Universidad de las Ciencias Informáticas

Facultad 7

Título: Sistema para la gestión de métricas.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Dunieski Sarmiento Méndez.

Romel Rodríguez Torres.

Tutora: Ing. Irina Napal Torres.

Co-tutores: Ing. Inalvys Hernández Manso.

Ing. Asmel Navarro Camero.

Ciudad de la Habana

Junio 2009

AGRADECIMIENTOS

Agradecemos a nuestro querido Comandante en Jefe Fidel y a la UCI por darnos la oportunidad de formar parte de este proyecto.
A nuestra tutora Irina y a Inalvys, por toda la ayuda y atención brindada.

Agradezco a mi mamá Luisa por ser partícipe y motor impulsor de este sueño que hoy se hace realidad. A mi papá Onelio, mis tíos Papito y Segundo, a mis tías María, Rosa, Gloria y Elizabeth, a mis primos Lisandra y Yosmel, a Eguita, Denia, Maruchi, Margot y Horacio, a mi madrastra Aidita y mi padrastro Eduardo, a mi abuela Evelia y a LLellita, gracias a todos por estar ahí cuando los necesito. Agradezco también a mis amistades, a mi compañero de tesis Dunieski, a Grette a la cual considero mi mejor amiga, Ariel, Yania, Osvaldo, Yislenys, Liliana, Débora, Adriana, Javier y Yansel. Agradecer en especial a Yadainy, mi mamá en la universidad, reconocer su apoyo incondicional y su eterna enseñanza en los aspectos de la vida y la profesión. A todos los que de una forma y otra contribuyeron a mi formación personal y profesional.

Romel.

A mis padres, por su apoyo incondicional, por guiarme siempre por un buen camino y haber hecho de mí la persona que soy hoy. A mi hermanita, a quien quiero mucho, y he apoyado y me ha apoyado siempre que lo he necesitado. A mi novia, compañera incondicional durante todos estos años, que ha hecho de mí una mejor persona, gracias por aguantar todas mis majaderías y por dejarme entrar en tu corazón. A Mamá y Papá que tanto me han ayudado durante todo este tiempo. A mis abuelos, mis tías, mis primotes y mis sobrinitos. A Yislenys, Grette, Lilianna, Débora, Solainy, Romel, Osvaldo y Ariel, amigos de siempre que han estado a mi lado en las buenas y en las malas. A Yania, una amiga que aunque nueva se ha ganado mi cariño y mi confianza. A Yadainy quien más que nuestra jefa ha sido nuestra otra mamá. A todos los que siendo amigos, compañeros, vecinos o profesores a lo largo de mi vida, hicieron posible de una forma u otra que llegara hasta aquí.

Dunieski.

DEDICATORIA

A mi mamá, lo más grande que tengo en la vida, por guiarme siempre por el camino correcto, ejemplo de sacrificio y dedicación, motivo de orgullo y respeto siempre.

A mi abuelita Sila que en paz descanse, por toda su Fe y amor brindado.

A mi hermanita Elenis, a quien quiero mucho y es parte inseparable de mi vida.

A mi tío Papito, por su apoyo incondicional durante todos mis estudios, por siempre estar ahí cuando lo necesité.

Romel.

A mi abuelo Toño quien de haber llegado a verme este día estaría muy orgulloso de mí.

A María y a Carlos, mis padres, a quien debo todo lo que soy, por quererme, por estar siempre conmigo cuando los necesité y sobre todo porque los quiero mucho.

A Dalenny, mi hermana, por estar a mi lado en todo momento.

A Lorena, por ayudarme y por guiarme cuando he creído que todo se viene abajo.

A mis abuelos, mis tías, mis primos, mis sobrinos y a todos los que me ayudaron para que este sueño se hiciera realidad.

Dunieski.

RESUMEN

La Industria Cubana del Software está llamada a convertirse a corto plazo, en uno de los principales renglones económicos del país. Para esto, producir un producto con excelente calidad, es la meta a seguir por las empresas productoras de software del país y también por la Universidad de las Ciencias Informáticas.

Uno de los aspectos de mayor impacto en la calidad del software son las métricas de software, las que tienen entre sus objetivos ayudar a entender qué ocurre durante el desarrollo y el mantenimiento del mismo.

