Universidad de las Ciencias Informáticas

Facultad 10

Sistema para la selección de portafolios de proyectos informáticos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Javier Ruiz Garcia

Yadira Mesa Valdés

Tutores:

Ing. Yadira Morales Alamo

Ing. Yanko Hernández Valdés

Co-tutor:

Msc. Maikel Leyva Vázquez

Ciudad de La Habana, junio de 2010

"Año 52 de la Revolución"

Declaración de autoría

Declaración de Autoría

Declaramos que somos los únicos autores del trabajo “Sistema para la selección de portafolios de proyectos informáticos” y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2010.

Autores:

_____

Javier Ruiz Garcia

_____

Yadira Mesa Valdés

Tutores:

_____

Ing. Yanko Hernández Valdés

_____

Ing. Yadira Morales Alamo

Co-tutor:

_____

Msc. Maikel Leyva Vázquez

Resumen

Tomando en consideración la necesidad de profundizar y mejorar las técnicas usadas por empresas productoras de software para la selección del grupo de proyectos a desarrollar, se presenta una solución matemática informatizada que suple carencias a este campo y tiene como aporte la inclusión de criterios actuales, múltiples y personalizables para la selección. El resultado se obtiene de la fusión de diferentes materias, entre las que se encuentran los fundamentos matemáticos con las Teorías de Grafos y Combinatoria, el Proceso Unificado de Desarrollo de Software junto al Lenguaje Unificado de Modelado con Visual Paradigm for UML 6.4 en la Ingeniería de Software y en la programación como herramientas constructoras se utilizó el gestor de base de datos MySql diseñándose el modelo relacional en DBDesigner 4 así como la implementación se desarrolló sobre la plataforma java con el Netbeans IDE 6.8 M2. Con la utilización del sistema informático obtenido se mejora la selección de portafolios de proyectos informáticos, garantizando un menor tiempo de aplicación con mayor flexibilidad en la configuración de los criterios de selección.

Palabras Claves

Portafolio de proyectos, selección, criterios, software, programación, modelo matemático.

Índice de Contenidos

Índice de Contenidos

Introducción

La gestión de proyectos es la disciplina que se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo y costo definidos. Según la Guía de los Fundamentos de la Dirección de Proyectos PMBOK, un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. La gestión de proyectos brinda la posibilidad de evaluar y planificar proyectos obteniendo así una estimación del esfuerzo necesario para su realización y de esta manera determinar el tiempo y los recursos a utilizar.

Una empresa desarrolladora de software con ventajas de mercado siempre tiene la posibilidad de escoger los proyectos a realizar. A la hora de la planificación, si la cantidad de proyectos candidatos es mayor al número que puede atender, según su capacidad productiva, es necesario seleccionar un portafolio de proyecto atendiendo a las necesidades de la organización. Según PMBOK un portafolio es un conjunto de proyectos o programas y otros trabajos, que se agrupan para facilitar la gestión efectiva de ese trabajo, a fin de cumplir con los objetivos estratégicos de negocio. Los proyectos o programas del portafolio no necesariamente tienen que ser interdependientes o estar directamente relacionados. La recaudación y el respaldo pueden asignarse sobre la base de categorías de riesgo / recompensa, líneas de negocio específicas o tipos generales de proyectos, como la mejora de la infraestructura y del proceso interno. En otras palabras un portafolio de proyecto es un conjunto de proyectos que, llevados a cabo en un determinado periodo de tiempo, comparten una serie de recursos entre los que pueden existir relaciones de complementariedad, incompatibilidad y sinergias producidas por compartir costes y beneficios derivados de la realización de más de un proyecto a la vez.

La metodología empleada por las organizaciones empresariales para distribuir su presupuesto a la hora de seleccionar qué proyectos deben ser escogidos para ejecutar y cubrir sus necesidades, ha evolucionado mucho desde que dichas organizaciones empezaron a apoyar sus decisiones de selección en alguna solución matemática, desde la década de los 60 se comienza a trabajar en estos fines. Muchas de las soluciones matemáticas desarrolladas constan de un gran cúmulo de cálculos que las hace muy trabajosas para realizarlas manualmente, de aquí que se comenzaran a automatizar computacionalmente estas soluciones. Con el desarrollo y las amplitudes alcanzadas por las empresas en las últimas décadas, se ha creado la alternativa de escoger las mejores combinaciones de proyectos y no proyectos individuales como se hacía inicialmente. Con estos nuevos métodos no se desea solo seleccionar los

mejores proyectos con los recursos disponibles, sino determinar el conjunto que aproveche de mejor manera dichos recursos.

Las organizaciones se enfrentan al problema de cómo invertir sus recursos en las distintas alternativas de proyectos candidatos. La correcta evaluación y posterior selección de proyectos de desarrollo de software brinda ventajas competitivas a las empresas, a la vez impone múltiples retos, entre ellos la dificultad de evaluar los beneficios intangibles y las restricciones impuestas por el entorno.

Aunque en los últimos años la selección de portafolios de proyectos ha tenido algún desarrollo, aún los métodos no son lo suficientemente flexibles como para poder aplicarlos completamente en la UCI pues están basados en criterios fijos, estrictos y poco actuales en mucho de los casos. Actualmente estas selecciones se realizan empíricamente en muchas organizaciones como en la UCI.

Analizando lo antes expuesto se plantea el siguiente problema científico: ¿Cómo realizar la selección de portafolios de proyectos informáticos en la UCI, con mínimo esfuerzo de aplicación?

El problema científico identificado anteriormente se enmarca en el objeto de estudio de la Gestión de proyectos de software teniendo como campo de acción la Gestión de portafolio de proyectos. Con el objetivo general de crear un sistema informático basado en una solución matemática capaz de realizar una selección de un grupo de proyectos informáticos a realizar utilizando criterios actuales, múltiples y personalizables.

Objetivos específicos:

- ✓ Analizar los aspectos teóricos que sustentan el proceso de selección de portafolios de proyectos informáticos para su desarrollo.
- ✓ Elaborar una solución matemática para la selección de portafolios de proyectos que satisfaga las necesidades de las empresas productoras de software, centrándonos en la UCI, y que sea adaptable a diferentes entornos.
- ✓ Desarrollar un sistema que implemente la solución matemática.
- ✓ Realizar pruebas de calidad al sistema obtenido.

Idea a defender: Con la automatización de la selección de portafolios de proyectos informáticos basada en criterios actuales, múltiples y personalizables; se brinda a la UCI una herramienta que ayude a seleccionar los proyectos que se desarrollarán de una manera más eficaz para maximizar sus intereses.

Métodos:

- ✓ Empíricos:
    1. Entrevista: Se utiliza este método para obtener información de cómo se realiza la selección de portafolios de proyectos en la UCI.
- ✓ Teóricos:
    2. Analítico-Sintético: Centrándose en el análisis de las teorías, documentos, y materiales, permite la extracción de los elementos más importantes para realizar el análisis y estudio de los procesos y normas que plantean cómo evaluar cualitativa y cuantitativamente cómo seleccionar un portafolio de proyectos.
    3. Análisis Histórico–Lógico: Permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo. Su objetivo en la investigación es el estudio de cómo ha evolucionado y se ha desarrollado la evaluación cualitativa y cuantitativa de la selección de portafolios de proyectos.
    4. Modelado: Se utiliza este método para representar una posible interfaz de la aplicación a través del diseño planteado para modelar el sistema.

Estructura capitular:

CAPÍTULO 1 TEORÍAS QUE SUSTENTAN LA INVESTIGACIÓN. En este capítulo se expone la fundamentación teórica del tema. Incluye una descripción detallada del objeto de estudio, con la finalidad de comprender el proceso de selección de portafolios de proyectos. Se se hace un resumen de las metodologías, lenguajes, herramientas, tecnologías, arquitecturas y patrones utilizadas en el proceso de desarrollo del software.

CAPÍTULO 2 MODELO MATEMÁTICO. En este capítulo se explica el modelo matemático para la selección de portafolios de proyecto. Además, se muestra un ejemplo de selección de portafolios de proyectos.

CAPÍTULO 3 CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA. En este capítulo se describe el negocio, se enumeran los requisitos del sistema para tener un mejor entendimiento de lo que se quiere concebir. Se definen los actores del sistema, se describen los casos de uso y se presenta el diagrama de casos de uso. Además, se realiza el análisis y el diseño del sistema.

CAPÍTULO 4 IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA. En este capítulo se presentan los diagramas de despliegue y componentes como objetivo primordial de la fase de implementación. Además, se realizan las pruebas del sistema a las cuáles se sometió la aplicación.

CAPÍTULO 1. TEORÍAS QUE SUSTENTAN LA INVESTIGACIÓN

1.1 Introducción

En este capítulo se hace referencia a los principales aspectos teóricos que constituyen la base de la investigación realizada. Se exponen los resultados del estado del arte en el cual se aborda la actualidad y antecedentes del tema. Se muestran las herramientas, tecnologías y metodologías a utilizar, fundamentando su selección en un minucioso estudio.

1.2 Estado del Arte

1.2.1 Conceptos fundamentales

Proyecto: Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. (1)

Temporal: Significa que cada proyecto tiene un comienzo definido y un final definido. El final se alcanza cuando se han logrado los objetivos del proyecto o cuando queda claro que los objetivos del proyecto no serán o no podrán ser alcanzados, o cuando la necesidad del proyecto ya no exista y el proyecto sea cancelado. Temporal no necesariamente significa de corta duración; muchos proyectos duran varios años. En cada caso, sin embargo, la duración de un proyecto es limitada.

- ✓ Productos, servicios o resultados únicos: Un proyecto crea productos entregables únicos. Productos entregables son productos, servicios o resultados.

Dirección de proyecto: La dirección de proyectos es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer los requisitos del proyecto. La dirección de proyectos se logra mediante la aplicación e integración de los procesos de dirección de proyectos de inicio, planificación, ejecución, seguimiento y control, y cierre. El director del proyecto es la persona responsable de alcanzar los objetivos del proyecto. (1)

Portafolio de proyecto: Un portafolio es un conjunto de proyectos o programas y otros trabajos, que se agrupan para facilitar la gestión efectiva de ese trabajo, a fin de cumplir con los objetivos estratégicos del negocio. Los proyectos o programas del portafolio no necesariamente tienen que ser interdependientes o estar directamente relacionados. La recaudación y el respaldo pueden asignarse sobre la base de

categorías de riesgo / recompensa, líneas de negocio específicas o tipos generales de proyectos, como la mejora de la infraestructura y del proceso interno. (1)

1.2.2 Estado Actual

Tradicionalmente, tal y como señalan Moore y Baker (1969), las empresas no utilizaban ninguna técnica específica para seleccionar proyectos, sino que este proceso de toma de decisiones se realizaba de manera subjetiva, recopilando la mayor cantidad de información disponible de cada una de las alternativas (proyectos candidatos), para, a partir de ésta, tomar una decisión, considerándose que no existían modelos que pudiesen resumir o agregar toda esta información y aportar una conclusión relevante. A pesar de que durante muchos años se mantuvo este modo de actuar, la evolución de las organizaciones, con el consiguiente incremento en su tamaño y, por tanto, de sus recursos y necesidades, así como la existencia de un entorno cada vez más competitivo y cambiante, unido al indudable avance de los recursos tecnológicos, requiere plantear una estrategia más racional a la hora de seleccionar proyectos por parte de los empresarios. (2)

La correcta evaluación y posterior selección de proyectos de desarrollo de software brinda ventajas competitivas a las organizaciones. La selección de proyectos en el campo de las tecnologías de la información presenta múltiples retos, entre ellos la dificultad de evaluar los beneficios intangibles, las interdependencias existentes entre proyectos y las restricciones que imponen las organizaciones. (3)

A lo largo de los años, muchos han sido los estudios publicados relacionados con esta problemática de selección de proyectos y portafolios de proyectos, de los que sólo unos pocos presentan una clasificación (categorización) de las técnicas o posibles metodologías utilizadas para dicha selección. Los primeros estudios que con cierto éxito realizaron una clasificación fueron los de Baker y Pound (1964), Baker (1974) y Baker y Freeland (1975). Todos ellos establecen que las técnicas se pueden clasificar fundamentalmente en dos categorías: *técnicas de medidas de beneficio* y *técnicas de selección de proyectos y asignación de recursos*. Como se evidencia en el trabajo *"Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos"* (4) la solución a la selección de portafolios de proyecto reside en la creación de un modelo que unifique las técnicas utilizadas dentro de las clasificaciones antes mencionadas.

Aún en la actualidad la mayoría de los modelos matemáticos existentes son incompletos y carecen de la flexibilidad necesaria para su correcta aplicación en distintos entornos, como en la UCI. Es válido aclarar que se han desarrollado algunos modelos matemáticos que satisfacen las necesidades actuales, pero estos son privados. Entiéndase por privado que son propiedad de empresas determinadas y su uso es exclusivo de las mismas.

La UCI no está exenta de esta situación pues actualmente la selección de los proyectos que se desarrollan en el centro se realiza de manera empírica y sin la ayuda de sistemas decisorios que apoyen sus propuestas en criterios actuales, múltiples y personalizables.

1.3 Metodologías de Desarrollo de Software

Es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla, que explica los pasos necesarios para terminar el proyecto.

1.3.1 Programación Extrema XP (Extreme Programming)

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. De forma que una metodología ágil es la que tiene como principios que (5):

- ✓ Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
- ✓ El software que funciona es más importante que la documentación exhaustiva.
- ✓ La colaboración con el cliente en lugar de la negociación de contratos.
- ✓ La respuesta ante el cambio en lugar de seguir un plan cerrado.

Se puede decir que este movimiento empezó a existir a partir de febrero de 2001, cuando se reunieron los representantes de cada una de estas metodologías y terminaron poniendo en común sus ideas en una declaración conjunta.

La programación extrema es una metodología de desarrollo ligera (ágil) basada en una serie de valores y de buenas prácticas que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación. Una de las características principales de este método de

programación es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha hecho esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, el resultado si es una nueva manera de ver el desarrollo de software. El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos, aplicando el sentido común. (5)

1.3.2 Proceso Unificado de Desarrollo de Software (RUP)

RUP es un proceso para el desarrollo de un proyecto de software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. Tiene 3 características esenciales; está dirigido por Casos de Uso, que orientan el proyecto a la importancia para el usuario y lo que este quiere, está centrado en la arquitectura, que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y es iterativo e incremental, dividiéndose el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada. (6)

Organiza el ciclo de vida de un producto en 4 fases:

Inicio: Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

Construcción: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.

