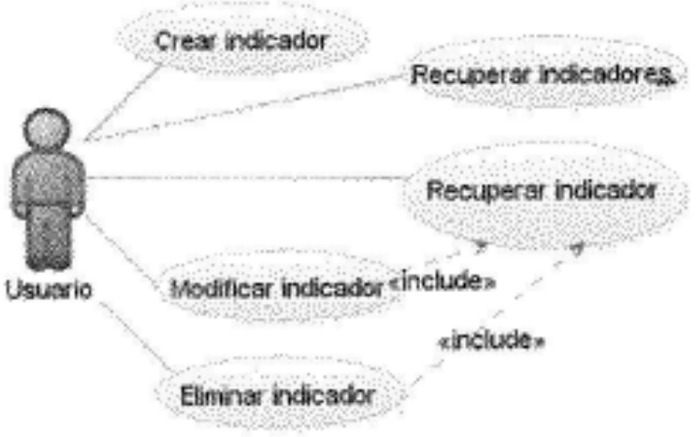


Figura No. 22 Caso de Uso Indicador

Identificador y nombre	CU-19 Crear indicador
Resumen	Ingresar un nuevo registro indicador
Esfuerzo	Medio
Prioridad	5
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona nuevo registro El sistema despliega la pantalla con los siguientes campos: nombre, descripción, unidad de medida, descripción operacional, tipo, periodo de medición, fecha a partir de la cual es valido, meta, estrategia, objetivo, perspectiva, conveniencia, valor umbral, valores para el rango de gestión para ingresar los datos

	El usuario ingresa el nombre, la descripción, la unidad de medida, la descripción operacional, el tipo, el periodo de medición, la fecha a partir de la cual es valido, la meta, la estrategia, el objetivo, la perspectiva, la conveniencia, el valor umbral y los valores para el rango de gestión del indicador El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona el item indicador
Postcondiciones	El sistema agrega el nuevo registro
Notas y preguntas	

Identificador y nombre	CU-20 Modificar Indicador
Resumen	Modificar un registro Indicador
Esfuerzo	Difícil
Prioridad	4
Rol	Usuario
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona un registro El sistema despliega la pantalla con los datos del registro seleccionado El usuario ingresa el nombre, la descripción, la unidad de medida, la descripción operacional, el tipo, el periodo de medición, la fecha a partir de la cual es valido, la meta, la estrategia, el objetivo, la perspectiva, la conveniencia, el valor umbral y los valores para el rango de gestión del indicador El usuario selecciona la opción Guardar
Flujo alterno	Si el usuario no ingresa alguno de los datos requeridos el sistema despliega un mensaje de error
Precondiciones	El usuario selecciona un registro indicador
Postcondiciones	El sistema actualiza el registro
Notas y preguntas	

Identificador y nombre	CU-21 Eliminar indicador
Resumen	Eliminar un registro indicador
Esfuerzo	Bajo
Prioridad	3
Rol	Usuario
Frecuencia de uso	Medio
Flujo de eventos	El usuario selecciona el registro que desea eliminar El sistema despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Eliminar
Flujo alterno	
Precondiciones	El usuario selecciona un registro indicador

Postcondiciones	El sistema elimina el registro
Notas y preguntas	

Identificador y nombre	CU-22 Recuperar registros
Resumen	Recupera el registro seleccionado para modificar o eliminar
Esfuerzo	Alto
Prioridad	5
Rol	Sistema
Frecuencia de uso	Alto
Flujo de eventos	El usuario selecciona el registro que desea modificar o eliminar El sistema realiza la secuencia indicada contra la base de datos y despliega la pantalla con los datos del registro seleccionado El usuario selecciona la opción Guardar o Eliminar
Flujo alterno	No se puede realizar la acción contra la base de datos Se presentan errores de programación en tiempo de ejecución
Precondiciones	El usuario selecciona un registro El sistema crea la conexión a la base de datos
Postcondiciones	El sistema se desconecta de la base de datos
Notas y preguntas	

Los casos de uso que se levantaron para este proyecto sirvieron de guía concreta para el desarrollo de la aplicación.

2.3 Diseño Interfaz Gráfica de Usuario

En esta misma etapa de diseño se propuso lo que sería la interfaz gráfica de usuario, la cual inicialmente se realizó en hojas de papel y cuando se diseño sufrió muy pocos cambios con respecto a lo que el usuario requería; aquí un bosquejo de las pantallas; en esta parte del proyecto se definió que como sólo se levantaron requerimientos funcionales mas no estéticos la aplicación no contaría en este alcance con un diseño visiblemente atractivo, además de adelantarnos a los futuros cambios que sufra la aplicación:

Visión y Misión

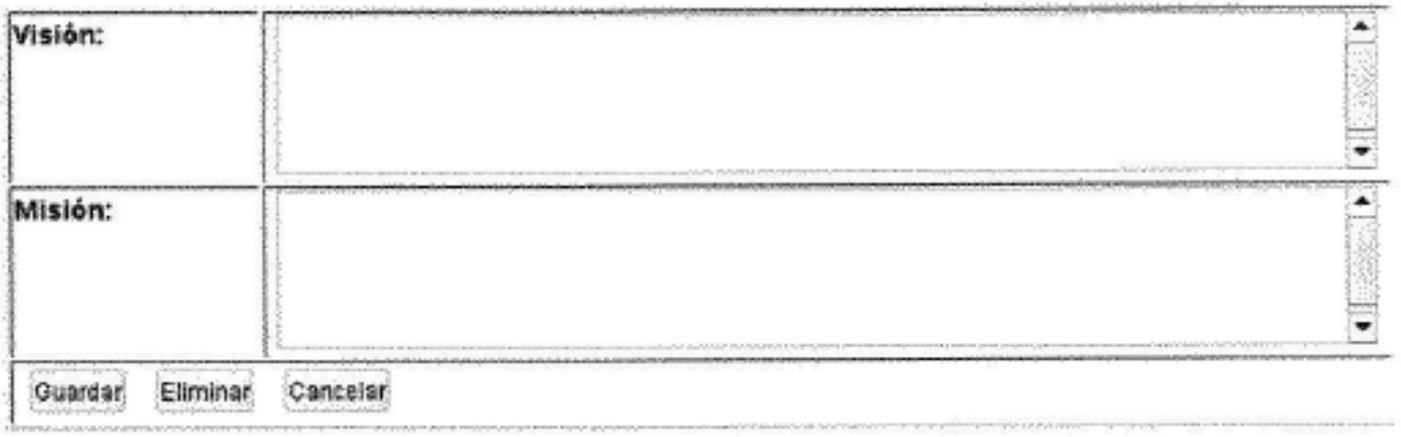


Figura No. 23 Pantalla Visión y Misión

Conceptos Misionales

Figura No. 24 Pantalla Conceptos Misionales

Objetivos

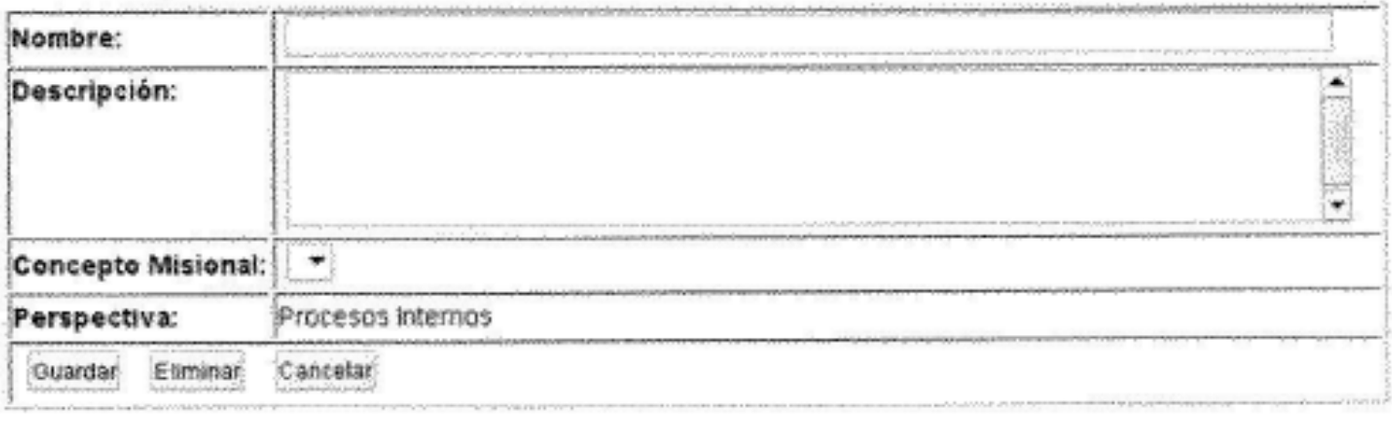


Figura No. 25 Pantalla Objetivos

Metas

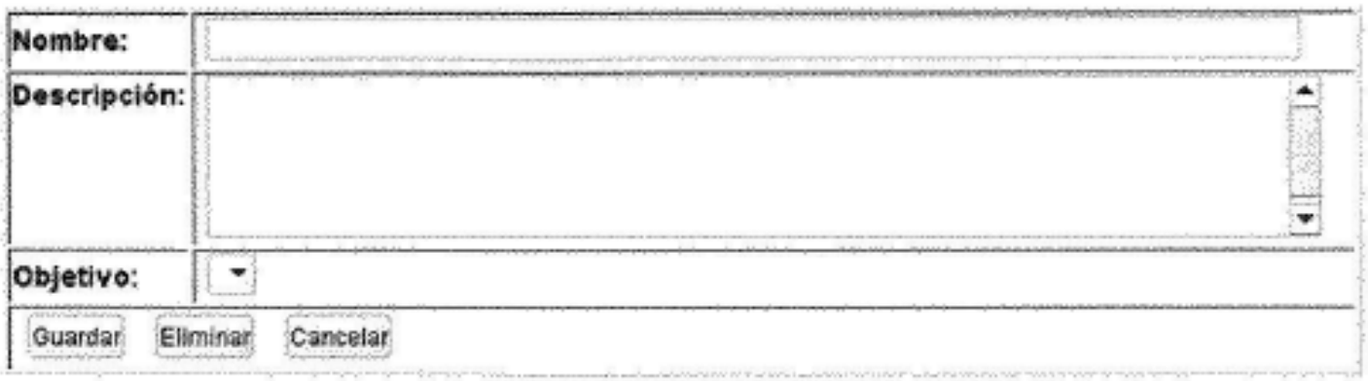


Figura No. 26 Pantalla Metas

Estrategias

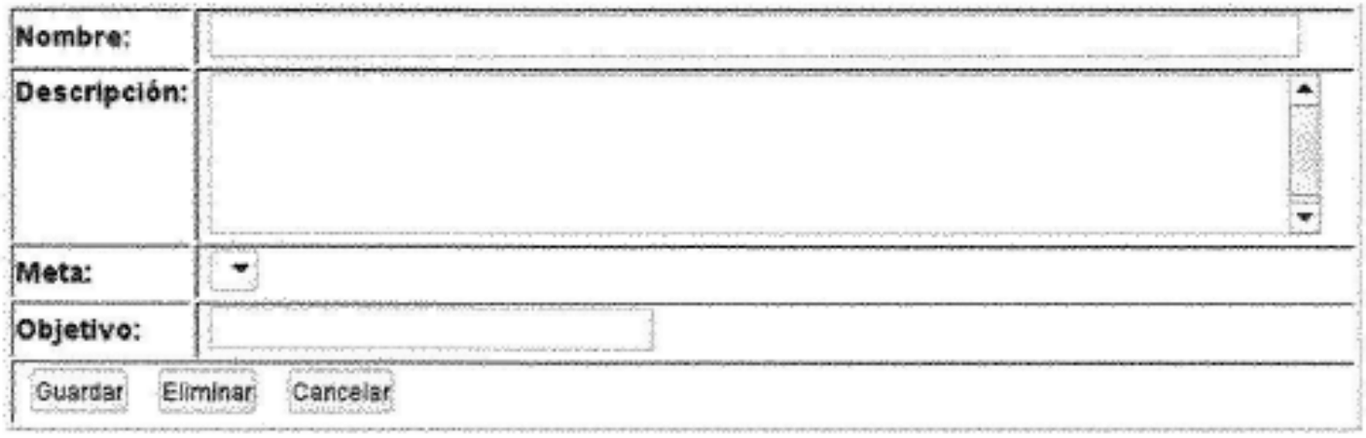