Este trabajo realizado en la Universidad de las Ciencias Informáticas, brinda una solución informática a modo de aplicación web que gestiona las métricas de software, basada en tecnología PHP y con gestor de base de datos PostgreSQL. Esta fue desarrollada a raíz de la necesidad por parte del Grupo de Métricas de la Dirección de Calidad de Software de la universidad, de contar con una herramienta que le permita mantener informados a todos los proyectos productivos con las últimas definiciones del tema.

Para dicha aplicación se trazaron varias tareas fundamentales, entre las que figuraron hacer un estudio del estado del arte sobre el tema a tratar, proponer la solución planteada, modelar e implementar la solución, y validarla con los usuarios finales, lo que permitió finalmente, hacer las recomendaciones necesarias para la continuidad y explotación de la propuesta elaborada.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad el mundo está guiado por el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC). A su vez estas tecnologías están regidas por la informática como ciencia fundamental y por el uso de programas de cómputo o software, que facilitan el trabajo en las diferentes actividades de la vida diaria.

La Industria de Software se ha convertido en una de las más poderosas y crecientes del momento. Cada día son más las empresas que se suman motivadas por la competencia y el desarrollo de productos de alta calidad, todas con el objetivo de ganar un lugar en el amplio mercado que se impone con el mundo moderno.

Cuba es un país que ha sabido abrirse camino en esferas como la salud, la educación, la biotecnología y el deporte, lo que sin duda alguna, la ha hecho merecedora de un prestigio que la sitúa en un alto pedestal a nivel mundial. Estos factores dan el índice de lo que es capaz de lograr una nación que ha sido asediada por medio siglo por una de las potencias más poderosas del mundo, los Estados Unidos de América (E.U.A). Pese a las limitaciones impuestas y ser la producción de software una tendencia de los países más desarrollados tecnológicamente, la mayor de las Antillas se ha trazado como meta convertir la informática en una significativa fuente de ingresos para el país, mediante el correcto aprovechamiento del capital humano disponible. Así, surgida como uno de los programas de la Batalla de Ideas, la Universidad de las Ciencias Informáticas (UCI) se convierte en el primer proyecto tangible para el logro de las metas trazadas en pos de desarrollar la Industria Cubana del Software (ICSW).

Las empresas cubanas al igual que las del resto del orbe, tienen entre sus pilares fundamentales obtener productos que logren satisfacer las expectativas del cliente y que cuenten con una calidad requerida. El software, como un producto más del mercado, debe estar también provisto de calidad, por lo que es necesario tener en cuenta una serie de aspectos que no pueden faltar, como es el caso de las métricas.

Las métricas del software tienen entre sus objetivos ayudar a entender qué ocurre durante el desarrollo y el mantenimiento del software, además de controlar qué es lo que acontece en los proyectos, y poder mejorar los procesos y los productos.

INTRODUCCIÓN

Desde sus inicios la UCI fue concebida para formar ingenieros con una alta preparación profesional, capaces de no solo producir, sino también de obtener productos de alta calidad. En la Universidad la Dirección de Calidad de Software (DCS) se encarga de asegurar que los productos desarrollados salgan con la calidad requerida al mercado. Esta dirección cuenta con un grupo de trabajo que se especializa en la implantación y el uso de las métricas en los diferentes proyectos productivos, así como la garantía de su cumplimiento.

Actualmente el Grupo de Métricas (GM) no cuenta con un espacio que permita, de manera estandarizada y centralizada, especificar las métricas a manejar en los proyectos, de manera tal que los mismos puedan consultarlas en cualquier momento y obtener información actualizada y detallada para su mejor aplicación. Del mismo modo es un proceso engorroso actualizar a cada proyecto siempre que sea modificada, agregada o eliminada una métrica al sistema de medición; lo que implica problemas de desinformación y/o inadecuada utilización de las mismas en los diferentes espacios por no contar con la información necesaria. Esto sin lugar a dudas influye en el correcto desarrollo del software, pues las mediciones constituyen un mecanismo de alta importancia debido a que permiten evaluar la productividad de los desarrolladores, además de medir parámetros como funcionalidad, eficiencia y costo; brindando la posibilidad de marcar un punto de partida para las estimaciones.

Analizando la situación anteriormente expuesta, se impone el siguiente problema: ¿Cómo gestionar el proceso de registro y actualización de las métricas establecidas por el GM de la DCS?