Transición: El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Cada Fase termina en un Hito y puede estar dividida en Iteraciones (6).

Hito: Punto de control. Permiten a la gestión evaluar el progreso.

Iteración: Unidad de desarrollo del producto. (7)

1.3.3 Adaptive Software Development (ASD)

Su característica fundamental es que se adapta al cambio, identificándose por ser iterativo y orientado a los componentes de software más que a las tareas (8).

Entre las ventajas de Adaptive Software Development se encuentran (9):

- ✓ La tercera fase del ciclo de vida, revisión de los componentes, sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
- ✓ Apunta hacia el Rapid Application Development (RAD), el cual enfatiza velocidad de desarrollo para crear un producto de alta calidad, bajo mantenimiento involucrando al usuario lo más posible.
- ✓ Utiliza información disponible acerca de cambios para mejorar el comportamiento del software.

Dado a que es una metodología ágil implica no realizar procesos que son requeridos en las metodologías tradicionales o por lo menos no realizarlos en procesos diferentes, lo cual conlleva que empresas grandes necesiten llevar un mayor control a procesos y personas, tener tareas asignadas a un estado o proceso, y en las cuales dichos procesos no afectan en gran medida al costo final del producto. Para dichas empresas seleccionar una metodología tradicional resulta mucho más rentable por el gran volumen de personal, de productos, y de costos que se manejan y que para los cuales tendrá un mayor control (9).

1.3.4 ¿Por qué RUP?

RUP se basa en la documentación exhaustiva del software y prioriza bastante el análisis y diseño, lo cual permite obtener un producto de mayor calidad y que esté acorde a las especificaciones del cliente, aspectos en los cuales aventaja al XP y al Adaptive Software Development. Además, RUP utiliza y se apoya en el Lenguaje Unificado de Modelado (UML), lenguaje el cual es muy potente y su uso facilita una mejor perspectiva de lo que se desea lograr.

1.4 Lenguaje de Modelado

1.4.1 UML

El lenguaje unificado de modelado UML (*Unified Modeling Language*), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres estudiosos de los más prominentes en la industria de las tecnologías, las metodologías y los sistemas de información: Grady Booch, James Rumbaugh e Ivar Jacobson. El lenguaje ha ganado un significante soporte de la industria de varias organizaciones vía el consorcio de socios de UML y fue presentado al *Object Management Group* (OMG) y aprobado como un estándar el 17 de noviembre de 1997.

UML es un lenguaje visual para especificar, construir y documentar sistemas (10)
Unified (UNIFICADO):

- ✓ El aporte de muchos métodos y notaciones.
- ✓ Independiente de implementaciones, plataformas y lenguajes.

Modeling (MODELADO):

- ✓ Los modelos son utilizados en todas las ingenierías.

Language (LENGUAJE):

- ✓ Las personas para comunicarse, intercambiar ideas, entenderse entre ellas necesitan un lenguaje común.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. Es un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes. (10)

1.4.2 Visual Paradigm

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (11) Herramienta que se encuentra accesible en esta dirección (http://www.visual-paradigm.com/download/).

1.5 Tecnologías y Lenguajes de Programación

1.5.1 Java

Java es una plataforma virtual de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales ("Diferentes plataformas").

La plataforma Java consta de las siguientes partes:

- ✓ El lenguaje de programación mismo.
- ✓ La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.
- ✓ El API Java, una biblioteca estándar para el lenguaje.

Originalmente llamado OAK por los ingenieros de Sun Microsystems, Java fue diseñado para correr en computadoras incrustadas. Sin embargo, en 1995, dada la atención que estaba produciendo la Web, Sun Microsystems la distribuyó para sistemas operativos, tal como Microsoft Windows.

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste. Incorpora sincronización y manejo de tareas en el lenguaje mismo e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado *bytecode*, que luego son interpretados por una máquina virtual (JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda "escribir el programa una vez, y correrlo en cualquier lado". También existen compiladores nativos de Java, tanto software libre como no libre. (12)

1.5.2 ¿Por qué Java?

Se decide optar por Java puesto que se encuentra bajo la licencia GNU GPL (*GNU General Public License*) lo que la convierte en una plataforma libre. Además brinda una excelente portabilidad y compatibilidad entre los distintos sistemas operativos, lo que hace a la herramienta completamente operacional sobre distintas plataformas.

1.5.3 ¿Por qué NetBeans?

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. SunMicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

El entorno integrado de desarrollo (Integrated Development Environment IDE) NetBeans es una herramienta para programadores; pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación, además cuenta con un número importante de módulos para extender sus funcionalidades. Es un producto libre y gratuito sin restricciones de uso, debido a esto se decide utilizar este IDE para el desarrollo de la herramienta. (13)

Se encuentra accesible en la dirección (http://www.dosideas.com/noticias/java/791-netbeans-ide-68-listo-para-descargar.html).

1.6 Arquitectura y patrones

1.6.1 Patrón Modelo Vista Controlador (MVC)

La arquitectura Modelo-Vista-Controlador, es la implementación de la arquitectura en tres capas más extendida (14).

Definición del Patrón

- ✓ Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos.
- ✓ Vista (View): Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario.
- ✓ Controlador (Controller): Reciben las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, entre otras. Los eventos son traducidos a solicitudes de servicio ("servicerequests") para el modelo o la vista. Es el que debe de controlar los eventos.

Responsabilidades de cada parte:

Después de definir los conceptos de Modelo, Vista y Controlador se definirán las responsabilidades de cada uno de ellos.

- ✓ El modelo deberá:
  - Acceder a los datos persistentes y a la capa de almacenamiento de datos. Algo ideal es que este sea independiente de la Base de datos utilizada.
  - Llevar las estadísticas de sistema (en caso de haberlas).
  - Definir las reglas de negocio.
  - Notificar los cambios que se hayan hecho.
- ✓ El controlador deberá:
  - Recibir los eventos de entrada.
- ✓ Las vistas deberán:
  - Conseguir los datos que aporta el modelo y mostrárselos al usuario.
  - Saber cuál es su controlador asociado, el cual puede pedirle algunos servicios típicos como el de actualizar.

Flujo de trabajo (WorkFlow)

- ✓ El usuario realiza una acción que generará un evento.
- ✓ El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista).
- ✓ El modelo (si es necesario) llama a la vista para su actualización.
- ✓ Para cumplir con la actualización la Vista puede solicitar datos al Modelo.
- ✓ El Controlador recibe el control.

Beneficios e inconvenientes de uso**.**

Se han ido mencionando a lo largo del epígrafe las ventajas de este patrón, sin embargo se considera importante agruparlas en este punto:

- ✓ Se consigue múltiples vistas del modelo.
- ✓ Todas las vistas están sincronizadas.
- ✓ No acoplamiento, y facilidad de evolución, para cambiar las vistas y los controladores.
- ✓ La aplicación puede soportar un tipo de interfaz para cada usuario (rol).

Pero también posee algunos inconvenientes:

- ✓ La complejidad va creciendo más rápido que el tamaño de la aplicación.

- ✓ Problemas al conectar las distintas capas.
- ✓ Acceso ineficiente, pues es necesario varias llamadas al modelo para actualizar los datos.
- ✓ La vista y el controlador son específicos para una plataforma.

1.6.2 Arquitectura en capas

Definición de Capas

- ✓ Capa de presentación: Esta es la que el usuario puede ver en su ordenador, es donde se tratan los datos que se van a mostrar. Se intenta que en esta capa haya el mínimo de procesamiento. Esta capa se comunicará solamente con la capa de negocio.
- ✓ Capa de negocios: En esta capa esta la lógica, se reciben las peticiones del usuario, y tras ejecutar una acción se le envía las respuestas del proceso. Esta capa se comunica, como se ha dicho, con la de presentación, la cual le envía peticiones y esta le responde con los resultados. Y también se comunica con la capa de datos para pedirle datos.
- ✓ Capa de datos: Es donde se accede a los datos. Se hace referencia a uno o más gestores de BD que realizan el almacenamiento, modificación y consulta de los datos. Recibe peticiones desde la capa de negocios.

Beneficios e inconvenientes de uso (14)

- ✓ Facilita que se pueda descomponer la aplicación en varios niveles de abstracción.
- ✓ Facilita la evolución del sistema, ya que los cambios solo deben de afectar a la capa donde se encuentre la modificación.
- ✓ Si la interfaz accede a la misma función, no se repetirá código. Lo que conlleva la ventaja de una mayor facilidad de mantenimiento de la aplicación, entre otros.
- ✓ Si se añade un nuevo formulario no ocasionará verdaderos quebraderos de cabeza, ya que los formularios ya no dependen unos de otros, con lo que el cambiar el flujo de trabajo es algo trivial.
- ✓ Los formularios ya no acceden de forma independiente a los datos, ya que no se accede a través de ellos, al modificar los datos, esto implica que no se nos mostrará diferencia alguna.

Y como inconvenientes se encuentran:

- ✓ No todo sistema podrá ser estructurado en capas.

- ✓ Y aún pudiendo ser estructurado en capas, la separación entre una y otra no es trivial. Ya no sólo porque para un desarrollador no lo es, sino también porque muchos lenguajes y/o frameworks no están preparados para ello.

1.6.3 ¿Por qué la Arquitectura de tres capas?

Para el sistema que se desarrollará, las desventajas que presenta esta Arquitectura son irrelevantes ya que se desea un sistema de fácil estructuración y será desarrollado con tecnología que facilite el trabajo con dicha arquitectura. El Modelo Vista Controlador presenta demasiados inconvenientes como son: La complejidad va creciendo más rápido que el tamaño de la aplicación, problemas al conectar las distintas capas, acceso ineficiente, pues es necesario varias llamadas al modelo para actualizar los datos, la vista y el controlador son específicos para una plataforma.

1.7 Sistemas gestores de Bases de datos

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Por tanto debe permitir:(15)

- ✓ Definir una base de datos: Especificar tipos, estructuras y restricciones de datos.
- ✓ Construir la base de datos: Guardar los datos en algún medio controlado por el mismo SGBD.
- ✓ Manipular la base de datos: Realizar consultas, actualizarla y generar informes.

Algunas de las características deseables en un Sistema Gestor de base de datos SGBD son:

- ✓ Control de la redundancia.
- ✓ Restricción de los accesos no autorizados: Cada usuario ha de tener unos permisos de acceso y autorización.
- ✓ Cumplimiento de las restricciones de integridad.

Un SGBD está compuesto de:

- ✓ El gestor de base de datos.
- ✓ Diccionario de datos.
- ✓ Administrador de la base de datos.
- ✓ Los lenguajes.

1.7.1 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca (16).

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales características de este gestor de bases de datos son las siguientes:

- ✓ Aprovecha la potencia de sistemas multiprocesador gracias a su implementación multi-hilo.
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, entre otros.).
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

1.7.2 PostGreSQL

PostGreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el National Science Foundation (NSF), y ESL, Inc. (13)

PostGreSQL es una derivación libre (OpenSource) de este proyecto y utiliza el lenguaje SQL92/SQL99.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, pues incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones,

restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

A continuación se enumeran las principales características de este gestor de bases de datos:

- ✓ Implementación del estándar SQL92/SQL99. (17)
- ✓ Soporta distintos tipos de datos, además del soporte para los tipos base, soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, entre otros. También permite la creación de tipos propios.
- ✓ Incorpora una estructura de datos array.
- ✓ Incorpora funciones de diversa índole: Manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

1.7.3 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio, hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. Como es un Sistema muy caro, no está tan extendido como otras Bases de datos, por ejemplo, Access, MySQL, SQL Server y PostGreSQL (18).

Para desarrollar en Oracle se utiliza PL/SQL, un lenguaje de quinta generación, bastante potente para tratar y gestionar la Base de datos, también por norma general se suele utilizar SQL al crear un formulario.

Principales características de Oracle

- ✓ Es una herramienta de administración gráfica que es mucho más intuitiva y cómoda de utilizar.
- ✓ Ayuda a analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.

- ✓ Apoya en el diseño y optimización de modelos de datos.
- ✓ Asiste a los desarrolladores con sus conocimientos de SQL y de construcción de procedimientos almacenados.
- ✓ Apoya en la definición de estándares de diseño y nomenclatura de objetos.
- ✓ Documenta y mantiene un registro periódico de las constantes actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos.

1.7.4 ¿Por qué MySQL?

Se decide utilizar MySQL por las siguientes razones:

- ✓ La eficacia y velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrece mayor rendimiento.
- ✓ Su bajo consumo lo hace apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.
- ✓ Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
- ✓ Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

1.7.5 DB Designer

DB Designer es un sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecerte un método efectivo para gestionar tus bases de datos.

Permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en MySQL, Oracle, MSSQL y otras bases de datos ODBC, modelos XML y soporte para la función *drag-and-drop*. (19)

El programa dispone, además, de una interfaz profesional y de detallados manuales de uso.

Herramienta que se encuentra accesible desde esta dirección (http://dbdesigner.softonic.com/).

1.7.6 Xampp

Xampp es un servidor independiente de la plataforma en la cual se esté ejecutando, es un software totalmente libre y se encuentra bajo la licencia GNU/GPL (*GNU General PublicLicense*). Básicamente Xampp consiste en una recopilación de aplicaciones y servidores tal es el caso que encontramos a el servidor web Apache, el motor de Bases de Datos MySQL, los lenguajes de programación PHP y Perl y otros servicios más.

El nombre Xampp proviene del acrónimo X (*para cualquiera de los Sistema Operativos en la que se está ejecutando*), Apache, MySQL, PHP, Perl.

Xampp se encuentra disponible para los sistemas Microsoft Windows, GNU/Linux, Solaris y MacOSX. Permite montar todo un ambiente de servicios pre configurado y listo para hacer uso del mismo, despreocupándose de conectar algunos componentes, configurarlo y realizar conexiones.

El proyecto Xampp es absolutamente software libre, esto quiere decir que todas las librerías, módulos, lenguajes de programación y servidores son considerados como Software Libre.

Xampp es la construcción de una versión fácil de instalar para los desarrolladores que desean entrar en el mundo web y de los servidores Apache. Como ventajas a los desarrolladores que no tienen muchos conocimientos de configuraciones, Xampp nos provee con una configuración totalmente funcional, activando todas las funciones y conexiones. Un punto a tener en cuenta, y el proyecto Xampp en su sitio oficial lo recalca, es que la configuración por defecto puede ser un punto positivo desde la perspectiva del usuario poco experimentado, pero no es buena desde el punto de vista de la seguridad, lo cual indica que no es lo suficientemente seguro para aquellos sistemas de producción o ambientes grandes. (20) Herramienta que se encuentra accesible desde esta dirección (http://www.apachefriends.org/en/xampp-windows.html).