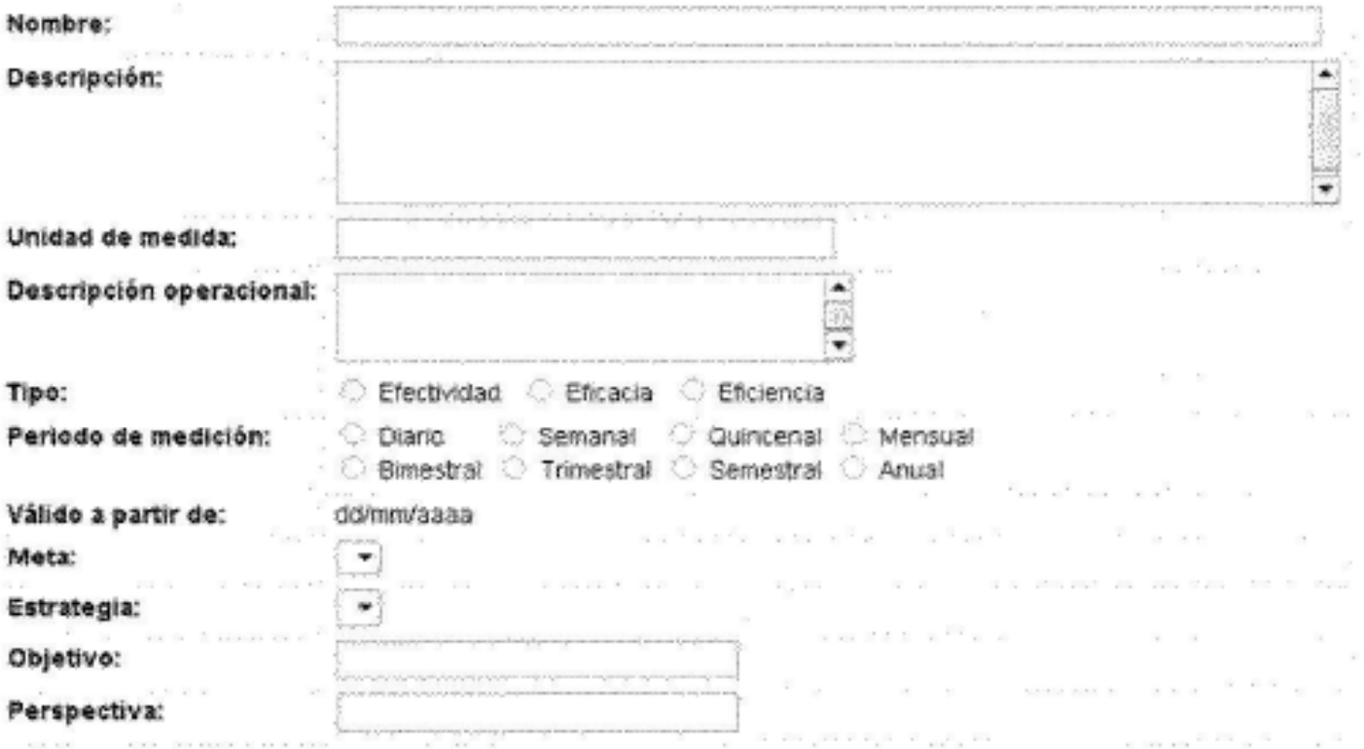


Figura No. 27 Pantalla Estrategias

Indicadores



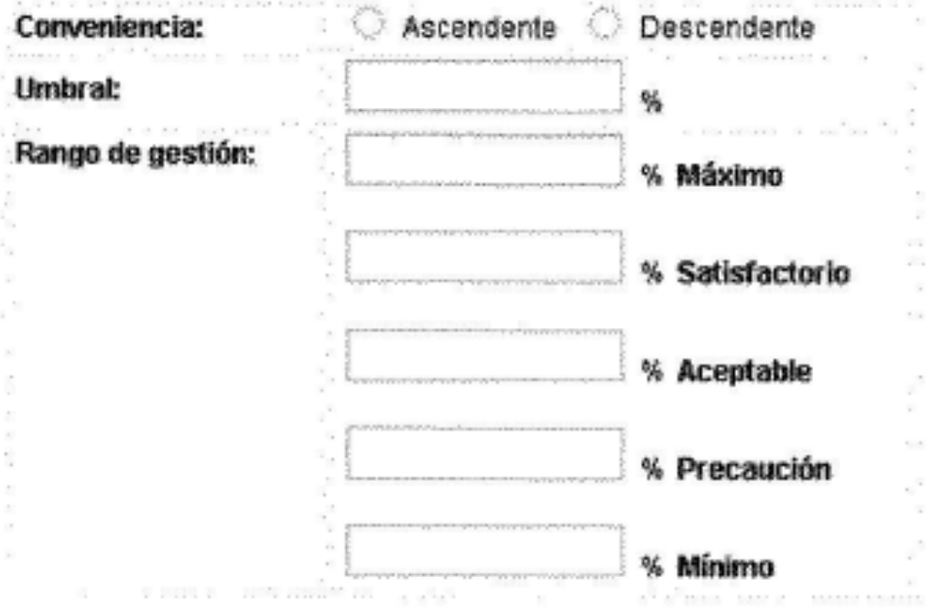


Figura No. 28 Pantalla Indicadores

2.4 Diagramas de UML

Una vez levantados los requerimientos funcionales (casos de uso) para la aplicación se diseñaron los diagramas de clases, componentes, despliegue y secuencias, propios de la metodología RUP.