Con la informatización del registro y actualización de las métricas establecidas por el GM el proceso sería menos engorroso, pues se verían beneficiados tanto los proyectos como los especialistas de dicho grupo. Por ende, el objeto de estudio de la investigación se centra en el proceso de gestión de las métricas, y el campo de acción que abarca el mismo es la informatización del proceso de registro y actualización de las métricas establecidas por el GM de la DCS.

El objetivo general del trabajo es desarrollar un sistema para la gestión de las métricas que permita mantener la información referente a las mismas actualizada para la consulta de los interesados.

Como objetivos específicos se plantean:

- Realizar un estudio del estado del arte del uso de las métricas.

- Caracterizar los elementos compositivos de las Métricas, en función de su tipificación.
- Proponer el Sistema para la Gestión de Métricas.
- Desarrollar el Sistema para la Gestión de Métricas.
- Caracterizar el ambiente de implantación del Producto.
- Validar la propuesta con los Usuarios Finales.
- Recomendar mejoras a partir de la retroalimentación.

Para dar cumplimento a los objetivos planteados, se hace necesaria la realización de las siguientes tareas:

- Hacer un análisis relacionado con los procesos de medición y realizar un estudio acerca del uso de las métricas, así como su composición y clasificación.
- Analizar las tecnologías vigentes para llevar a cabo sistemas como el que se pretende desarrollar.
- Determinar la metodología de desarrollo a utilizar para la concesión del producto.
- Selección y fundamentación de las herramientas a utilizar y de la plataforma en la que se desarrollará la aplicación.
- Modelar el sistema obteniendo todos los artefactos necesarios para el desarrollo del flujo de implementación.
- Diseñar la Base de Datos que soportará al sistema.
- Implementar una aplicación que gestione las métricas definidas por el GM.
- Realizar la validación del software con los Usuarios Finales mediante pruebas de aceptación.
- Hacer las recomendaciones necesarias a partir de los resultados obtenidos.

Los métodos científicos utilizados en la investigación realizada permitieron apreciar el problema integralmente y con objetividad en todas sus dimensiones. Para dar cumplimientos a las tareas investigativas se emplearon los siguientes:

Métodos teóricos:

- **Analítico – Sintético:** Utilizar este método resultó importante a la hora de fundamentar los elementos más significativos de la investigación. Permitió además hacer una selección correcta de

las tecnologías, metodologías de desarrollo y herramientas a utilizar elaborando conclusiones de las mismas.

- **Histórico-Lógico:** Este método sirvió de apoyo en el estudio del estado del arte de la problemática a resolver. Se analizaron las ventajas y desventajas de cada una de las tendencias posibles en la solución del problema.

Métodos empíricos:

- **Entrevista:** La misma se llevó a cabo a la hora de definir con el cliente qué es lo que el sistema debe hacer, identificando las funcionalidades requeridas por el mismo y las restricciones que se imponen para su desarrollo. Además concluyó también identificando cuales serian de manera general los resultados esperados de la presente investigación.

Como resultado de este trabajo, se obtiene por primera vez en la Universidad de las Ciencias Informáticas, tributando a la Dirección de Calidad de Software, una aplicación Web para gestionar las métricas de software. Este sistema informático proporciona comodidad, rapidez y confiabilidad a la hora del control de las métricas por parte de los directivos del GM y brinda un espacio de consulta para los proyectos productivos de la universidad.

El contenido de este trabajo se encuentra estructurado en cuatro capítulos de la siguiente forma:

En el capítulo 1, “Fundamentación teórica”, se hace un análisis acerca del uso e importancia de las métricas, así como de las diferentes herramientas que se usan en este campo actualmente. Se enuncian las tendencias y tecnologías actuales seleccionadas a emplear en el desarrollo de la propuesta y se fundamenta el por qué de su utilización.

El capítulo 2, “Características del sistema”, describe la situación actual, define el objeto de informatización y la propuesta del sistema. Se definen además las funcionalidades del sistema, a través de los requerimientos funcionales y la descripción de los casos de uso.

El capítulo 3, “Análisis y diseño del sistema”, aborda sobre la elaboración de la propuesta de solución. Se modelan los diagramas de clases del análisis y del diseño, los diagramas de secuencia del diseño, se

obtiene el modelo de datos, y se plantean los principios de diseño a tener en cuenta en la implementación del sistema.