1.8 Conclusiones

- ✓ Se realizó un estado del arte de la selección de portafolios de proyectos informáticos, concluyendo la necesidad de un sistema de esta índole en la UCI.
- ✓ Se realizó un estudio de las tecnologías, metodologías y herramientas que serán utilizadas para el desarrollo de dicha aplicación, fundamentando la selección y asegurando con su uso la correcta

construcción de un sistema de selección de portafolios de proyectos informáticos que implemente una solución matemática basada en criterios actuales, múltiples y personalizables.

- ✓ Finalmente se llegó a la conclusión de que el sistema se desarrollará sobre las siguientes herramientas y tecnologías:
  - Como metodología de desarrollo de software: RUP con notación UML.
  - Gestor de base de datos: MySQL.
  - Servidor de base de datos: XAMPP.
  - IDE de desarrollo: Netbeans.
  - Lenguaje de programación: Java.
  - Para apoyar el proceso de implementación del software se utilizarán herramientas de desarrollo como: Visual Paradigm y DB Designer.

CAPÍTULO 2: MODELO MATEMÁTICO

2.1 Introducción

En este capítulo se explica el funcionamiento del modelo matemático para la selección de portafolios de proyecto. Se aborda sobre el cálculo de la importancia de los proyectos, la selección de las combinaciones de estos y la estructura de los criterios de selección. Además, se muestra un ejemplo de selección de portafolios de proyectos.

2.2 Clasificación de los criterios de selección

Los criterios se clasifican en raíz, interno u hoja. En el caso de las hojas se clasifican en cuantitativos o cualitativos y en positivo o negativo.

- ✓ Raíz: Es el padre del árbol de criterios, no contiene ponderación alguna ni se le asigna valor.
- ✓ Interno: Criterio cuyo valor es calculado a partir del impacto de sus hijos.
- ✓ Hoja: Criterio que no tiene hijos y su valor es insertado directamente por el usuario.

Un criterio hoja se clasifica en positivo cuando el número asignado a este en cada proyecto es directamente proporcional al impacto que implica este valor en el proyecto; y se clasifica en negativo cuando el impacto en la importancia del proyecto es maximizada a medida que sea menor el valor asignado a este.

Las hojas, cuyos valores son numerables, se clasifican en cuantitativos, mientras que las hojas con valores no numerables para las cuales se establecen desgloses o categorías ponderadas ejemplo: Bueno, Regular y Malo con valores de 90, 50 y 15 respectivamente, se clasifican en cualitativos.

2.3 Estructura de los criterios de selección

Se define un árbol como un grafo no dirigido, conexo y acíclico; de la misma forma un árbol enraizado es un árbol en el que marcamos uno de los vértices como vértice disinguido. Dicho vértice se denomina raíz del árbol (21). Los criterios están estructurados en forma de árbol y la raíz es el criterio inicial el cual no toma valor en ningún momento de la selección y su objetivo es el de agrupar a los criterios del primer nivel. Los criterios se comienzan a ramificar a través de los internos hasta llegar a las hojas a las cuales el

usuario le inserta su valor. Todos los criterios tienen una ponderación que es completamente configurable, pero debe cumplir que la suma de las ponderaciones de los criterios hijos de un mismo padre sea 1. El número de criterios y su organización es completamente variable evidenciando flexibilidad en el modelo.

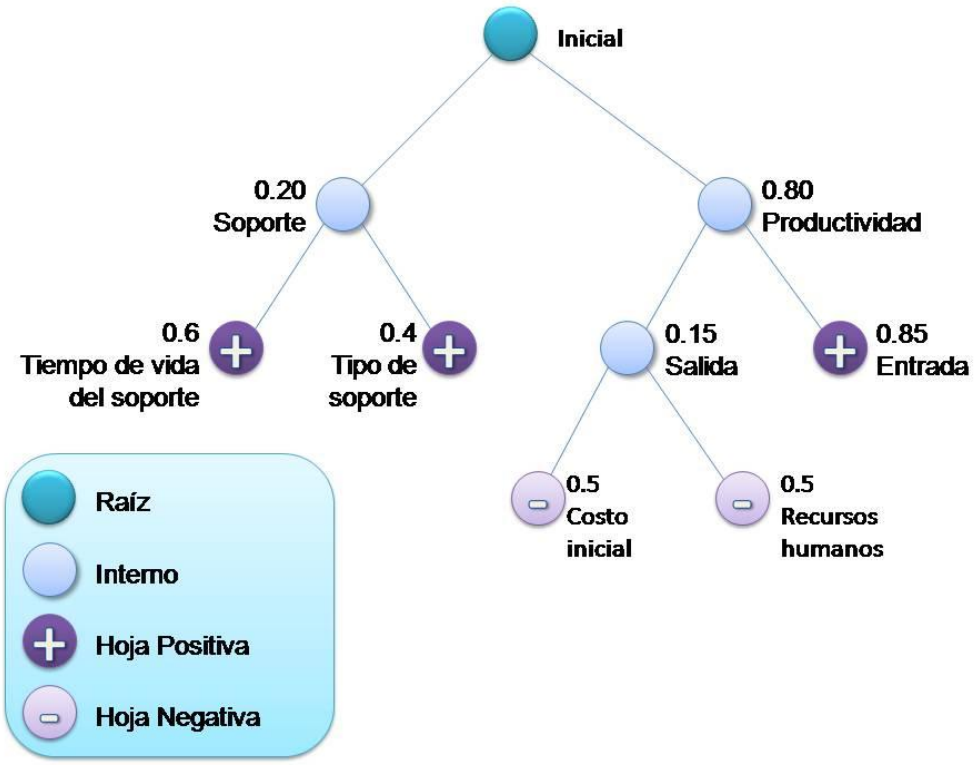


Figura 1: Ejemplo de árbol de criterios.

2.4 Cálculo de la Importancia de un Proyecto

La importancia de un proyecto (*I*) es la relevancia del mismo dependiendo de los criterios de selección especificados. Cada proyecto tiene un árbol de valores asociados, el cual es estructuralmente igual al de los criterios. Para calcular la I_r del proyecto r-ésimo se deben llenar todos los valores referentes a los criterios hojas.

Los criterios internos se calculan a partir de la siguiente fórmula:

$$X_{f,d} = \sum_{i=1}^{n} (P_{d,i} * X_{d,i})$$

Donde: $d, f \in N;\ P_{d,i}, X_{d,i}, X_{f,d} \in \mathrm{R};\ 0 \leq P_{d,i} \leq 1;\ 0 \geq \sum_{i=1}^{n} P_{d,i};\ 0 \leq X_{d,i} \leq 1;\ 0 \leq X_{f,d} \leq 1.$

d: Indicador del criterio del cual $X_{f,d}$ es el valor *i-ésimo* en el proyecto que se analiza. En $P_{d,i}$ y $X_{d,i}$ se referencia a que estos son los valores *i-ésimos* de los hijos del criterio$X_{f,d}$.

$X_{f,d}$: Valor del criterio *d-ésimo* hijo del criterio *f-ésimo*.

i: Indicador de criterio.

n: Cantidad de criterios.

$X_{d,i}$: Valor asociado al criterio *i-ésimo* hijo del criterio *d-ésimo*.

$P_{d,i}$: Ponderación del criterio asociado al valor $X_{d,i}$.

f: Indicador del padre del criterio.

La importancia es un caso particular del cálculo de los criterios internos y su diferencia reside en que sólo se recorren los criterios hijos de la raíz.

$$I_r = \sum_{i=1}^{b} (P_{1,i} * X_{1,i})$$

Donde: $X_{1,i},\ I_r, P_{1,i} \in$ R, $0 \leq P_{1,i} \leq 1, 0 \geq \sum_{i=1}^{n} P_{1,i},\ 0 \leq X_{1,i} \leq 1.$

I: Importancia o relevancia de un proyecto dependiendo de los criterios de selección dados.

r: Indicador de proyecto.

$X_{1,i}$: Valor del proyecto en el criterio *i-ésimo* hijo del criterio raíz.

$P_{1,i}$: Ponderación *i-ésima* del criterio *i-ésimo* hijo del criterio raíz.

b: grado del criterio raíz.

El valor de $X_{f,i}$ se calcula dependiendo de la clasificación del criterio que se analice.

Hojas positivas.

$$X_{f,d} = {}^{V_{r,f,d}}\Big/_{\max_s(V_{t,f,d})}$$

Donde: *s, t* ∈ N; $V_{r,f,d}, V_{t,f,d} \in$ R, $0 \leq V_{r,f,d} \leq 1$; $0 \leq V_{t,f,d} \leq 1$.

$\max_s(V_{t,f,d})$: Máximo valor que toma $V_{t,f,d}$ *d-ésimo* en los *s-ésimos* proyectos.

t: Indicador del proyecto en el max().

$V_{t,f,d}$: Valor del criterio *d-ésimo* hijo del *f-ésimo* en el proyecto *t-ésimo*.

$V_{t,f,d}$: Valor del criterio *d-ésimo* hijo del *f-ésimo* en el proyecto *r-ésimo* que refiere al proyecto actual.

s: Cantidad de proyectos.

Hojas negativas:

$$X_{f,d} = {}^{\min_s(V_{t,f,d})}\Big/_{V_{r,f,d}}$$

Donde:

$\min_s(V_{t,f,d})$: Mínimo valor que toma $V_{t,f,d}$ *d-ésimo* en los *s-ésimos* proyectos.

Como se aprecia todos los valores de los criterios hojas son comparados con sus homólogos en otros proyectos logrando que desde la base se comience a establecer prioridades a los proyectos con mejor impacto en los distintos criterios.

2.5 Cálculo del esfuerzo

En este método es necesario calcular el esfuerzo disponible o capacidad productiva de la empresa y el esfuerzo necesario para completar un proyecto.

El esfuerzo disponible de la empresa se mide en (horas/hombre) y se calcula de la siguiente manera:

$$ET = D * H * M$$

Donde: $E \in$ N, $D \in$ N, $H \in$ N, $M \in$ N.

ET: Esfuerzo total de la empresa.

D: Días de trabajo hábiles.

H: Horas de trabajo por días.

M: Número de personas de la empresa.

El esfuerzo necesario para realizar un proyecto se mide en (horas/hombre). Se estima por el usuario empleando alguna de las técnicas que propone la disciplina Ingeniería del Software y se denota por E_r.

2.6 Selección de los portafolios de proyectos

Luego de haber calculado la importancia y esfuerzo necesario para la realización de cada proyecto se pasa a la selección de los portafolios de proyectos, con este método se puede visualizar la cantidad deseada de portafolios de proyectos posibles a desarrollar.

Las combinaciones sin repeticiones de *s* elementos disponibles, tomados en grupos de *v* elementos se calcula según la fórmula: (22)

$$C_s^v = \binom{s}{v} = \frac{s!}{v!\,(s-v)!}$$

Donde: s, v $\in$ N, $v \leq s$.

Debido a que la extensión de los portafolios puede ser desde 1 hasta *s*, la cantidad de combinaciones posibles se calcula según la fórmula:

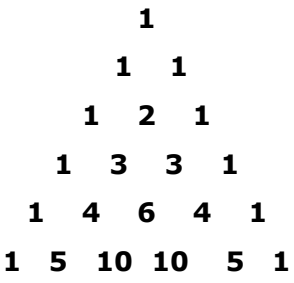
$$C\binom{s}{1} + C\binom{s}{2} + C\binom{s}{3} \ldots . + C\binom{s}{s}$$

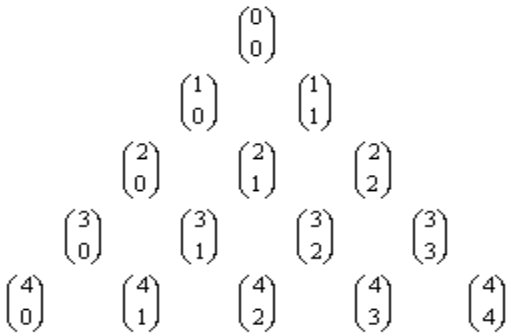
Teniendo en cuenta que estos valores coinciden con los de la fila *s-ésima* del triángulo de Pascal, eliminando la primera columna del mismo, dado que esta es referente a la combinación vacía y no se tiene en cuenta para el cálculo, la suma de una fila del triángulo de Pascal se puede calcular por la fórmula:

2^s

Donde: $s \in$ N.

s: número de proyectos.





$2^0 = 1$

$2^1 = 1+1 = 2$

$2^2 = 1+2+1 = 4$

$2^3 = 1+3+3+1 = 8$

$2^4 = 1+4+6+4+1 = 16$

.

Por lo que el número máximo de portafolios de proyecto que se pueden obtener es:

$$C\binom{S}{1} + C\binom{S}{2} + C\binom{S}{3} \ldots . + C\binom{S}{S} = 2^s - 1$$

Luego de los $2^s - 1$ portafolios posibles, según la teoría combinatoria, se dejan de generar las combinaciones que sumado el esfuerzo de realización de sus proyectos excedan el esfuerzo disponible de la empresa en la etapa seleccionada, también se puede agregar como variable restrictiva el presupuesto necesario y el disponible. Estas combinaciones se van guardando en una lista con longitud igual a la

cantidad de portafolios que se desea, y están ordenadas de mayor a menor según la suma de la importancia de los proyectos del portafolio.

En el caso de que existan proyectos con selección obligatoria se asegura de que estos proyectos nunca falten en los portafolios de la lista. A medida que se van generando nuevas combinaciones se comprueba que cumplan con los requisitos necesarios y se van adicionando a la lista si tienen una importancia total mayor a la de alguno de los portafolios existentes en ella y se elimina el portafolio de menor importancia en caso de que la lista supere el número deseado.

2.7 Aplicación del modelo

Para la aplicación del modelo se muestra un ejemplo simple donde se listan los 3 mejores portafolios para una empresa *"Y",* con 10 trabajadores que laboran 8 horas diarias en un espacio de tiempo de 100 días, con 5 proyectos hipotéticos y el árbol de criterios que se presenta anteriormente en la figura 1. En este ejemplo no se tiene en cuenta la restricción de presupuesto. En la siguiente tabla se reflejan los valores de los criterios hojas para cada proyecto, en el caso del criterio *"tipo de soporte"* que es una hoja cualitativa se usan las categorías muy frecuente, periódico y esporádico con pesos de 150, 100 y 50 respectivamente.