Los diagramas de clase, modelan las clases o "bloques de construcción" que se utilizan en un sistema, agrupan un conjunto de características comunes de un objeto. (Ejemplo, *Persona*)
Para mi caso en este proyecto, generé un diagrama de clases inicialmente sólo identificando las clases que conformarían el sistema, luego, generé un diagrama con estas clases y sus respectivos atributos:

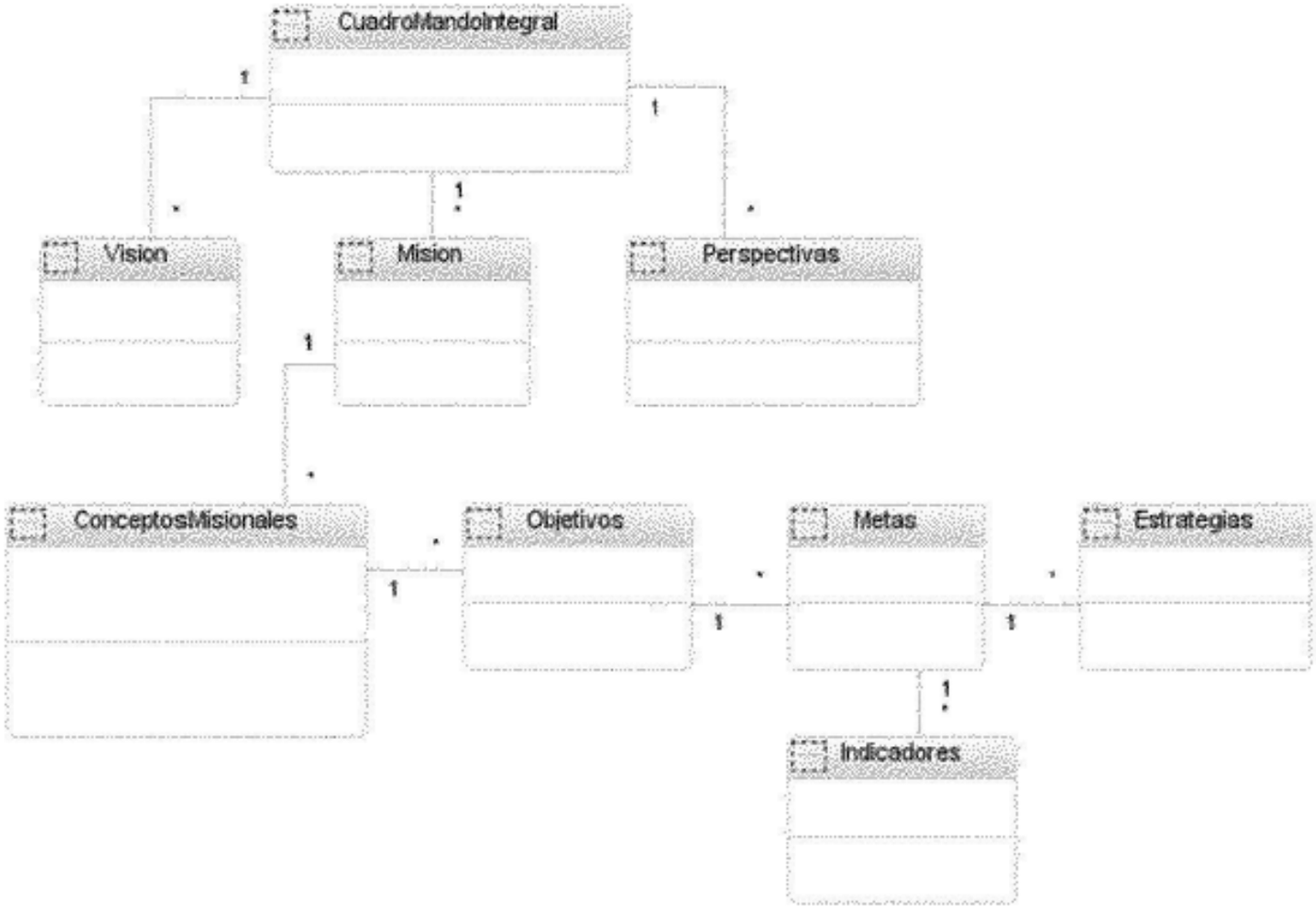


Figura No. 29 Diagrama General de Clases CMI

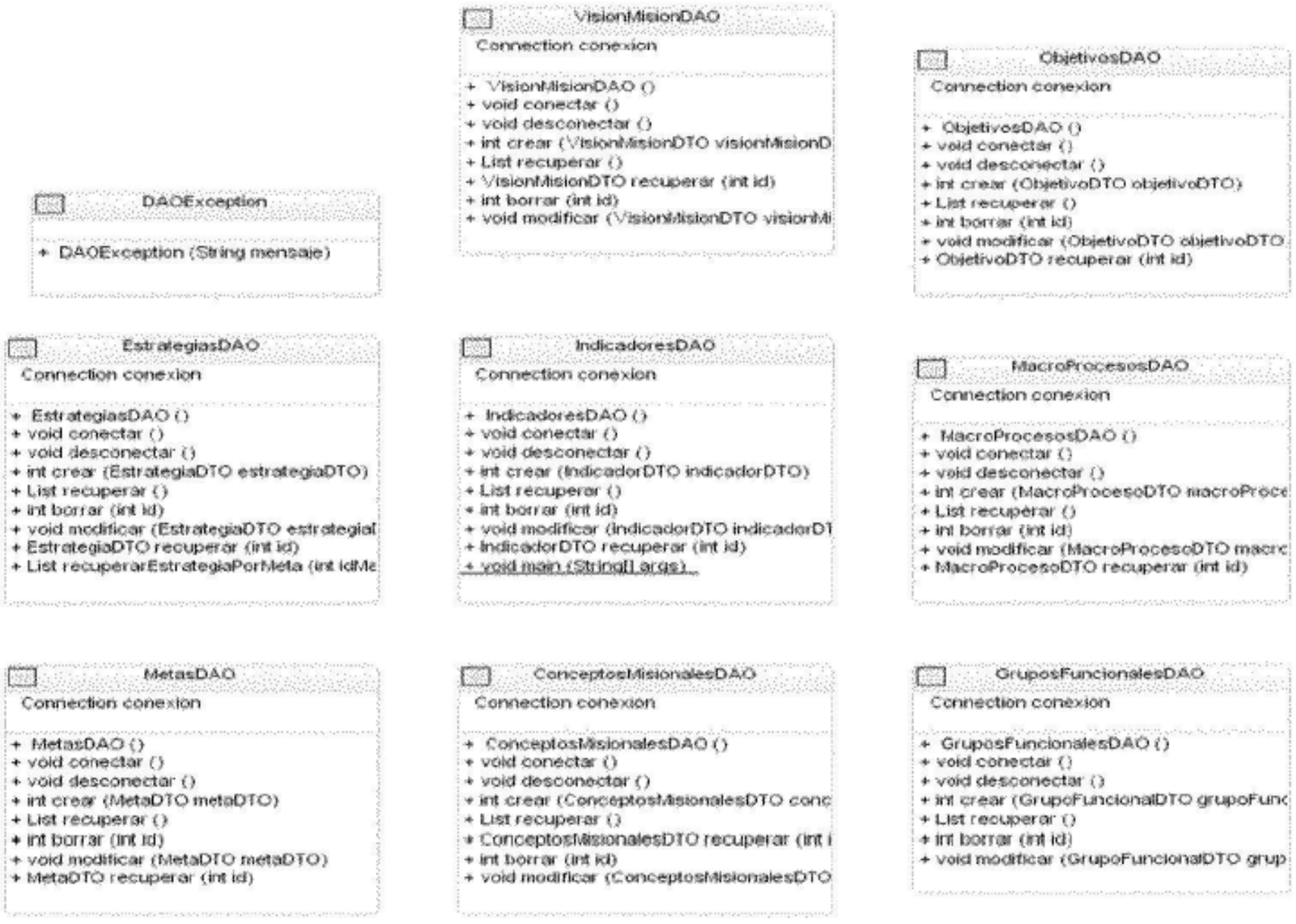


Figura No. 30 Diagrama de las Clases DAO

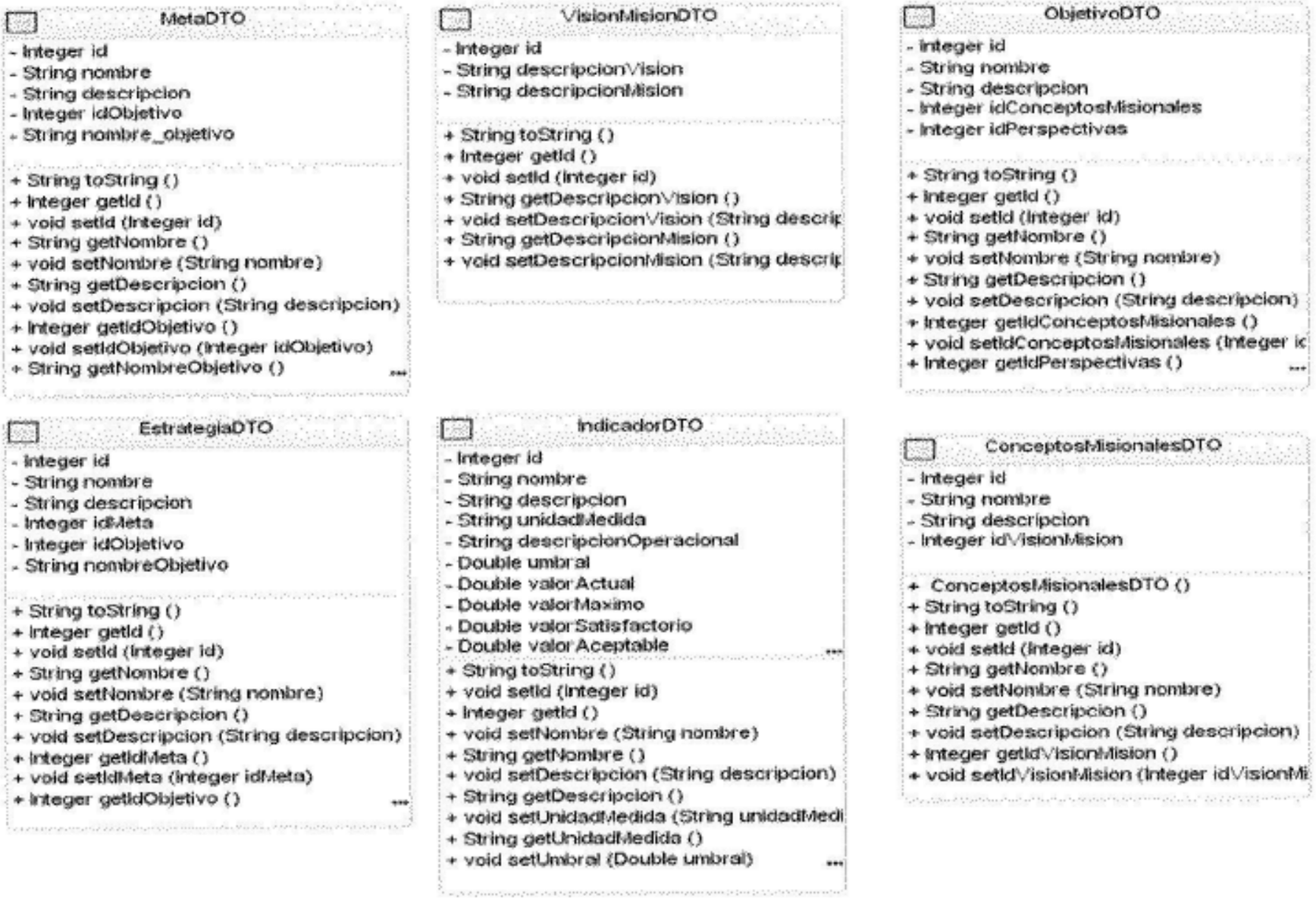


Figura No. 31 Diagrama de las Clases DTO

Los diagramas de componentes y despliegue, permiten visualizar los elementos que utiliza la aplicación para su funcionalidad y la comunicación entre éstos:

DIAGRAMA DE COMPONENTES

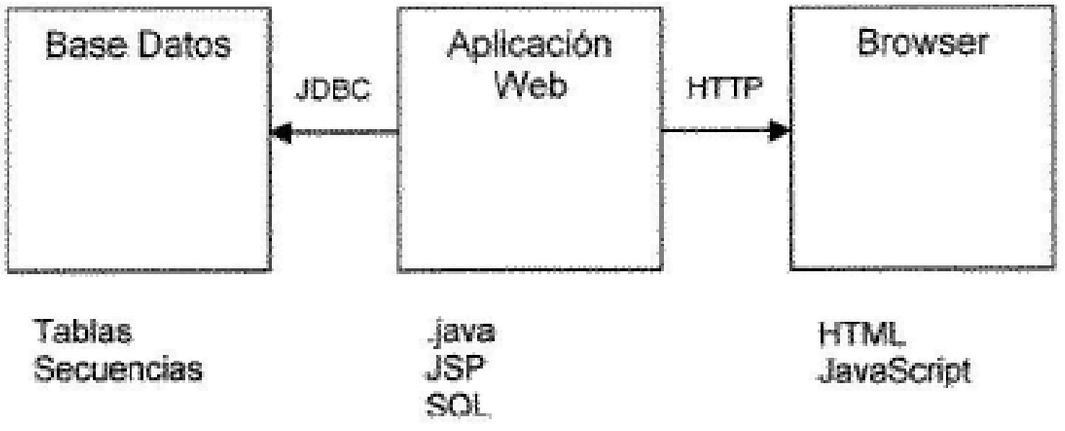


Figura No. 32 Diagrama de Componentes

DIAGRAMA DE DESPLIEGUE

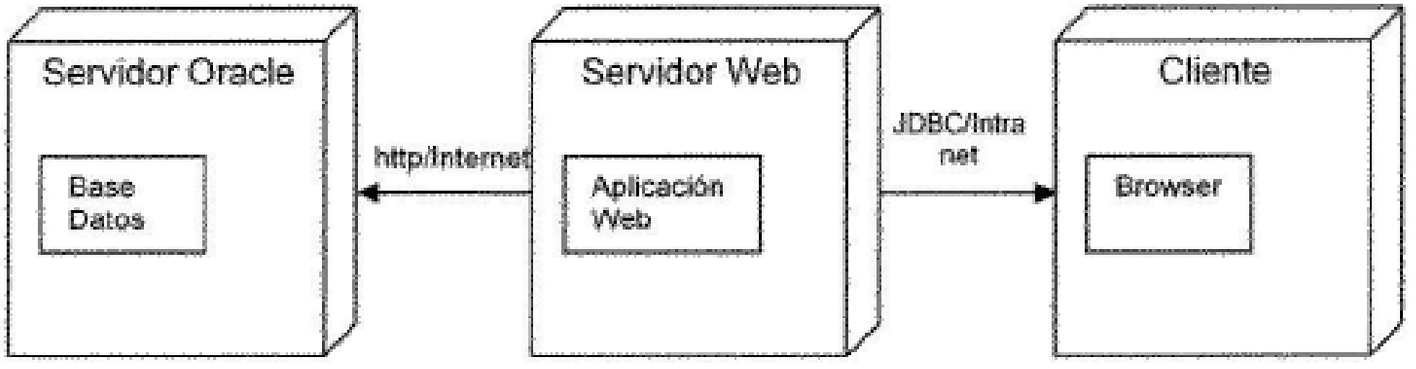


Figura No. 33 Diagrama de Despliegue

Esta es la secuencia de actividades de algunas clases de la aplicación:

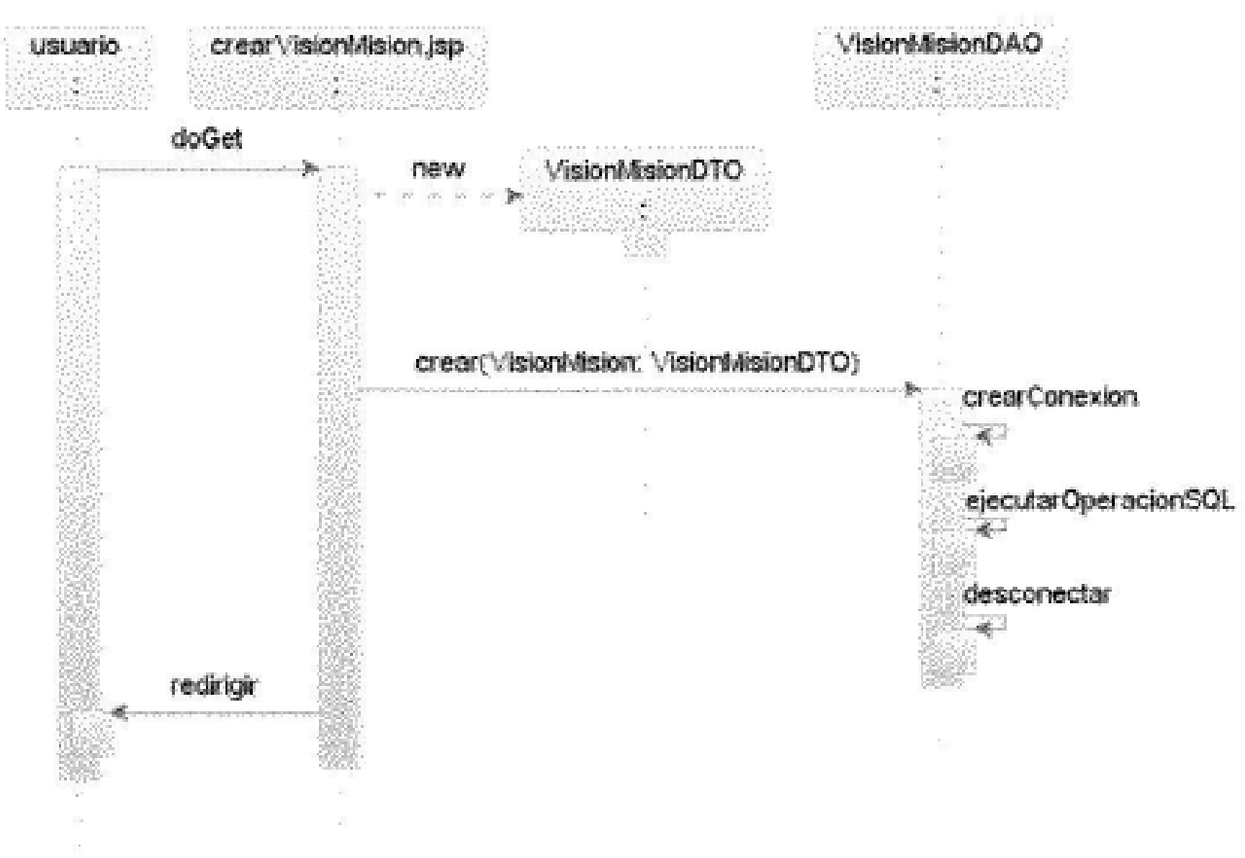


Figura No. 34 Diagrama de Secuencia crearVisionMision

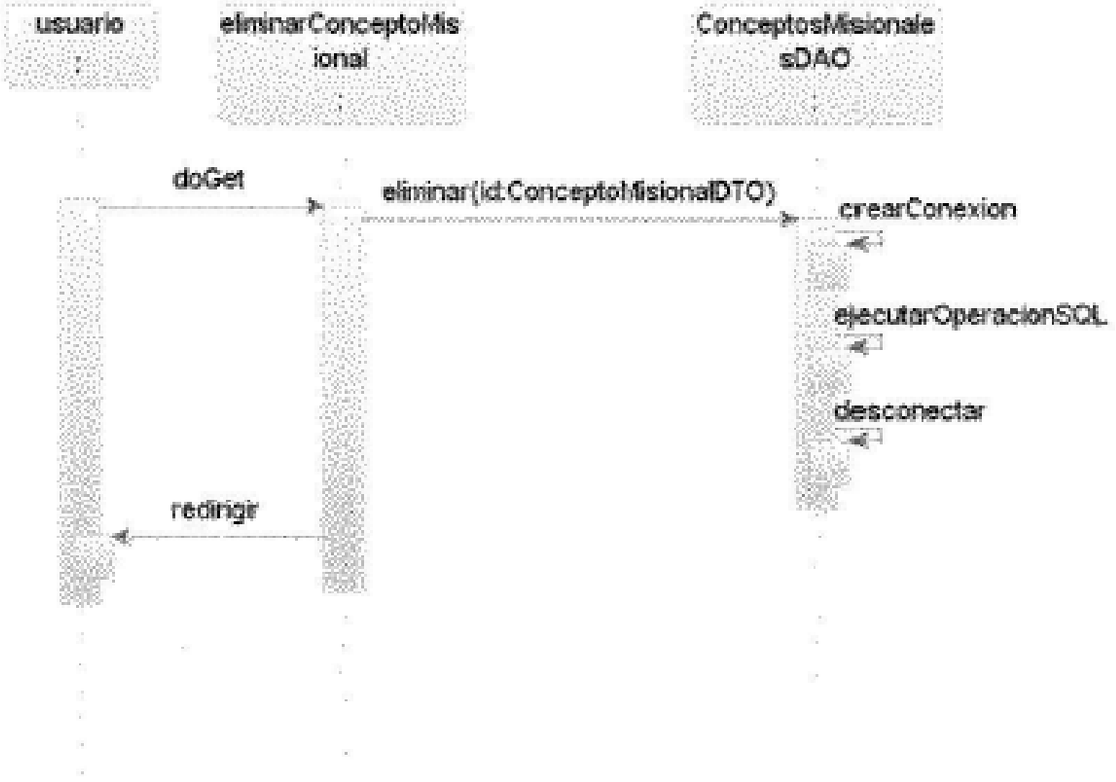


Figura No. 35 Diagrama de Secuencia eliminarConceptoMisional

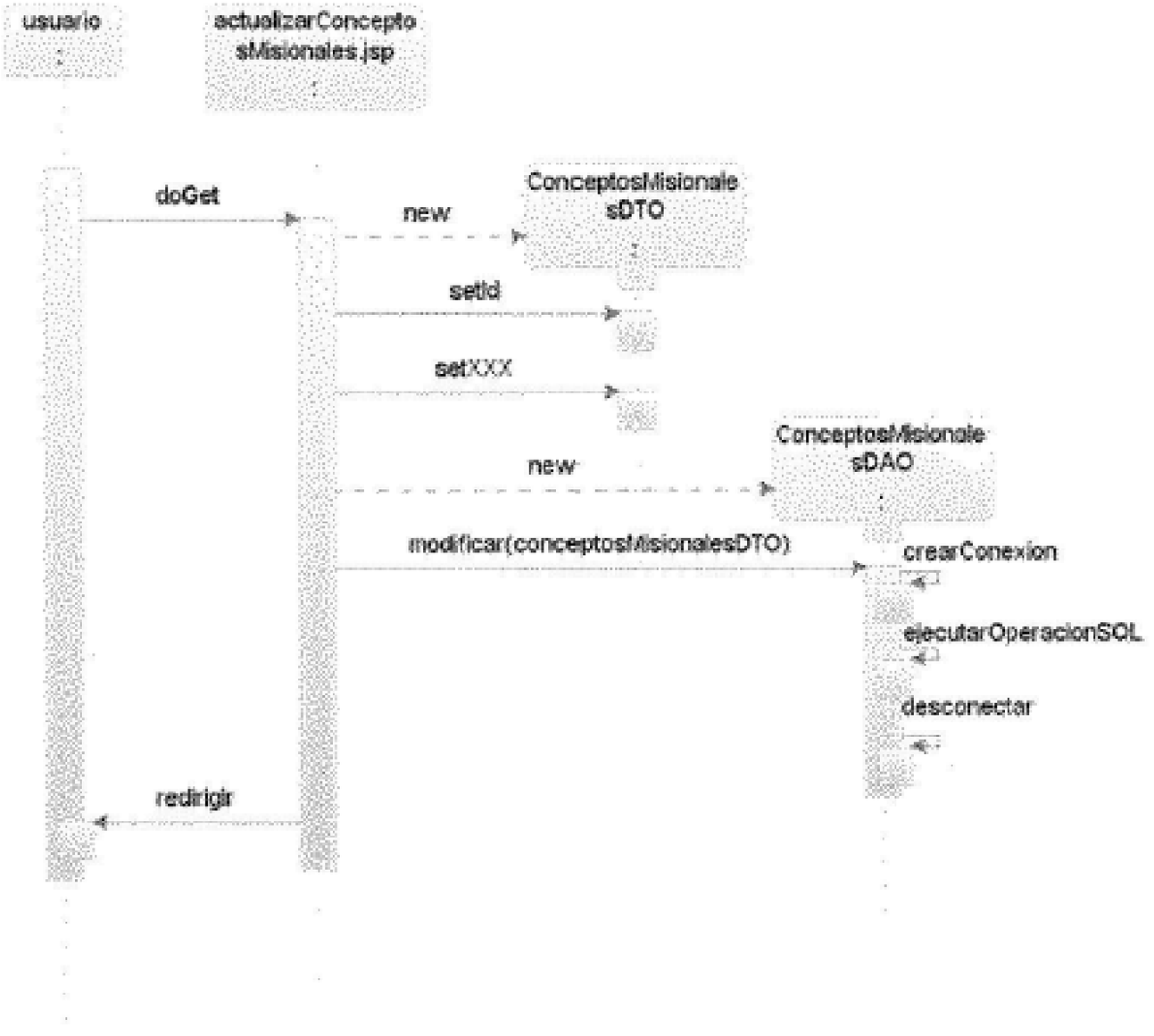


Figura No. 36 Diagrama de Secuencia actualizarConceptosMisionales

3. Etapa de Desarrollo y Pruebas

Paralelo al desarrollo (codificación) de la aplicación, se hicieron pruebas unitarias de cada una de las clases desarrolladas, estas pruebas se hicieron con el usuario del sistema (para este caso, el Ingeniero de Sistemas del Archivo de Bogotá); para probar el sistema se desarrollaron unos módulos de prueba que en el código denomine: "model.prueba"; de igual manera cada vez que se probaba una clase a medida que avanzaba el desarrollo se hicieron pruebas integrales de la aplicación, los errores que iban apareciendo, se iban corrigiendo, lo que significa que no se iniciaba el desarrollo de otra clase si la anterior no estaba funcionando correctamente. Las pruebas contra la base de datos se realizaron en una versión de escritorio de Oracle (XE). A continuación uno de los módulos prueba de la clase IndicadoresDAO:

```
package model.prueba;

import java.util.Iterator;
import java.util.List;
import model.dao.DAOException;
import model.dao.IndicadoresDAO;
import model.dto.IndicadorDTO;

public class MainIndicadores {
   public void probarCrear() {
      IndicadorDTO indicadorDTO = new IndicadorDTO();
      indicadorDTO.setId();
      indicadorDTO.setNombre("");
      indicadorDTO.setDescripcion("");
      indicadorDTO.setUnidadMedida("");
      indicadorDTO.setDescripcionOperacional("");
      indicadorDTO.setIdTipo();
      indicadorDTO.setIdPeriodoMedicion();
      indicadorDTO.setUmbral();
      indicadorDTO.setValorActual();
      indicadorDTO.setIdConveniencia();
      indicadorDTO.setValorMaximo();
      indicadorDTO.setValorSatisfactorio();
      indicadorDTO.setValorAceptable();
      indicadorDTO.setValorPrecaucion();
      indicadorDTO.setValorMinimo();
      indicadorDTO.setResponsable("María Cristina Herrera Calderón");
      indicadorDTO.setIdMeta();

      IndicadoresDAO indicadoresDAO = new IndicadoresDAO();
      try {
         indicadoresDAO.crear(indicadorDTO);
      }
      catch (DAOException e) {
         System.out.println("Imposible crear el registro: " + indicadorDTO + " " +
         e.getMessage());
         System.exit(1);
      }
   }

   public void probarRecuperar() {
      IndicadorDTO indicadorDTO = null;
      List indicadores = null;
      IndicadoresDAO indicadoresDAO = new IndicadoresDAO();
```

```
    try {
        indicadores = indicadoresDAO.recuperar();
    }
    catch (DAOException e) {
        System.out.println("Imposible recuperar los indicadores: " +
      e.getMessage());
        System.exit(2);
    }

    Iterator iterador = indicadores.iterator();
    while (iterador.hasNext()) {
        indicadorDTO = (IndicadorDTO) iterador.next();
        System.out.println(indicadorDTO);
    }
}