En el capítulo 4, “Implementación y prueba” se procede con la construcción de la propuesta de solución. Se modela el diagrama de despliegue y los diagramas de componentes, y se valida la propuesta mediante pruebas de aceptación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se tratan una serie de elementos teóricos y conceptuales relacionados con los procesos de medición y el uso de las métricas en el desarrollo del software. También se expone un análisis de las herramientas y tecnologías utilizadas actualmente, así como las que han sido seleccionadas; además de una caracterización de la metodología a utilizar para desarrollar el sistema.

1.2. Métricas de software.

La medición no es más que determinar la proporción entre la dimensión de lo que se mide con una determinada unidad de medida. Su concepto está muy ligado al de las métricas, ya que estas se pueden definir como: “La continua aplicación de técnicas basadas en la medición al proceso de desarrollo de software y a sus productos para proveer información administrativa significativa y oportuna, junto con el uso de esas técnicas para mejorar el proceso y sus productos.” **[1]**

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas. La informática aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software, la cual tiene entre sus principales objetivos producir ya sea un sistema, un producto o una aplicación con alta calidad.

En el cumplimiento de este objetivo las métricas juegan un papel primordial, pues antes de lograr una buena calidad del producto, es necesario definir las mediciones y los criterios que serán utilizados para identificar el nivel deseado de la calidad y además poder determinar si este ha si ha sido o no alcanzado.

1.3. Tipos de métricas.

De acuerdo a lo que miden, las métricas están orientadas a tres entidades fundamentales: producto, proceso y proyecto.

- **Métricas del producto:**

  Los productos están compuestos por artefactos, los cuales pueden ser documentos, modelos, módulos, o componentes, por tanto, las métricas del producto deben hacerse sobre la base de medir

cada uno de los artefactos. Las métricas del producto describen características como el tamaño, complejidad, rasgos del diseño, rendimiento y nivel de calidad. **[2]**

- **Métricas del proceso:**

Los procesos de software se pueden definir como una secuencia de pasos requeridos para desarrollar o dar mantenimiento al mismo. Las métricas del proceso son las que cuantifican el comportamiento de los procesos, los cuales son generalmente objetivos, absolutos, explícitos y dinámicos. **[3]**

- **Métricas del proyecto:**

Las métricas del proyecto son aquellas que describen las características del proyecto y la ejecución de éste. Algunos ejemplos pudieran ser: el número de programadores de un software, el costo, planificación y productividad del equipo. **[4]**

Además de estar divididas en estos tres grupos, una métrica puede ser base o derivada. Una métrica base es aquella de la cual se pueden realizar mediciones sin depender de ninguna otra métrica y cuya forma de medir es un método de medición. En tanto una métrica derivada es una métrica cuya forma de medir es una función de cálculo, es decir, las mediciones de dicha métrica utilizan las medidas obtenidas en mediciones de otras métricas bases o derivadas.

1.3.1. Importancia de las mediciones.

Las mediciones son fundamentales en cualquiera de las áreas de la ingeniería y la ingeniería de software no está exenta de ello. Las mismas se pueden aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Además se pueden utilizar para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos.

Mediante la medición, se pueden señalar las tendencias buenas o malas, realizar mejores estimaciones, llevar a cabo una verdadera mejora sobre el tiempo. **[5]**

La base fundamental de las mediciones son los datos estadísticos. Sin importar cual sea la empresa, negocio o entidad, si se carece de estos datos, no es posible tomar decisiones y adoptar medidas preventivas y correctivas con tiempo suficiente, para evitar daños que pueden llegar a ser irreparables en el peor de los casos.

Todo análisis, toda decisión e incluso todo presupuesto, depende, en gran medida de poseer datos estadísticos suficientes y de realizar un buen proceso de medición. Por tal razón es necesario saber como llevar a cabo un buen proceso de medición y conocer al detalle cuales son las métricas que necesitamos para ello.

1.4. Herramientas y tecnologías usadas en la medición y aplicación de las métricas.

El uso de las métricas está presente en casi todas las esferas de la industria, por lo que medir ha dejado de ser algo sencillo. Ante tal situación ha sido necesaria la búsqueda de soluciones que ayuden no solo a medir, sino a registrar las mediciones que se hagan. De aquí, que actualmente existan en el mundo herramientas que de una manera u otra registran y/o almacenan métricas bases, algunas que se encargan de calcular y obtener las métricas derivadas, así como otras que gestionan el proceso de medición.