Proyecto	Tiempo de vida del soporte(meses)	Tipo de soporte	Costo inicial (dinero)	Recursos humanos	Entrada (dinero)
PRCV	30	Muy frecuente(150)	4000	30	120 000
COFER	10	Esporádico(50)	500	20	150 000
RITME	20	Periódico(100)	2000	15	20 000
ALISOFT	15	Periódico(100)	1000	25	50 000
CMV	10	Esporádico(50)	200	20	10 000

Tabla 1: Valores de criterios asociados a los proyectos.

Proyecto	Horas de trabajo estimadas para completar un proyecto(E_r)
PRCV	600
COFER	3000

RITME	2000
ALISOFT	1000
CMV	2600

Tabla 2: Valores para el cálculo del esfuerzo.

El esfuerzo total de la empresa para el período dado es de:

$$ET = D * H * M$$

ET=100*8*10=8000

A continuación se calculan las importancias y el esfuerzo necesario de cada proyecto *r*.

Sustituyendo para el proyecto PRCV.

$$I_1 = P_{0,1} * \left(P_{1,3} * \frac{V_{1,1,3}}{\max_r V_{r,1,3}} + P_{1,4} * \frac{V_{1,1,4}}{\max_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min_r V_{r,5,7}}{V_{1,5,7}} + P_{5,8} * \frac{\min_r V_{r,5,8}}{V_{1,5,8}} \right) + P_{2,6} * \frac{V_{1,2,6}}{\max_r V_{r,2,6}})$$

I_1=0.2*(0.6*30/30 + 0.4*150/150)+0.8*(0.15*(0.5*200/400 + 0.5*15/30) + 0.85*120000/150000)

Resolviendo I_1=0.804

$$E_1 = 600$$

Sustituyendo para el proyecto COFER.

$$I_2 = P_{0,1} * \left(P_{1,3} * \frac{V_{2,1,3}}{\max_r V_{r,1,3}} + P_{1,4} * \frac{V_{2,1,4}}{\max_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min_r V_{r,5,7}}{V_{2,5,7}} + P_{5,8} * \frac{\min_r V_{r,5,8}}{V_{2,5,8}} \right) + P_{2,6} * \frac{V_{2,2,6}}{\max_r V_{r,2,6}})$$

I_2=0.2*(0.6*10/30 + 0.4*50/150) +0.8*(0.15*(0.5*200/500 + 0.5*15/20) + 0.85*150000/150000)

Resolviendo I_2= 0.81566666666666666666666666666667

$$E_2 = 3000$$

Sustituyendo para el proyecto RITME.

$$I_3 = P_{0,1} * \left(P_{1,3} * \frac{V_{3,1,3}}{\max_r V_{r,1,3}} + P_{1,4} * \frac{V_{3,1,4}}{\max_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min_r V_{r,5,7}}{V_{3,5,7}} + P_{5,8} * \frac{\min_r V_{r,5,8}}{V_{3,5,8}} \right) + P_{2,6} * \frac{V_{3,2,6}}{\max_r V_{r,2,6}})$$

I_3=0.2*(0.6*20/30 + 0.4*100/150)+0.8*(0.15*(0.5*200/2000 + 0.5*15/15) + 0.85*20000/150000)

Resolviendo I_3= 0.29

$E_3 = 2000$

Sustituyendo para el proyecto ALISOFT.

$$I_4 = P_{0,1} * \left(P_{1,3} * \frac{V_{4,1,3}}{\max_r V_{r,1,3}} + P_{1,4} * \frac{V_{4,1,4}}{\max_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min_r V_{r,5,7}}{V_{4,5,7}} + P_{5,8} * \frac{\min_r V_{r,5,8}}{V_{4,5,8}} \right) + P_{2,6} * \frac{V_{4,2,6}}{\max_r V_{r,2,6}})$$

I_4=0.2 * (0.6 * 15 / 30 + 0.4 * 100 / 150) + 0.8 * (0.15 * (0.5 * 200 / 1000 + 0.5 * 15 / 25) + 0.85 * 50000 / 150000)

Resolviendo I_4= 0.388

$E_4 = 1000$

Sustituyendo para el proyecto CMV.

$$I_5 = P_{0,1} * \left(P_{1,3} * \frac{V_{5,1,3}}{\max_r V_{r,1,3}} + P_{1,4} * \frac{V_{5,1,4}}{\max_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min_r V_{r,5,7}}{V_{5,5,7}} + P_{5,8} * \frac{\min_r V_{r,5,8}}{V_{5,5,8}} \right) + P_{2,6} * \frac{V_{5,2,6}}{\max_r V_{r,2,6}})$$

I_5=0.2*(0.6*10/30 + 0.4*50/150) +0.8*(0.15*(0.5*200/200 + 0.5*15/20) + 0.85*10000/150000)

Resolviendo I_5= 0.217

$E_5 = 2600$

Proyectos	Importancia	Esfuerzo

PRCV	0.804	600
COFER	0.81567	3000
RITME	0.29	2000
ALISOFT	0.388	1000
CMV	0.217	2600

Tabla 3: Importancias y esfuerzos por proyectos.

Teniendo en cuenta que la empresa contiene un esfuerzo disponible de 8000 horas/hombres en el periodo dado y el esfuerzo necesario para realizar todos los proyectos es de 9200 horas/hombres, se evidencia la insuficiencia para cumplir con la demanda de la empresa y se concluye la necesidad de escoger los mejores proyectos para su desarrollo.

Tras la generación y organización de las combinaciones se obtienen las 3 propuestas de portafolios.

Portafolio No.1

PRCV importancia 0.804
COFER importancia 0.8156666666666668
RITME importancia 0.29000000000000004
ALISOFT importancia 0.38800000000000007

Portafolio No.2

PRCV importancia 0.804
COFER importancia 0.8156666666666668
ALISOFT importancia 0.38800000000000007
CMV importancia 0.21700000000000003

Portafolio No.3

PRCV importancia 0.804
COFER importancia 0.8156666666666668

ALISOFT importancia 0.38800000000000007

2.8 Conclusiones

- ✓ En este capítulo se realizó una explicación del modelo matemático elaborado lo cual permite un mejor entendimiento del mismo.
- ✓ Se presentó la clasificación y estructura de los criterios de selección para explicar la organización de dichos criterios.
- ✓ Se mostró el cálculo de la importancia de los proyectos y la selección de los portafolios de proyectos lo que muestra teóricamente cómo calcular la importancia de un proyecto y cómo se seleccionan los portafolios paso a paso.
- ✓ Finalmente, se expuso un ejemplo simple en el que se muestra la aplicación del modelo matemático.

CAPÍTULO 3: CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En el presente capítulo se hace una descripción del negocio mediante un diagrama de dominio, se enumeran los requisitos funcionales y no funcionales que el sistema que se propone debe poseer, permitiendo hacer una concepción general del mismo e identificar mediante diagramas de casos de usos las relaciones de los actores que se involucran con el sistema. Además, se realiza el análisis y el diseño del sistema donde se modelan los principales casos de uso seleccionados para la próxima iteración del producto, se definen los diagramas de clases del análisis de la aplicación y se especifica qué clases del análisis forman parte del caso de uso, las relaciones entre ellas y las entidades. En esta etapa se obtienen además los diagramas de secuencia. Posteriormente, mediante los diagramas del diseño, se lleva a cabo todo el flujo de los procesos, además del diagrama de clases persistentes y el modelo de datos.

3.2 Modelo del dominio

El modelo de dominio es una de las alternativas que brinda RUP para la identificación de requisitos y la comprensión del contexto cuando existe poca estructuración en los procesos de negocio, y con la que se le puede mostrar al usuario, de manera visual, los principales conceptos que se manejan en el dominio del sistema, sus partes y sus relaciones. Esto permite a todos los que de alguna manera están involucrados en el proceso de desarrollo del producto, manejar un vocabulario común que posibilite el entendimiento del contexto en que se sitúa el sistema. Este modelo se realiza a través de un diagrama de clases de UML simplificado, en el cual se representan las clases conceptúales que pueden intervenir en el sistema y sus asociaciones preliminares, así como los objetos más importantes en el mismo. Estos objetos del dominio representan “cosas” que existen o los eventos que acontecen en el medio en el que se desenvuelve la aplicación. (23)

3.2.1 Conceptos fundamentales

A continuación se proporciona un marco conceptual donde se identifican los conceptos fundamentales dentro del dominio, que permitirán una mejor comprensión del diagrama del modelo del dominio.

Cliente: Personas o empresas independientes que proponen la realización de proyectos a la UCI.

UCI: Universidad de las Ciencias Informáticas.

ONI: Oficina Nacional de Informatización, la cual le propone la realización de proyectos a la UCI.

Portafolio de proyectos: Conjunto de proyectos que se seleccionan a partir de un grupo principal.

Vicerrectoría de producción UCI: Oficina que se encarga de recibir los proyectos de los clientes y de la ONI para obtener los portafolios.

3.2.2 Diagrama del modelo del dominio

El diagrama del modelo del dominio es un diagrama de clases de UML que permite representar de manera visual los conceptos fundamentales que forman parte del dominio del problema, así como las relaciones existentes entre ellos.

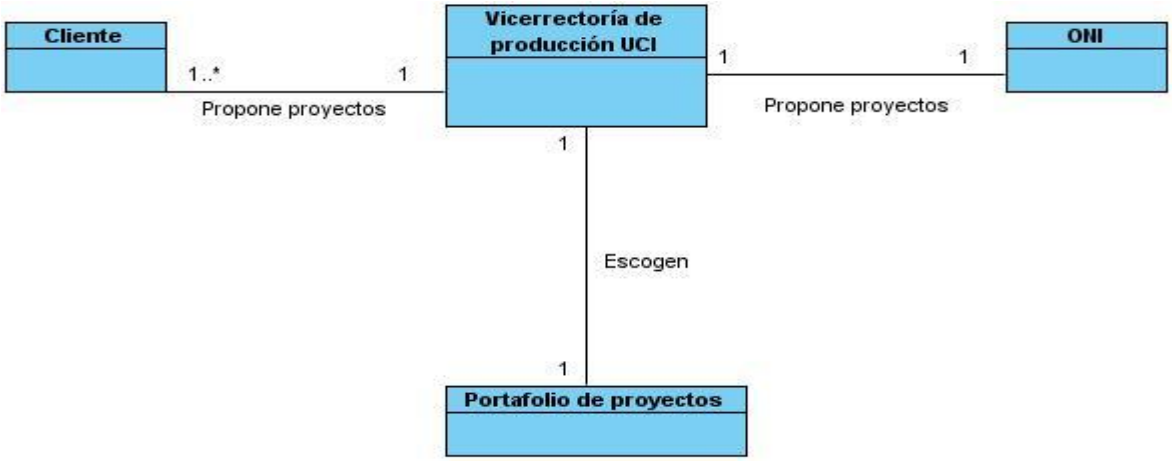


Figura 2: Diagrama del modelo del dominio

3.3 Modelo del sistema

3.3.1 Especificación de requerimientos de software

Un requerimiento es una capacidad o condición que un usuario necesita para resolver un problema o lograr un objetivo. Estos deben enfocarse en qué debe hacer el sistema y no cómo debe hacerlo. Un requerimiento debe ser bien formulado, de no ser así, se debe volver a formular. Existen requerimientos funcionales y no funcionales.

3.3.2 Requerimientos funcionales del sistema

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

Para el sistema propuesto se determinaron los siguientes requisitos funcionales.

RF1- Conectar base de datos.
RF2- Gestionar base de datos.
2.1 Crear base de datos.
2.2 Leer base de datos existente.
RF3- Gestionar proyecto.
3.1 Adicionar proyecto.
3.2 Eliminar proyecto.
3.3 Editar proyecto.
RF4- Gestionar criterio.
4.1 Adicionar criterio.
4.2 Eliminar criterio.
4.3 Editar criterio.
RF5- Asignar valores de criterios.
RF6- Gestionar categoría.
6.1 Adicionar categoría.
6.2 Eliminar categoría.
6.3 Editar categoría.

RF7- Generar portafolios.
RF8- Exportar criterio.
RF9- Importar criterio.
RF10- Guardar reporte.
RF11- Gestionar etapa.
11.1 Adicionar etapa.
11.2 Eliminar etapa.
11.3 Editar etapa.
RF12- Mostrar árbol de criterios.
RF13- Seleccionar etapa.
RF14- Imprimir reporte de portafolios.

3.3.3 Requerimientos no funcionales del sistema

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas cualidades como las características que hacen al producto atractivo, usable, rápido y confiable.

Software:
RNF1- Máquina Virtual de Java.
RNF2- MySQL versión 5.
Interfaz externa:
RNF3- El sistema debe presentar una interfaz amigable.
Usabilidad:
RNF4- El sistema debe brindar un fácil manejo para que pueda ser usado por cualquier persona que posea conocimientos básicos de computación.
Portabilidad:
RNF5- Necesidad que el sistema sea multiplataforma.
Rendimiento:
RNF6- Ante cualquier acción el sistema debe dar una respuesta rápida.
Diseño e Implementación:

RNR7- Lenguaje de programación: Java.
RNF8- Herramienta para el almacenamiento de información: Sistema gestor de base de datos MySQL.
Hardware:
RNF9- Se requiere un procesador Pentium 4 con velocidad de 2 GHz o superior.
RNF10- Se requiere como mínimo 256 megabytes de memoria RAM.

3.4 Definición de casos de uso

La arquitectura de un sistema está concretada entre otras cualidades por los casos de uso. Los casos de uso definen la manera en que se comportará el sistema.

Los actores son todas aquellas personas o sistemas que interactúan con el sistema para beneficiarse con sus resultados. Cada requisito especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la misma.

3.4.1 Definición de actores del sistema

Actores del sistema	Descripción
Experto	Es el responsable de entrar los datos al sistema y obtener las mejores combinaciones de proyectos.

Tabla 4: Actores del sistema.

3.4.2 Listado de casos de uso del sistema

CU-1	Conectar base de datos
Actor	Experto
Descripción	El experto conecta la base de datos con el sistema.
Referencia	RF-2

Tabla 5: Conectar base de datos.