public void probarBorrar() throws DAOException {
    IndicadoresDAO indicadoresDAO = new IndicadoresDAO();
    //int id = 1;
    indicadoresDAO.borrar(1);
}

public void probarModificar() {
    IndicadorDTO indicadorDTO = new IndicadorDTO();
    indicadorDTO.setId();
    indicadorDTO.setNombre("");
    indicadorDTO.setDescripcion("");
    indicadorDTO.setUnidadMedida("");
    indicadorDTO.setDescripcionOperacional("");
    indicadorDTO.setIdTipo();
    indicadorDTO.setIdPeriodoMedicion();
    indicadorDTO.setUmbral();
    indicadorDTO.setValorActual();
    indicadorDTO.setIdConveniencia();
    indicadorDTO.setValorMaximo();
    indicadorDTO.setValorSatisfactorio();
    indicadorDTO.setValorAceptable();
    indicadorDTO.setValorPrecaucion();
    indicadorDTO.setValorMinimo();
    IndicadorDTO.setResponsable("");
    indicadorDTO.setIdMeta();

    IndicadoresDAO indicadoresDAO = new IndicadoresDAO();
    try {
        indicadoresDAO.modificar(indicadorDTO);
```

```
        }
        catch (DAOException e) {
            System.out.println("Imposible modificar el registro: " + indicadorDTO + " " +
             e.getMessage());
            System.exit(3);
        }
    }

    public static void main(String[] args) throws DAOException {
        MainIndicadores app = new MainIndicadores();

        app.probarCrear();
        //app.probarRecuperar();
        //app.probarBorrar();
        //app.probarRecuperar();
        //app.probarModificar();
        //app.probarRecuperar();
    }


}
```

Existen unos errores que no son fáciles de manejar y se deben mantener bajo control, estos son los errores de programación que se presentan en tiempo de ejecución de la aplicación, para manejar estos errores utilice excepciones.

En Java, pueden declararse nuevos tipos de excepciones extendiendo la clase **Exception.** En la declaración de un método, la palabra reservada **throws** especifica las excepciones que lanza ese método. Esta cláusula aparece después de la lista de parámetros y antes del cuerpo del método. La cláusula contiene una lista separada por comas de las excepciones que lanzará el método, si ocurre un problema cuando éste se ejecute. El manejador de una excepción se indica por las palabras clave **try** y **catch**, seguidas de un bloque de código que maneja la excepción.

Es decir, que en cada uno de los métodos creados para las distintas clases que emplea la aplicación, maneje excepciones para el control de errores en tiempo de ejecución. Este es uno de ellos:

```
    VisionMisionDAO visionMisionDAO = new VisionMisionDAO();
      try {
         visionMisionDAO.crear(visionMisionDTO);
      }
      catch (DAOException e) {
```

```
            out.println("Imposible crear la visión y la misión: " + e.getMessage());
            return;
        }
```

Esta es una excepción manejada al invocar el método crear visionMisionDTO, ejecutado cuando el usuario desea ingresar un nuevo registro visionMision. Cabe resaltar que la clase DAOException se creó dentro del modelo DAO de la aplicación.