1.4.1. PSM Insigth.

PSM Insight es una herramienta de medición, desarrollada en software libre, que implementa el proceso completo de PSM (*Practical Software and Systems Measurement*). **[6]**

La herramienta permite:

- Implementar PSM Insight en su área de trabajo.
- Ajustar sus necesidades de información a la herramienta.
- Configurar las plantillas de PSM acorde a su proyecto.
- Seleccionar y especificar medidas relevantes.
- Crear y analizar indicadores complejos.
- Importar datos de otras aplicaciones.
- Aplicar PSM y los principios ISO/IEC 15939 a sus proyectos.

Pese a todos estos beneficios, PSM Insight es una aplicación de escritorio que solo puede ser ejecutada por una persona a la vez, por lo que aunque gestiona y organiza el proceso de medición, no brinda un espacio de uso colectivo que ayude a mantener a todos los equipos de desarrollo bien informados en cuanto al uso de las métricas.

1.4.2. Process Dashboard.

Otra de las herramientas utilizadas en el campo de las mediciones es el Process Dashboard (PD). El mismo fue desarrollado originalmente en 1998 por la fuerza aérea de Estados Unidos, y ha continuado desarrollándose bajo modelo open-source. Está libremente disponible para la descarga bajo condiciones de la GNU Public License (GPL: Licencia Pública de GNU).

El Process Dashboard es un software libre de apoyo a PSP con módulos que incluyen los guiones y formularios que aparecen en el libro de Watts Humphrey “A discipline for Software Engineering”, donde se describe el Proceso Personal de Software. El PD fue desarrollado en Java, y necesita para correr en cualquier sistema de una Máquina Virtual Java. Corre en sistemas Windows, Linux, Unix, y Macintosh. Para acceder a los formularios y guiones se requiere disponer de algún navegador web. **[7]**

Esta herramienta brinda a todos los miembros del equipo de desarrollo la posibilidad de medir su tiempo de trabajo de una forma sencilla, pero al igual que PSM Insight es de escritorio y solo puede ser ejecutada por una persona a la vez. Además es una aplicación para recoger métricas ya definidas, de modo que no permite la definición de nuevas métricas.

1.4.3. Repositorios de métricas.

Desde sus inicios las técnicas de estimación de proyectos informáticos o de software se han fundamentado en el estudio y observación de datos históricos de proyectos realizados anteriormente. Los repositorios juegan un papel elemental a la hora de almacenar estos datos, por lo que el término Repositorio de métricas ya no resulta ignorado cuando de mediciones se trata.

Un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. **[8]**

Los repositorios están preparados para distribuirse habitualmente sirviéndose de una red informática como Internet o en un medio físico como un disco compacto. Y pueden ser de acceso público, o pueden estar protegidos y necesitar de una autentificación previa. Los repositorios más conocidos son los de carácter académico y los institucionales. **[8]**

Hoy en día existen varios repositorios de métricas en el mundo que son usados en la recolección de datos estadísticos de un sinnúmero de proyectos. Esto sin duda alguna ayuda a hacer estimaciones más seguras de esfuerzo, tiempo y costo, conocer la productividad, y analizar y reducir los riesgos.

Un ejemplo de ello es el ISBSG (International Software Benchmarking Standards Group) organización que se creó en el año 1997. La misma ha establecido y mantiene dos repositorios de métricas de software, uno de proyectos de desarrollo, reingeniería y mejora con datos de cerca de 3000 proyectos y un nuevo repositorio dedicado a proyectos de mantenimiento y soporte de aplicaciones.

Implantar un repositorio de métricas en la UCI facilitaría la recolección de un gran número de métricas asociadas a los proyectos productivos, pero no sería la solución al problema de desinformación existente respecto al tema. Los repositorios brindan la posibilidad de almacenar datos y lo que se necesita es un sistema que gestione información referente a las métricas, de modo que se facilite su conocimiento y entendimiento.

1.5. Tendencia y tecnologías actuales para el desarrollo de sistemas de gestión.

Los sistemas de gestión representan un eslabón fundamental en el control de la información en el mundo de hoy. El creciente uso de la Internet en los diferentes ámbitos de la vida diaria, ha hecho que un gran número de las herramientas y tecnologías utilizadas giren en torno a elementos como las aplicaciones Web y los sistemas gestores de bases de datos. A continuación se muestran diferentes herramientas, tecnologías y la metodología usada en el desarrollo de la aplicación.