CU-2	Gestionar base de datos
Actor	Experto
Descripción	El experto puede Crear una nueva base de datos o Leer base de datos existente.
Referencia	RF-1

Tabla 6: Gestionar base de datos.

CU-3	Gestionar proyecto
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar proyectos.
Referencia	RF-5

Tabla 7: Gestionar proyecto.

CU-4	Gestionar criterio
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar criterios.
Referencia	RF-5

Tabla 8: Gestionar criterio.

CU-5	Asignar valores de criterios
Actor	Experto
Descripción	El experto asigna valores de criterios a los proyectos.
Referencia	RF-3, RF-4

Tabla 9: Asignar valores de criterios.

CU-6	Gestionar categoría
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar categorías.
Referencia	RF-4, RF-5

Tabla 10: Gestionar categoría.

CU-7	Generar portafolios
Actor	Experto
Descripción	El experto obtiene las mejores combinaciones de proyectos que desee generar.
Referencia	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6

Tabla 11: Generar portafolios.

CU-8	Exportar criterios
Actor	Experto
Descripción	El experto tiene la posibilidad de exportar los criterios.
Referencia	RF-4, RF-6

Tabla 12: Exportar criterios.

CU-9	Importar criterios
Actor	Experto
Descripción	El experto tiene la posibilidad de importar criterios que tenga guardados.
Referencia	RF-4, RF-6

Tabla 13: Importar criterios.

CU-10	Guardar reporte de portafolios
Actor	Experto
Descripción	El experto puede guardar el reporte de la combinación de proyectos obtenidos.
Referencia	RF-7

Tabla 14: Guardar reporte de portafolios.

CU-11	Gestionar etapa
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar etapas de trabajo.

Referencia	RF-3, RF-4, RF-5, RF-7

Tabla 15: Gestionar etapa.

CU-12	Mostrar árbol de criterios
Actor	Experto
Descripción	El experto visualiza la estructura del árbol de criterio.
Referencia	RF-1

Tabla 16: Mostrar árbol de criterios.

CU-13	Seleccionar etapa
Actor	Experto
Descripción	El experto selecciona la etapa que desea analizar.
Referencia	RF-11

Tabla 17: Seleccionar etapa.

CU-14	Imprimir reporte de portafolios reporte
Actor	Experto
Descripción	El experto puede imprimir el reporte de la combinación de proyectos obtenidos.
Referencia	RF-7

Tabla 18: Imprimir reporte de portafolios reporte.

3.4.3 Diagrama de casos de uso del sistema

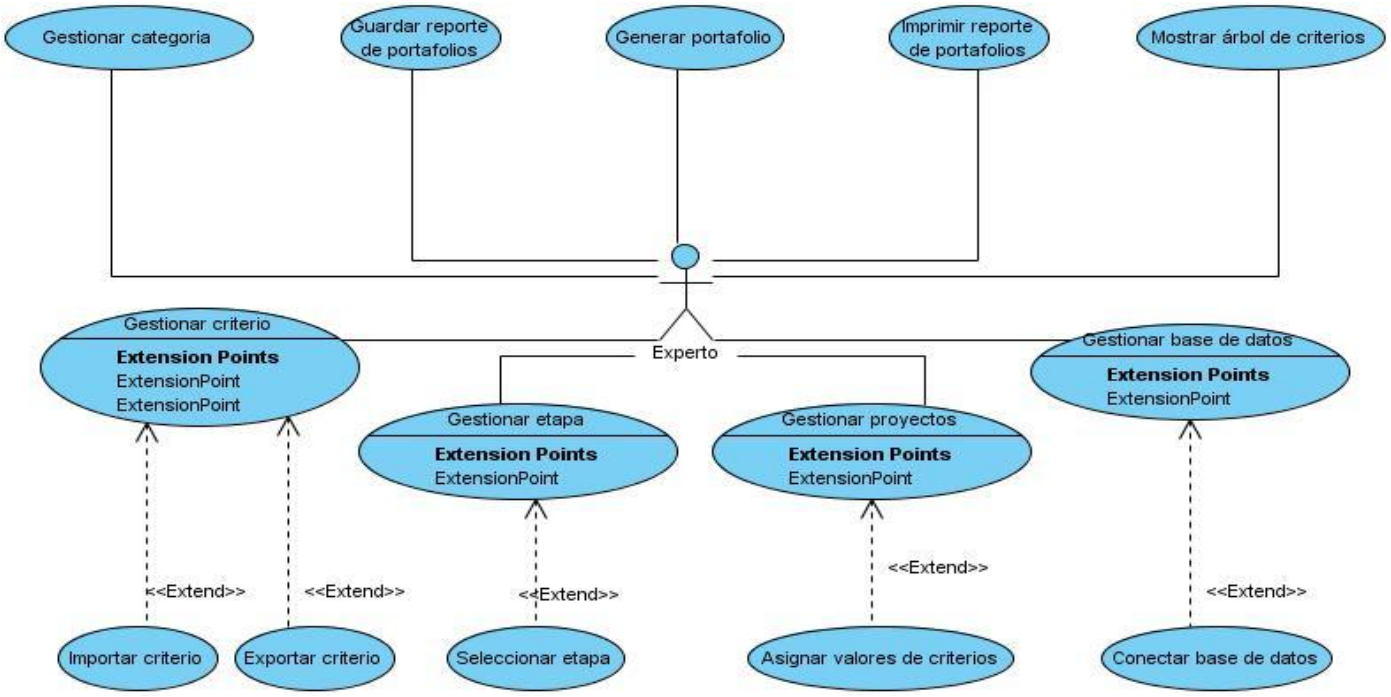


Figura 3: Diagrama de casos de uso del sistema.

3.4.4 Descripción de casos de uso del sistema

La descripción de casos de uso del sistema se realiza con el objetivo de describir brevemente cómo funcionarán los casos de uso en el sistema. En cada descripción se especifica el propósito general de cada uno de ellos, el resumen de su funcionamiento, así como las condiciones que deben existir para que estos ocurran.

En este epígrafe se puede encontrar la descripción del caso de uso Generar portafolios y pueden encontrarse las restantes descripciones en el Anexo 1.

3.4.4.1 Descripción del caso de uso Generar portafolios

Caso de uso	Generar portafolios
Resumen	El caso de uso inicia cuando el experto selecciona la opción Generar portafolios, introduce los datos, el sistema muestra la información y termina el caso de uso.
Referencia	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6
Precondiciones	Tener la base de datos conectada, haber insertado cierta cantidad de proyectos y haber adicionado criterios de selección para cada uno de ellos.
Pos condiciones	Se obtiene las mejores combinaciones de proyectos que deseen generar.
Prioridad	Crítico

Flujo normal de eventos	
Acción del actor	**Acción del sistema**
1. El caso de uso inicia cuando el experto selecciona la opción Portafolio. 3. El experto introduce los datos correspondientes y escoge la opción Generar Portafolios.	2. El sistema muestra un formulario donde el experto puede introducir los datos. 4. El sistema verifica que los valores de criterios estén llenos. 5. Si los valores de criterio no están llenos, véase flujo alterno 5.1. 6. Si los valores de los criterios están llenos el sistema muestra las mejores combinaciones de proyectos terminando así el caso de uso.
Flujo alterno 5.1	
Acción del actor	**Acción del sistema**
	5.1 Si los valores de los criterios no están llenos el sistema muestra un mensaje de error para que se llenen y retorna al paso 1.

Tabla 19: Descripción del caso de uso Generar portafolios.

3.5 Análisis del Sistema

El análisis posibilita un conocimiento razonable del sistema, ayuda a refinar y a estructurar los requisitos, permite entender mejor los aspectos internos del sistema y a través de sus diagramas ofrece una mayor formalización y poder expresivo.

3.5.1 Diagramas de clases del análisis

Uno de los principales artefactos del análisis es el diagrama de clases de análisis, en él se representan los conceptos en un dominio del problema, además se representan las clases de análisis (clase interfaz, clase controladora y clase entidad) y sus relaciones entre sí. En este epígrafe se puede encontrar el diagrama del análisis del caso de uso Generar portafolios. Pueden encontrarse los restantes diagramas en el Anexo 2.

3.5.1.1 Diagrama de clases del análisis Generar Portafolios

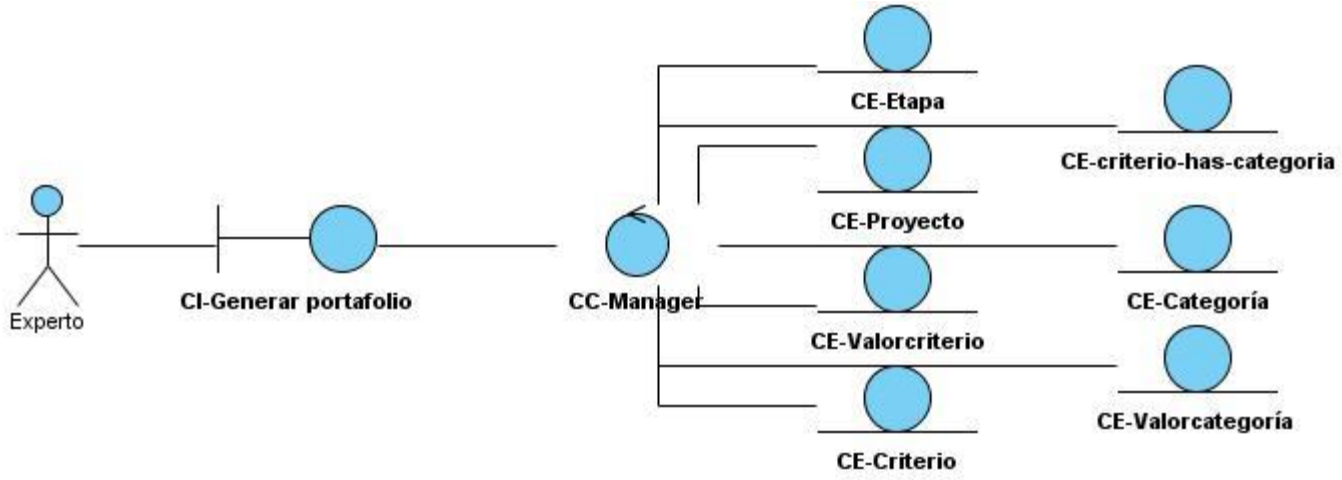


Figura 4: Diagrama de clases del análisis Generar Portafolios.

3.5.2 Diagramas de interacción

Los diagramas de interacción se usan para modelar los aspectos dinámicos del sistema, contienen objetos, enlaces y mensajes. Tienen como objetivo describir cómo interactúan las clases con sus respectivos modelos de análisis mediante diagramas de colaboración o secuencia. En este epígrafe se puede encontrar el diagrama de interacción para el caso de uso Generar portafolios. Pueden encontrarse los restantes diagramas en el Anexo 3.

3.5.2.1 Diagrama de interacción del análisis Generar portafolios

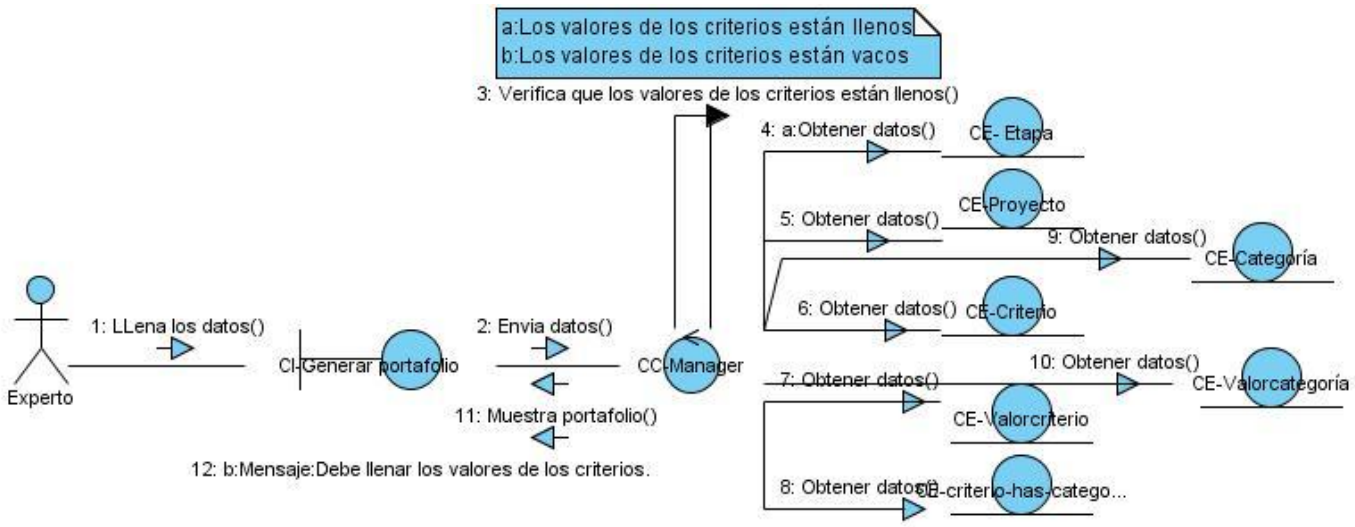


Figura 5: Diagrama de interacción Generar Portafolios.

3.6 Diseño del sistema

El modelo del diseño es otro de los tantos modelos que se tiene en cuenta en el ciclo de vida del software, consumiendo una elevada cantidad de esfuerzo para su realización. En el diseño se modela el sistema y se encuentra la forma para que soporte todos los requisitos, lo cual es de vital importancia pues sirve de abstracción de la implementación y es utilizado fundamentalmente como esbozo para la construcción del software.

3.6.1 Diagramas de clases del diseño

Los diagramas de clases de diseño describen gráficamente las especificaciones de las clases del software y contienen las clases, atributos, métodos, navegabilidad y dependencias existentes entre ellas. En este epígrafe se puede encontrar el diagrama de clases de diseño para el caso de uso Generar portafolios. Pueden encontrarse los restantes diagramas en el Anexo 4.

3.6.1.1 Diagrama de clases de diseño Generar Portafolios

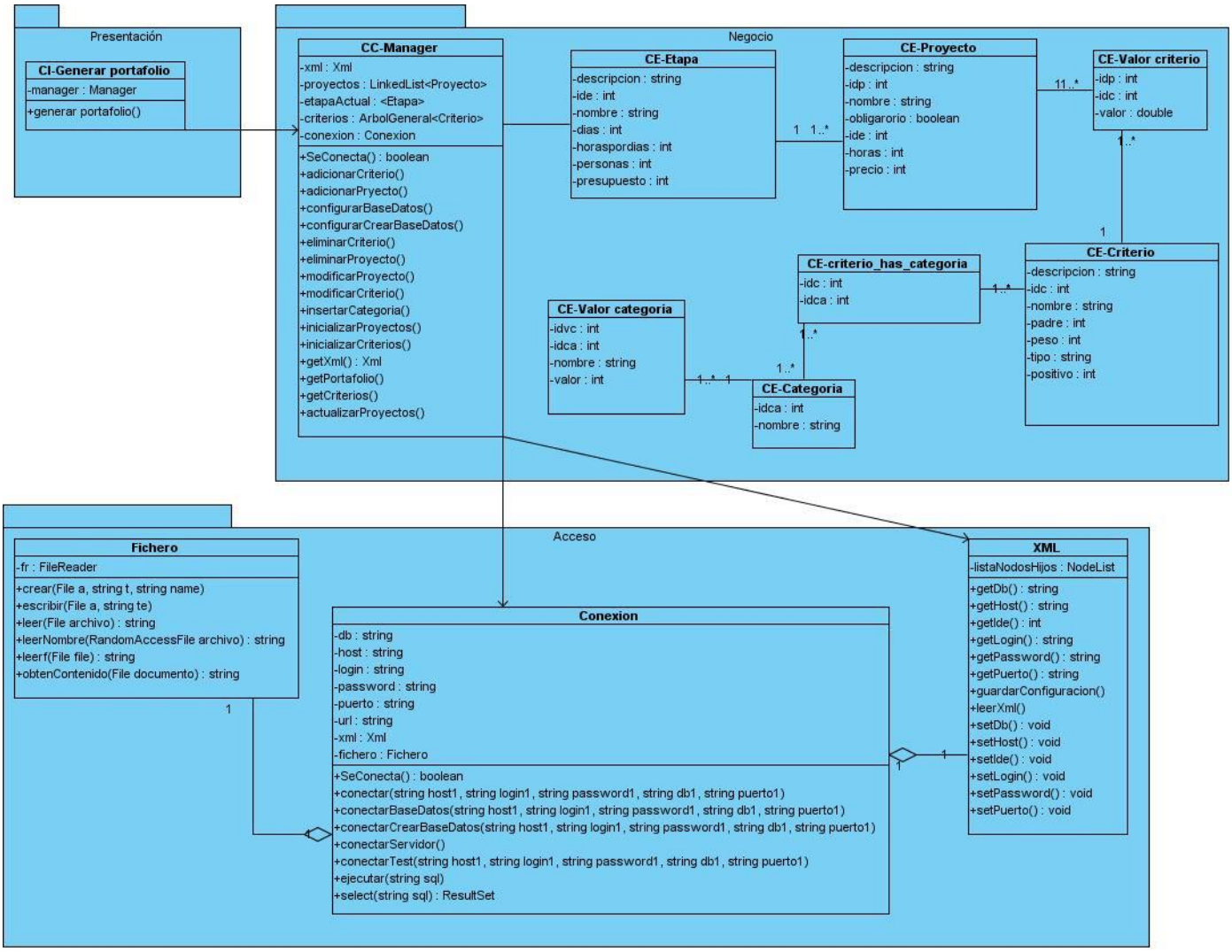


Figura 6: Diagrama de clases de diseño Generar Portafolios.

3.6.2 Diseño de la base de datos

El diseño de la base de datos es de suma importancia pues debe almacenar información y permitir al experto recuperarla y actualizarla de acuerdo a sus peticiones. Se ofrece el diagrama de clases persistentes y el modelo de datos que dan soporte al contenido manejado por el sistema.

3.6.2.1 Diagrama de clases persistentes

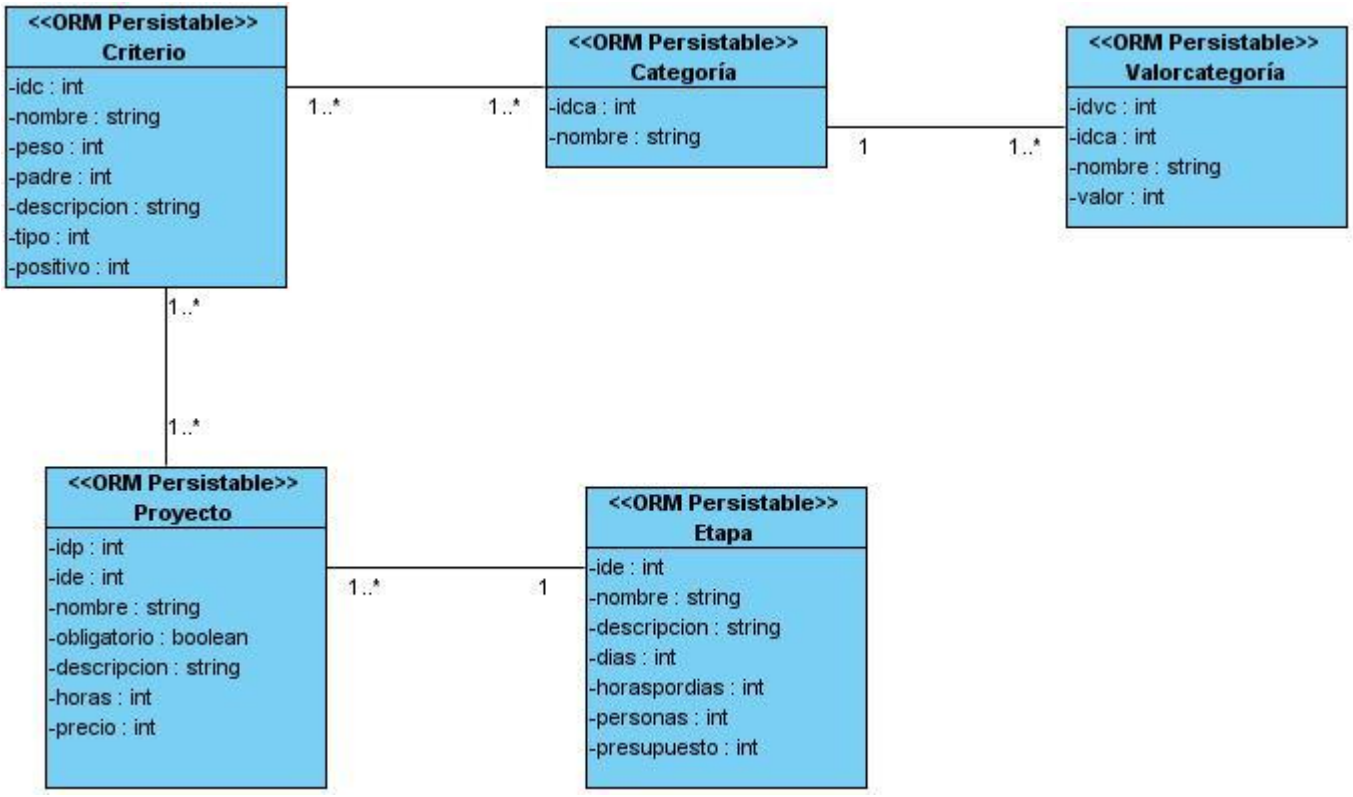


Figura 7: Diagrama de clases persistentes.

3.6.2.2 Modelo de datos

A partir del diagrama de clases persistentes se obtuvo el siguiente modelo de datos:

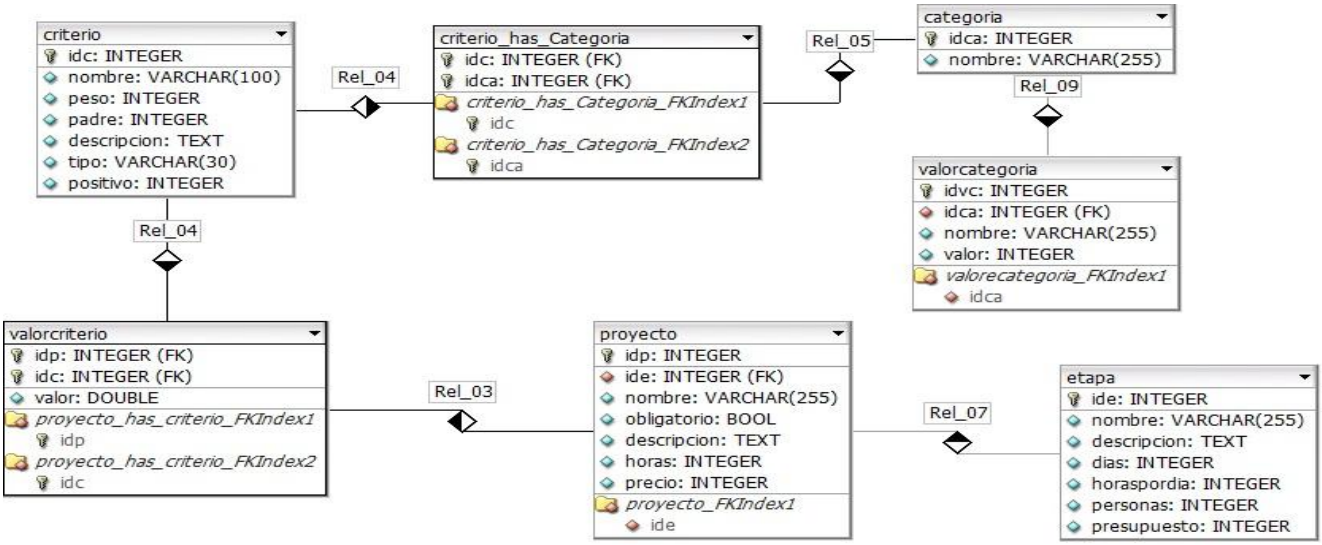


Figura 8: Modelo de datos.

3.6.2.3 Descripción de las tablas

Nombre: Etapa		
Descripción: Esta tabla guarda la información referente a las etapas.		
Atributo	**Tipo**	**Descripción**
Ide	Integer	Número que identifica a la etapa.
Nombre	Varchar	Nombre de la etapa.
Descripción	Text	Descripción de la etapa.
Días	Integer	Especifica la cantidad de días.
Horaspordias	Integer	Indica la cantidad de horas por días.
Personas	Integer	Número de personas que trabajarán en la etapa.
Presupuesto	Integer	Presupuesto destinado a esta etapa.

Tabla 20: Descripción tabla Etapa.

Nombre: Proyecto		
Descripción: Esta tabla guarda la información referente a los proyectos.		
Atributo	**Tipo**	**Descripción**
Idp	Integer	Número que identifica el proyecto.
Ide	Integer	Número que identifica la etapa.
Nombre	Varchar	Nombre del proyecto.
Obligatorio	Varchar	Representa si el proyecto es obligatorio o no.
Descripción	Text	Descripción del proyecto.
Horas	Integer	Número de horas en los proyectos.
Precio	Integer	Precio de un proyecto.

Tabla 21: Descripción tabla Proyecto.

Nombre: Valorcriterio		
Descripción: Esta tabla surge a partir de la relación muchos a muchos de las tablas Proyecto y Criterio.		
Atributo	**Tipo**	**Descripción**
Idp	Integer	Número que identifica el proyecto.
Idc	Integer	Número que identifica el criterio.
Valor	Double	Valor asignado al criterio.

Tabla 22: Descripción tabla Valorcriterio.

Nombre: Criterio
Descripción: Esta tabla guarda la información referente a los criterios.

Atributo	Tipo	Descripción
Idc	Integer	Número que identifica el criterio.
Nombre	Varchar	Nombre del criterio.
Peso	Integer	Peso asignado a cada criterio.
Padre	Integer	Padre del criterio.
Descripción	Text	Descripción del criterio.
Tipo	Varchar	Tipo de criterio.
Positivo	Integer	Número positivo.

Tabla 23: Descripción tabla Criterio.

Nombre: Criterio_has_Categoría		
Descripción: Esta tabla surge a partir de la relación muchos a muchos de las tablas Criterio y Categoría.		
Atributo	**Tipo**	**Descripción**
Idc	Integer	Número que identifica el criterio.
Idcc	Integer	Número que identifica la categoría.

Tabla 14: Descripción tabla Criterio_has_Categoría.

Nombre: Categoría		
Descripción: Esta tabla guarda la información referente a las categorías.		
Atributo	**Tipo**	**Descripción**
Idca	Integer	Número que identifica la categoría.
Nombre	Varchar	Nombre de la categoría.

Tabla 25: Descripción tabla Categoría.

Nombre: Valorcategoría		
Descripción: Esta tabla guarda los valores de las categorías.		
Atributo	**Tipo**	**Descripción**
Idca	Integer	Número que identifica a la categoría.
Idvc	Integer	Número que identifica el valor de la categoría.
nombre	Varchar	Nombre del valor de la categoría.
Valor	Integer	Valor que se le asigna a la categoría.

Tabla 26: Descripción tabla Valorcategoría

3.7 Conclusiones

- ✓ En este capítulo se modeló el proceso de funcionamiento del sistema a través del modelo del dominio, el cual permitió comprender los procesos básicos del funcionamiento.
- ✓ Se obtuvo un listado de funcionalidades que debe tener el sistema, expresados en los requerimientos funcionales, además se describieron los requerimientos no funcionales que definen las cualidades que el sistema debe cumplir.
- ✓ Quedó evidenciado que la captura de los requisitos es un paso esencial para el análisis del sistema, dando así una visión del futuro de la aplicación.
- ✓ Se elaboraron los diagramas de casos de uso del sistema y se definieron los mismos.
- ✓ Se identificaron los casos de usos críticos para permitir dar prioridad a su realización.
- ✓ Se logró modelar los diagramas de clases del análisis de los casos de usos y los diagramas de interacción para cada escenario de los mismos.
- ✓ Se definió el diseño de la aplicación, esforzándose por conservar la estructura del sistema que sirve como esquema para la implementación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

4.1 Introducción

Partiendo de los resultados del diseño, en el presente capítulo se analiza la disciplina de implementación, se muestra el diagrama de despliegue y el diagrama de componentes como objetivo primordial de esta fase. Además, se realiza una validación de la solución propuesta mediante las pruebas a las cuales se sometió la aplicación para verificar su completitud.