1.5.1. Lenguajes de programación para la Web.

Actualmente existen un gran número de lenguajes que son utilizados a la hora de desarrollar aplicaciones Web. Estos lenguajes se agrupan en dos tipos: los lenguajes de lado del cliente y los lenguajes del lado del servidor.

Entre los lenguajes del lado del servidor podemos encontrar entre los más sobresalientes por el auge que estos han tenido, algunos como PERL, ASP, PHP, Java, JSP, los módulos CGIs e ISAPIs, etc. Estos se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a Bases de Datos, tratamiento de la Información etc. Del lado del cliente se encuentran principalmente el JavaScript y el Visual Basic Script, que son los encargados de aportar dinamismo a la aplicación en los navegadores. **[9]**

1.5.1.1. Active Server Pages (ASP).

Es un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos, aplicaciones Web y servicios XML. Forma parte de la plataforma NET de Microsoft, de lo cual se deduce que es un lenguaje privativo y de un alto costo. Este soporta acceso a bases de datos, trabajo con archivos y carpetas, envío de email, paginación de resultados, procesado de formularios, y muchas otras opciones. **[10]**

En una página ASP podemos incluir casi todo: HTML plano, código de scripting, texto. No hay una distinción formal entre el contenido de una página y su comportamiento. Facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente, hasta la implementación y la configuración de sitios. **[10]**

ASP es un sistema que requiere necesariamente de un servidor Windows, con todas las implicaciones de alto costo, poca flexibilidad y escasa seguridad que estos equipos brindan

1.5.1.2. Java.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Independencia de la plataforma es una de las mayores ventajas del lenguaje. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza su axioma, ‘’’write once, run everywhere’’’

En un sentido estricto, Java no es absolutamente orientado a objetos. Por motivos de eficiencia, Java ha relajado en cierta medida este paradigma y así por ejemplo, no todos los valores son objetos. Su código puede ser a veces redundante en comparación con otros lenguajes. Esto es debido a las frecuentes declaraciones de tipos y conversiones de tipo manual (casting). También se debe a que no se dispone de operadores sobrecargados, y a una sintaxis relativamente simple.

1.5.1.3. Hypertext Preprocessor (PHP).

PHP (siglas que originalmente significaban *Personal Home Page*), es un lenguaje script para el desarrollo de páginas Web dinámicas del lado del servidor. Es Open Source (código abierto), interpretado y de alto nivel, especialmente pensado para desarrollos Web. **[11]**

Es un lenguaje robusto y estable donde se pueden hacer grandes cosas con pocas líneas de código, por lo que es útil su utilización. El código de este es mucho más legible que el de PERL y muy sencillo de aprender, además de que viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor. **[11]**

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos. Este lenguaje es multiplataforma, tiene la capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL, aunque puede interactuar también con motores de bases de datos tales como MSSQL, Oracle, Informix, PostgreSQL, y otros muchos, con un excelente soporte. **[12]**

PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz y está completamente escrito en C, por lo que se ejecuta rápidamente utilizando poca memoria. **[11]**

Este lenguaje soporta en cierta medida la programación orientada a objeto (Clases y herencia) y también realiza de forma automática el análisis léxico para recoger las variables que se pasan en la dirección, lo cual libra al usuario de tener que separar las variables y sus valores. La comprobación de que los parámetros son válidos se hace en el servidor y no en el cliente, de forma que se puede evitar que se reciban solicitudes adulteradas. Además PHP viene equipado con un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos. **[12]**

1.5.1.4. Fundamentación del lenguaje a utilizar.

Luego de un análisis de estos lenguajes y sus características fundamentales, se decidió que PHP resultada el más idóneo para la construcción del sistema.

Entre otras ventajas, PHP es un lenguaje gratuito y multiplataforma que permite al cliente interactuar con la aplicación de forma rápida, segura y eficiente. Además cuenta con una amplia librería de funciones que permiten realizar cualquier tipo de operaciones, como procesamiento de formularios, trabajo con archivos y carpetas, entre otras que le brindan al desarrollador la posibilidad de realizar un producto a la altura de las necesidades del usuario.

1.5.2. Arquitectura.

La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. **[13]**

La misma establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. Su objetivo fundamental es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto.