4.2 Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo del diseño y las clases se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros. Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación al describir los componentes y construir su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

4.2.1 Modelo de Despliegue

El modelo de despliegue agrupa los objetos que describen la distribución física entre los nodos. Cada nodo representa un recurso que interviene en el despliegue del sistema. Describe la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de software, además describe la topología del sistema, es decir, la estructura de los elementos de hardware y software que ejecuta cada uno de ellos.

4.2.1.1 Diagrama de despliegue

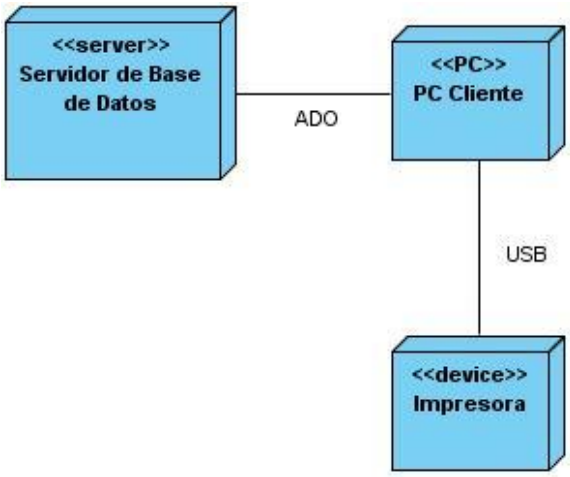


Figura 9: Diagrama de despliegue.

4.2.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software (código fuente, binarios o ejecutables). Se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

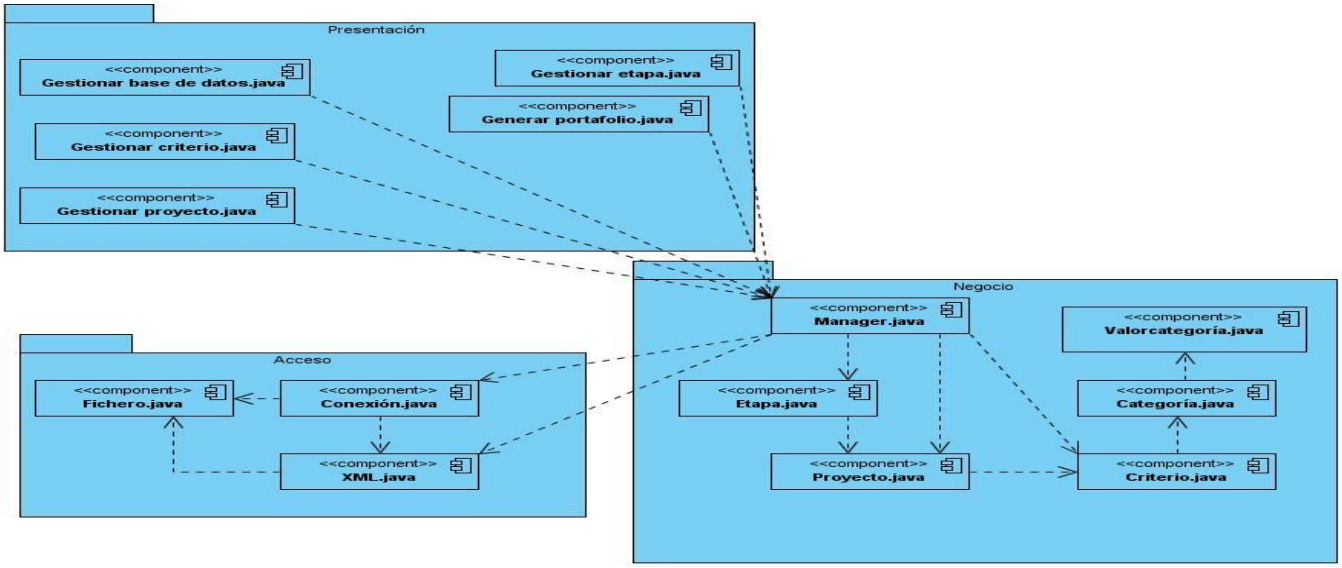


Figura 10: Diagrama de componentes.

4.3 Pruebas

Una de las últimas fases del ciclo de vida del desarrollo de software es el flujo de trabajo de pruebas, al cual es necesario dedicarle un importante tiempo pues dicha actividad está encaminada a encontrar errores en el software.

La actividad de probar el sistema y verificar que se encuentre libre de defectos tiene muchos beneficios. Por ejemplo la calidad es una variable muy importante para todo producto y uno de los caminos para garantizarla es siguiendo esta doctrina, además proporciona una medida del progreso del trabajo que se despliega.

4.3.1 Pruebas de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. (24)

4.3.2 Casos de pruebas

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada.

Los casos de pruebas deben verificar: (24)

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.
- ✓ Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Tabla 27: Prueba del caso de uso Gestionar base de datos escenario "Crear nueva base de datos"

Condición de entrada	Casos válidos	Casos no válidos

Servidor	Nombre del servidor	Campo vacío
Base de datos	Nombre de la base de datos	Campo vacío
Usuario	Cadena de caracteres	Campo vacío o usuario existente
Contraseña	Cadena de caracteres	Campo vacío
Puerto	Número entero	Campo vacío o cadena de caracteres

Caso de uso	**Gestionar base de datos escenario "Crear nueva base de datos"**
Caso de prueba	Permitir crear una nueva base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos de la base de datos correctamente de la forma: Servidor: localhost Base de datos: mysppo Usuario: root Contraseña: root Puerto: 3306
Resultado	El sistema crea una nueva base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar base de datos escenario "Crear nueva base de datos"**
Caso de prueba	Crear una nueva base de datos introduciendo mal los datos o dejando campos vacíos.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Tabla 28: Prueba del caso de uso Gestionar base de datos escenario "Leer base de datos existente"

Condición de entrada	Casos válidos	Casos no válidos

Servidor	Nombre del servidor	Campo vacío
Base de datos	Nombre de la base de datos	Campo vacío
Usuario	Cadena de caracteres	Campo vacío o usuario no existente
Contraseña	Cadena de caracteres	Campo vacío
Puerto	Número entero	Campo vacío o cadena de caracteres

Caso de uso	**Gestionar base de datos escenario "Leer base de datos existente"**
Caso de prueba	Permitir leer una base de datos existente introduciendo correctamente los datos.
Entrada	El usuario introduce los datos de la base de datos correctamente de la forma: Servidor: localhost Base de datos: mysppo Usuario: root Contraseña: root Puerto: 3306
Resultado	El sistema lee la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar base de datos escenario "Leer base de datos existente"**
Caso de prueba	Leer una base de datos existente introduciendo mal los datos o dejando campos vacíos.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Después de haberle realizado los casos de pruebas a la funcionalidad Gestionar base de datos se llegó a la conclusión de que los resultados obtenidos se corresponden con lo esperado de acuerdo con el funcionamiento correcto de esta funcionalidad.

Tabla 29: Prueba del caso de uso Gestionar proyecto escenario "Adicionar proyecto"

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del proyecto	Campo vacío o proyecto existente
Descripción	Breve descripción del proyecto	Campo vacío
Horas estimadas	Número entero	Campo vacío o cadena de caracteres
Precio	Número entero	Campo vacío o cadena de caracteres

Caso de uso	**Gestionar proyecto "Adicionar proyecto"**
Caso de prueba	Permitir insertar un proyecto en la base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos del proyecto correctamente de la forma: Nombre: RITME Descripción: Proyecto para la informatización de la empresa RITME. Horas estimadas: 60 Precio: 12000 MN
Resultado	El sistema introduce el proyecto en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar proyecto "Adicionar proyecto"**
Caso de prueba	Adicionar un proyecto introduciendo mal los datos o dejando algún campo vacío.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Tabla 30: Prueba del caso de uso Gestionar proyecto escenario "Eliminar proyecto"

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del proyecto	Campo vacío

Caso de uso	Gestionar proyecto “Eliminar proyecto”
Caso de prueba	Eliminar un proyecto de la base de datos seleccionando su nombre.
Entrada	El usuario selecciona el nombre del proyecto que desea eliminar haciendo clic en la lista de proyectos que se muestra en la interfaz
Resultado	El sistema elimina el proyecto de la base de datos.
Condiciones	No debe existir ningún campo vacío.

Caso de uso	Gestionar proyecto “Eliminar proyecto”
Caso de prueba	Eliminar un proyecto en la base de datos sin seleccionar el nombre.
Entrada	El usuario no selecciona el nombre del proyecto que desea eliminar de la lista de proyectos que se muestra en la interfaz
Resultado	El sistema muestra un mensaje de error especificando que no se admiten campos en blanco.
Condiciones	Deben existir campos vacíos.

Tabla 31: Prueba del caso de uso Gestionar proyecto escenario “Editar proyecto”

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del proyecto	Campo vacío o proyecto existente
Descripción	Breve descripción del proyecto	Campo vacío
Horas estimadas	Número entero	Campo vacío o cadena de caracteres
Precio	Número entero	Campo vacío o cadena de caracteres

Caso de uso	Gestionar proyecto “Editar proyecto”
Caso de prueba	Permitir editar un proyecto en la base de datos introduciendo correctamente los datos.
Entrada	El usuario selecciona el nombre del proyecto que desea editar e introduce los

	nuevos datos del proyecto: Nombre: ALISOFT Descripción: Proyecto para la informatización de la empresa ALISOFT. Horas estimadas: 40 Precio: 8000 MN
Resultado	El sistema modifica los datos el proyecto en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar proyecto "Editar proyecto"**
Caso de prueba	Editar un proyecto introduciendo mal los datos o dejando algún campo vacío.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Después de haberle realizado los casos de pruebas a la funcionalidad Gestionar proyecto se llegó a la conclusión de que los resultados obtenidos se corresponden con lo esperado de acuerdo con el funcionamiento correcto de esta funcionalidad.

Tabla 32: Prueba del caso de uso Gestionar criterio escenario "Adicionar criterio"

Condición de entrada	**Casos válidos**	**Casos no válidos**
Nombre	Nombre del criterio	Campo vacío o criterio existente
Ponderación	Número entero	Campo vacío o cadena de caracteres
Padre	Nombre del padre	Campo vacío
Descripción	Breve descripción del criterio	Campo vacío
Tipo de criterio	Nombre del tipo de criterio	Campo vacío

Caso de uso	Gestionar proyecto “Adicionar criterio”
Caso de prueba	Permitir insertar un criterio en la base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce el nombre del criterio, selecciona la ponderación que le va a asignar, selecciona el nombre del padre, introduce una breve descripción y selecciona el tipo de criterio.
Resultado	El sistema introduce el criterio en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	Gestionar proyecto “Adicionar criterio”
Caso de prueba	Adicionar un criterio introduciendo mal los datos o dejando algún campo vacío.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Tabla 33: Prueba del caso de uso Gestionar criterio escenario “Eliminar criterio”

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del criterio	Campo vacío

Caso de uso	Gestionar proyecto “Eliminar criterio”
Caso de prueba	Eliminar un criterio de la base de datos seleccionando el nombre del criterio.
Entrada	El usuario selecciona el nombre del criterio que desea eliminar haciendo clic en la lista de criterios que se muestra en la interfaz.
Resultado	El sistema elimina el criterio de la base de datos.
Condiciones	No debe existir ningún campo vacío.

Caso de uso	Gestionar proyecto “Eliminar criterio”
Caso de prueba	Eliminar un criterio en la base de datos sin seleccionar el nombre.
Entrada	El usuario no selecciona el nombre del criterio que desea eliminar de la lista de criterios que se muestra en la interfaz
Resultado	El sistema muestra un mensaje de error especificando que no se admiten campos en blanco.
Condiciones	Deben existir campos vacíos.

Tabla 34: Prueba del caso de uso Gestionar criterio escenario “Editar criterio”

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del criterio	Campo vacío o criterio existente
Ponderación	Número entero	Campo vacío o cadena de caracteres
Padre	Nombre del padre	Campo vacío
Descripción	Breve descripción del criterio	Campo vacío
Tipo de criterio	Nombre del tipo de criterio	Campo vacío

Caso de uso	Gestionar proyecto “Editar criterio”
Caso de prueba	Permitir editar un criterio en la base de datos introduciendo correctamente los datos.
Entrada	El usuario selecciona el nombre del criterio que desea editar y modifica los datos nombre del criterio, selecciona la ponderación que le va a asignar, selecciona el nombre del padre, introduce una breve descripción y selecciona el tipo de criterio.
Resultado	El sistema edita el criterio en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	Gestionar proyecto “Editar criterio”
Caso de prueba	Editar un criterio introduciendo mal los datos o dejando algún campo vacío.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Después de haberle realizado los casos de pruebas a la funcionalidad Gestionar criterio se llegó a la conclusión de que los resultados obtenidos se corresponden con lo esperado de acuerdo con el funcionamiento correcto de esta funcionalidad.

Tabla 35: Prueba del caso de uso Generar portafolios

Caso de uso	Generar portafolios
Caso de prueba	Permitir generar los portafolios de proyectos introduciendo correctamente los datos.
Entrada	El usuario selecciona la cantidad de portafolios que desea generar.
Resultado	El sistema muestra los portafolios de proyectos.
Condiciones	La suma de las ponderaciones de los hijos de cada criterio interno debe ser uno y los criterios hijos tienen valor en todos los proyectos.

Caso de uso	Generar portafolios
Caso de prueba	Generar portafolios de proyectos cuando la suma de las ponderaciones de los hijos de cada criterio interno no es uno o los criterios hijos no tienen valor en todos los proyectos.
Entrada	
Resultado	El sistema muestra un mensaje de error especificando el problema.
Condiciones	La suma de las ponderaciones de los hijos de cada criterio interno no es uno o los criterios hijos no tienen valor en todos los proyectos

Después de haberle realizado los casos de pruebas a la funcionalidad Generar portafolios se llegó a la conclusión de que los resultados obtenidos se corresponden con lo esperado de acuerdo con el funcionamiento correcto de esta funcionalidad.

4.4 Conclusiones

- ✓ En este capítulo se expuso la fase de implementación verificando todos los requisitos establecidos.
- ✓ Se representó el diagrama de componentes que muestra la organización y la dependencia entre un conjunto de componentes.
- ✓ Se representó el diagrama de despliegue que describe el modelo físico del sistema.
- ✓ Se provee la vista de implementación del sistema, esta etapa se caracteriza por brindar los resultados visibles ya que queda implementada la aplicación.
- ✓ Se logró desarrollar los casos de prueba para probar el curso fundamental de las funcionalidades.