1.5.2.1. Framework Symfony.

Comenzar desde cero, no es la mejor opción a la hora de desarrollar un software, por lo que el uso de un framework o un marco de trabajo, siempre será una alternativa viable para cualquier solución. En el desarrollo de software un framework no es más que una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

El creciente uso de la programación web y del software libre ha traído consigo que actualmente exista una gran variedad de frameworks orientados a diferentes lenguajes. Symfony es un framework desarrollado íntegramente en PHP 5, diseñado para optimizar el desarrollo de las aplicaciones web. El mismo proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja, además de automatizar las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony posee varias características que lo sitúan entre los frameworks más populares según encuestas de internet: **[14]**

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.

- Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las librerías de otros fabricantes.

Todas estas características han dado al traste con su elección para implementar el sistema, aunque su mayor ventaja radica en la amplia documentación con la que cuenta desde sus inicios en el año 2005, razón que lo ha hecho sobresalir por encima de muchos frameworks usados en la actualidad.

1.5.2.2. El patrón Modelo Vista Controlador.

Symfony está basado en un patrón del diseño Web muy usado en la actualidad conocido como Modelo Vista Controlador (MVC), que está formado por tres niveles:

- El **modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La **vista** transforma el modelo en una página Web que permite al usuario interactuar con ella.
- El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones, en tanto el modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. (Ver anexo 1)

1.5.3. Gestores de bases de datos.

Los Sistemas de gestión de base de datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Entre las ventajas que brindan estos sistemas están:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consultas.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial al terminar los registros.

1.5.3.1. MySQL.

MySql es un servidor multi-hilos de base de datos. El mismo posee un rápido nivel de procesamiento, además de ser confiable, multiproceso, compacto y permitir la creación de bases de datos a código abierto. Es utilizado mayormente cuando se emplean lenguajes de programación como PHP y Perl. Al mismo tiempo es un software multiplataforma, multiusuario y permite elaborar consultas con el robusto SQL.

Entre sus ventajas figura que:

- Consume muy pocos recursos tanto del CPU como de memoria.
- Presenta mejoras en utilidades de administración.
- No hay límites en el tamaño de los registros.
- Gran velocidad en la lectura de datos.

1.5.3.2. PostgreSQL.

PostgreSQL es el servidor de bases de datos de código abierto más utilizado por aquellos programadores que realizan aplicaciones cliente/servidor, complejas o críticas. Es además la alternativa más cercana a MySQL cuando se precisa de operaciones avanzadas como transacciones, procedimientos almacenados, vistas, o cuando se precisa de una base de datos que soporte gran cantidad de información.

Este servidor es muy utilizado actualmente y es una opción económica a SQL Server, pues su costo es menor y las prestaciones son similares. El mismo se puede utilizar sobre cualquier sistema operativo, característica que lo pone por encima de SQL Server y a la par con MySQL.

Algunas de sus principales características son:

- Es libre.
- Alta concurrencia.
- Amplia variedad de tipos nativos.
- Integridad transaccional.
- Herencia de tablas.
- Replicación que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.
- Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby.
- Procedimientos almacenados. **[15]**

1.5.3.3. Oracle.

Oracle es un sistema de gestión de base de datos relacional desarrollado por Oracle Corporation. La gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes por norma general. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de dato como, Access, MySQL, SQL Server, etc.
Se basa en la tecnología cliente/servidor proporcionando todas las ventajas de la misma. Ha sido criticado por la seguridad de su plataforma y su política de parches. A pesar de esto se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

1.5.3.4. MSSQL Server.

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales. Capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Este constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL, Interbase, Firebird o MySQL.

Entre sus principales características se pueden destacar: **[16]**

- Soporte de transacciones.

- Estabilidad, seguridad y escalabilidad
- Soporte de procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los clientes de la red sólo acceden a la información.

Su desventaja radica en que no posee soporte multiplataforma y su distribución no es libre.

1.5.3.5. Fundamentación de la selección del SGBD a utilizar.

Después de realizar un análisis entre los dos SGBD expuestos anteriormente y sus principales características, se decidió optar por PostgreSQL dado que ofrece muchas ventajas respecto a otros sistemas de bases de datos.

PostgreSQL es multiplataforma, extensible y está diseñado para ambientes de alto volumen. Además de su estabilidad y confiabilidad, cuenta con herramientas gráficas de diseño y administración de bases de datos.