Conclusiones Generales

En el presente trabajo se analizaron aspectos relacionados con el creciente avance de la industria de software y la necesidad de seleccionar portafolios de proyectos empleando herramientas basadas en criterios actuales, múltiples y personalizables. Se cumplieron los objetivos del trabajo, permitiendo arribar a las siguientes conclusiones:

- ✓ El análisis de los principales aspectos teóricos permitió obtener un estado del arte de la selección de portafolios de proyectos.
- ✓ Con el modelo matemático creado se enriquece la rama de la selección de portafolios de proyectos unificando en un modelo varias técnicas de selección.
- ✓ Se desarrolló un sistema que implementó la solución matemática elaborada en dicho trabajo.
- ✓ El sistema implementado permite el análisis de criterios de selección que varias veces se obviaban por la complejidad de los mismos y el inmenso cúmulo de cálculos que acarreaban.
- ✓ La utilización del software obtenido permite mayor eficiencia en la selección de portafolios de proyectos, aligerando el tiempo de aplicación.

Recomendaciones

Para dar continuidad al trabajo, se recomienda:

- ✓ Continuar con el estudio de la rama de la selección de portafolios de proyectos.
- ✓ Enriquecer el modelo de manera tal que planifique la selección de proyectos en función del tiempo.
- ✓ Realizar un manual de usuario más específico.
- ✓ Hacer extensiva la propuesta de este trabajo para las empresas cubanas productoras de software y las universidades.

Referencias bibliográficas

1. **Project Management Institute, Inc.***Guía de los Fundamentos de la Dirección de Proyectos.Tercera Edición.* Newtown Square, Pennsylvania : s.n., 2004.

2. **Carazo, Ana.** Un estudio holístico de la selección y planificación temporal de carteras de proyectos. [Online] [Cited: Diciembre 20, 2009.] http://www.uv.es/asepuma/recta/ordinarios/9/9_1.pdf.

3. **Pérez, Maikel Y. Leyva Vázquez y Pedro Y. Piñero.** Modelo para la evaluación y selección de proyectos de innovación en las tecnologías de la información. [Online] 2009. [Cited: Noviembre 10, 2009.]

http://octi.guanajuato.gob.mx/sinnco/formulario/MT/MT2009/MT9/SESION1/MT91_MLEYVA_145.pdf.

4. **Carazo, Ana.** Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos. [Online] [Cited: Noviembre 23, 2009.]

http://biblioteca.universia.net/ficha.do?id=39847449.

5. **Anónimo.** Programación Extrema. [Online] [Cited: Enero 31, 2010.]

http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf.

6. **RUP (Proceso Unificado de Rational).** [Online] [Cited: Enero 30, 2010.]

http://www.scribd.com/doc/7844685/CONCEPTOS-DE-RUP.

7. **Gallón, Álvaro Rendón.** Proceso Unificado de Rational para el Desarrollo de Programas. [Online] [Cited: Enero 31, 2010.] ftp://jano.unicauca.edu.co/cursos/Maestria/AplicacionesInternet/transp/RUP.pdf.

8. **Adaptive Software Development** [Consultado el: 9 de Enero de 2010]. Disponible en: http://aidanamx.blogspot.com/.

9. HUARACHI, M. Trabajo de Investigación y Exposición. Universidad de las Ciencias Informáticas.

10. **Espinola, Raúl.** ¿Qué es UML? [Online] [Cited: Enero 31, 2010.] www.robocode-argentina.com.ar/descargas/uml.ppt.

11. **Sitio de descargas de software.** [En línea] 05 de marzo de 2007. [Citado el: 28 de enero de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

12. **Anónimo.** ¿Qué es Java? [Online] [Cited: Febrero 2, 2010.]

http://www.webtaller.com/construccion/lenguajes/java/lecciones/que-es-java.php.

13. **Softonic.** [En línea] 17 de junio de 2004. [Citado el: 31 de enero de 2010.] http://netbeans-ide.softonic.com/.

14. **Juan Pablo López, Juan Pablo Martín y Javier de la Rosa.** Modelos avanzados de comunicación de recursos. [Online] http://forge.morfeo-project.org/wiki/index.php/D.3.2.2_Modelos_avanzados_de_comunicaci%C3%B3n_de_recursos#Arquitectura_Modelo-Vista-Controlador.

15. **Anónimo.** Sistema Gestor de base de datos SGBD. [Online] [Cited: Enero 30, 2010.]

http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

16. **PostGreSQL vs. MySQL.** [Online] [Cited: Enero 31, 2010.]

http://www.netpecos.org/docs/mysql_postgres/x57.html.

17. **PostGreSQL vs. MySQL.** [Online] [Cited: Enero 31, 2010.]

http://www.netpecos.org/docs/mysql_postgres/x15.html.

18. **Masip, David.** Oracle es una potente herramienta cliente/servidor para la gestión de Bases de Datos. Explicamos la herramienta y las ayudas que ofrece al desarrollador. [Online] Julio 19, 2002. [Cited: Febrero 2, 2010.] http://www.desarrolloweb.com/articulos/840.php. 12.

19. **Softonic.** [En línea] 15 de julio de 2004. [Citado el: 4 de febrero de 2010.]

http://dbdesigner.softonic.com/.

20. **WordPress.** [En línea] [Citado el: 4 de febrero de 2010.]

http://softpechis.files.wordpress.com/2009/11/instalacion-de-xampp-y-moodle.pdf.

21. **Grafos.** [En línea] [Citado el: 25 de 03 de 2010.]

http://www.upseros.com/fotocopiadora/ficheros/Matematica%20Discreta/apuntes%20de%20teoria%20de%20grafos.pdf.

22. **Pintos, Salvador y Urdaneta, Guido.** Matemáticas para la computación. Teoría Combinatoria. [En línea] septiembre de 2003. [Citado el: 25 de marzo de 2010.] http://www.ica.luz.ve/matematica/combinatoria.pdf.

23. **Jacobson, I. y Booch, G.** El proceso unificado de desarrollo de software. 2004 . Vol. 1.

24. **Entorno Virtual del Aprendizaje**. [En línea] 19 de abril de 2010. [Citado el: 25 de abril de 2010.] http://eva.uci.cu/mod/resource/view.php?id=14103.

Bibliografía

1. **Carazo, Ana.** Un estudio holístico de la selección y planificación temporal de carteras de proyectos. [Online] [Cited: Diciembre 20, 2009.] http://www.uv.es/asepuma/recta/ordinarios/9/9_1.pdf.

2. **Carazo, Ana.** Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos. [Online] [Cited: Noviembre 23, 2009.]

http://biblioteca.universia.net/ficha.do?id=39847449.

3. **Gallón, Álvaro Rendón.** Proceso Unificado de Rational para el Desarrollo de Programas. [Online] [Cited: Enero 31, 2010.] ftp://jano.unicauca.edu.co/cursos/Maestria/AplicacionesInternet/transp/RUP.pdf.

4. **Grafos.** [En línea] [Citado el: 25 de 03 de 2010.]

http://www.upseros.com/fotocopiadora/ficheros/Matematica%20Discreta/apuntes%20de%20teoria%20de%20grafos.pdf.

5. **Jacobson, I. y Booch, G.** El proceso unificado de desarrollo de software. 2004 . Vol. 1.

6. **LARMAN, C.** "UML y Patrones. Introducción al análisis y diseño orientado a objetos". 507 p.

7. **Pérez, Maikel Y. Leyva Vázquez y Pedro Y. Piñero.** Modelo para la evaluación y selección de proyectos de innovación en las tecnologías de la información. [Online] 2009. [Cited: Noviembre 10, 2009.]

http://octi.guanajuato.gob.mx/sinnco/formulario/MT/MT2009/MT9/SESION1/MT91_MLEYVA_145.pdf.

8. **Pintos, Salvador y Urdaneta, Guido.** Matemáticas para la computación. Teoría Combinatoria. [En línea] septiembre de 2003. [Citado el: 25 de marzo de 2010.] http://www.ica.luz.ve/matematica/combinatoria.pdf.

9. **Project Management Institute, Inc.***Guía de los Fundamentos de la Dirección de Proyectos.Tercera Edición.* Newtown Square, Pennsylvania : s.n., 2004.

10. **Sitio de descargas de software.** [En línea] 05 de marzo de 2007. [Citado el: 28 de enero de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

Anexo 1

Descripción del caso de uso Gestionar base de datos.

Caso de uso	Gestionar base de datos	
Resumen	El caso de uso inicia cuando el experto selecciona la opción Gestionar base de datos, el sistema muestra las opciones crear una nueva base de datos o leer una existente, el usuario introduce los datos necesarios, el sistema realiza la acción seleccionada por el experto y termina el caso de uso.	
Referencia	RF-1	
Precondiciones	Tener un servidor donde gestionar la base de datos.	
Pos condiciones	Nueva base de datos en el sistema o una existente conectada.	
Prioridad	Crítico	
Flujo normal de eventos		
Acción del actor		**Acción del sistema**
1. El caso de uso comienza cuando el experto selecciona la opción Gestionar base de datos. 3. El experto selecciona una opción: a. Si selecciona Crear nueva base de datos, véase sección Crear nueva base de datos. b. Si selecciona Leer base de datos existente, véase sección Leer base de datos existente.		2. El sistema muestra una ventana para insertar datos con las opciones de Leer base de datos existente o Crear nueva base de datos.
Sección "Crear nueva base de datos"		
Acción del actor		**Acción del sistema**
1. El experto llena los datos de la base de datos (BD)		2. El sistema verifica que todos los campos estén llenos. 3. Si hay campos vacíos véase flujo alterno 3.1. 4. El sistema verifica que los datos no existan en la base de datos. 5. Si la base de datos especificada ya existe véase flujo

	alterno 5.1. 6. Si todos los campos están llenos y la base de datos no existe, el sistema inserta la nueva BD notificándoselo al cliente mediante un mensaje, terminando así el caso de uso.
Flujo alterno 3.1	
Acción del actor	**Acción del sistema**
	3.1 Si hay campos vacíos se muestra un mensaje para que se llenen todos los campos y retorna al paso 1.
Flujo alterno 5.1	
Acción del actor	**Acción del sistema**
	5.1 Si se verifica que la BD especificada existe se muestra un mensaje de error y retorna al paso 1.
Sección "Leer base de datos existente"	
Acción del actor	**Acción del sistema**
1. El experto llena los datos de la base de datos que desea leer.	2. El sistema verifica que todos los campos estén llenos. 3. Si hay campos vacíos véase flujo alterno 3.1. 4. El sistema verifica que la base de datos exista 5. Si la base de datos especificada no existe véase flujo alterno 5.1. 6. Si todos los campos están llenos y la base de datos existe, el sistema lee la base de datos y muestra un mensaje "La conexión de la base de datos se realizó exitosamente", terminando así el caso de uso.
Flujo alterno 3.1	
Acción del actor	**Acción del sistema**
	3.1 Si hay campos vacíos se muestra un mensaje para que se llenen todos los campos y retorna al paso 1.
Flujo alterno 5.1	
Acción del actor	**Acción del sistema**

	5.1 Si se verifica que la BD especificada no existe se muestra un mensaje de error y retorna al paso 1.

Tabla 32: Descripción del caso de uso Gestionar base de datos.

Descripción del caso de uso Gestionar proyecto.

Caso de uso	Gestionar proyecto.
Resumen	El caso de uso inicia cuando el experto selecciona la opción Proyectos, el sistema le brinda la opción de adicionar, eliminar o editar un proyecto, el experto introduce los datos necesarios, el sistema realiza la acción seleccionada y termina el caso de uso.
Referencia	RF-5
Precondiciones	Tener la base de datos conectada y haber creado una etapa.
Pos condiciones	Adicionado, eliminado o editado un proyecto.
Prioridad	Crítico
Flujo normal de eventos	
Acción del actor	**Acción del sistema**
1. El caso de uso inicia cuando el experto selecciona la opción Proyectos. 3. El experto selecciona una opción: a. Si selecciona Adicionar proyecto, véase sección Adicionar proyecto. b. Si selecciona Eliminar proyecto, véase sección Eliminar proyecto. c. Si selecciona Editar proyecto, véase sección Editar proyecto.	2. El sistema brinda la posibilidad de Adicionar proyecto, Eliminar proyecto o Editar proyecto.
Sección "Adicionar proyecto"	
Acción del actor	**Acción del sistema**
2. El experto introduce los datos del proyecto.	1. El sistema muestra un formulario para adicionar un proyecto. 3. El sistema verifica que todos los campos obligatorios estén llenos. 4. Si los campos obligatorios no están llenos véase flujo

	alterno 4.1. 5. El sistema verifica que el proyecto no exista en la base de datos. 6. Si el proyecto especificado existe véase flujo alterno 6.1. 7. Si los capos obligatorios están llenos y el proyecto no existe el sistema inserta el proyecto notificándoselo al cliente mediante un mensaje, terminando así el caso de uso.
Flujo alterno 4.1	
Acción del actor	**Acción del sistema**
	4.1 Si hay campos obligatorios vacíos se muestra un mensaje para que se llenen todos los campos y retorna al paso 1.
Flujo alterno 6.1	
Acción del actor	**Acción del sistema**
	6.1 Si se verifica que el proyecto especificado existe se muestra un mensaje de error y retorna al paso 1.
Sección "Eliminar proyecto"	
Acción del actor	**Acción del sistema**
2. El experto selecciona el proyecto que desea eliminar. 4. El experto elimina el proyecto.	1. El sistema muestra un formulario para eliminar un proyecto. 3. El sistema muestra la información del proyecto. 5. El sistema elimina el proyecto seleccionado notificándoselo al cliente mediante un mensaje, terminando así el caso de uso.
Sección "Editar proyecto"	
Acción del actor	**Acción del sistema**
	1. El sistema muestra un formulario para editar un proyecto.

2. El experto selecciona el proyecto que desea editar. 4. Luego modifica los datos que desea editar.	3. El sistema muestra la información del proyecto. 5. El sistema verifica que los campos obligatorios estén llenos. 6. Si los campos obligatorios no están llenos véase flujo alterno 6.1. 7. Si todos los campos obligatorios están llenos el sistema edita el proyecto notificándoselo al cliente mediante un mensaje, terminando así el caso de uso.
Flujo alterno 6.1	
Acción del actor	**Acción del sistema**
	6.1 Si hay campos obligatorios vacíos se muestra un mensaje para que se llenen todos los campos y retorna al paso 1.

Tabla 33: Descripción del caso de uso Gestionar proyecto.