1.5.4. Servidor Web utilizado.

Un servidor Web es un programa que implementa el protocolo HTTP (Hypertext Transfer Protocol). Se encarga de atender las peticiones HTTP llevada a cabo por un cliente y responder con el contenido que el cliente solicita. Este protocolo HTTP está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML.

Entre los servidores Web más usados a nivel mundial están el Apache y el Internet Information Server (IIS). Debido a las características que presenta el sistema que se desea construir y a la factibilidad del servidor Web Apache, el mismo ha sido el seleccionado para llevar acabo el desarrollo de la aplicación.

Las principales características de Apache son: **[17]**

- Funcionalidad en múltiples plataformas.
- Elaborado índice de directorios.
- Soporte del último protocolo http 1.1.
- Sencilla administración basada en la configuración de un único archivo.

- Soporte para CGI (Common Gateway Interface) y FastCGI.
- Mensajes de error altamente configurables.
- Bases de datos de autenticación y negociado de contenido.

El servidor Apache ofrece una perfecta combinación entre estabilidad y sencillez, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma y se distribuye en casi todas las implementaciones de Linux.

1.5.5. Metodologías de desarrollo de software.

Ningún proceso de desarrollo de software está exento de riesgos, pero si a pesar de esto no se lleva una metodología adecuada, el resultado final serán clientes y/o desarrolladores insatisfechos. Una metodología define quién hace qué, cuándo y cómo lo hace. Antes de elegir la metodología que se usará para la implementación de un software lo más importante es determinar el alcance que tendrá y ver cual es la que más se acomoda a la aplicación.

1.5.5.1. Extreme Programming (XP).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. **[18]**

1.5.5.2. SCRUM 5.

Esta metodología está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

1.5.5.3. Proceso Unificado de Desarrollo de Software (RUP).

RUP fue publicado en 1998 como resultado de varios años de experiencia y unifica los mejores elementos de metodologías anteriores. Es una propuesta de proceso para el desarrollo de software orientado a objetos que utiliza Unified Modeling Language (UML) como lenguaje de representación visual, y que además cubre el ciclo de vida de desarrollo de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software.

El ciclo de vida de RUP se caracteriza por estar: **[19]**

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.
- **Iterativo e Incremental:** RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y las cuales se definen según el nivel de madurez que alcanzan los productos que se van obteniendo con cada actividad ejecutada. La terminación de cada fase ocurre en el hito correspondiente a cada una, donde se evalúa que se hayan cumplido los objetivos de la fase en cuestión.

1.5.5.4. Fundamentación de la metodología a utilizar.

Luego de analizar las metodologías anteriores y de acuerdo con la necesidad de optar por una metodología fuerte, se decidió que el Proceso Unificado de Desarrollo de Software (RUP) será la metodología de Ingeniería de Software que guiará el desarrollo de la aplicación. Además de ser una de las metodologías más utilizadas para el análisis implementación y documentación de sistemas orientados a

objetos, RUP aporta un gran número de facilidades y la posibilidad de ser adaptada a un proyecto específico.

1.5.6. Herramientas utilizadas.

Para llevar a cabo el sistema se hace indispensable tener en cuenta la utilización de algunas herramientas necesarias en el diseño de interfaz, el trabajo con las imágenes y el modelado del proyecto.

1.5.6.1. Adobe Dreamweaver CS4.

Es una aplicación en forma de estudio basada en la forma de estudio de Adobe Flash pero con más parecido a un taller destinado para la construcción y edición de sitios y aplicaciones Web basados en estándares. Fue creado inicialmente por Macromedia y actualmente es producido por Adobe Systems.

Por sus funcionalidades, soporte tanto para edición de imágenes, como para animación a través de su integración con otras herramientas y recientemente por su soporte de los estándares del World Wide Web Consortium, Dreamweaver es el programa de su tipo más utilizado en el sector del diseño y la programación web.

1.5.6.2. Zend Studio.

Zend Studio es uno de los ambientes de desarrollo integrado o Integrated Development Environment (IDE) disponible para desarrolladores y agrupa todos los componentes necesarios para el desarrollo de aplicaciones usando PHP.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. Existen versiones del producto para Windows, Linux y MacOS. **[20]**

Se considera a Zend Studio el entorno IDE más capacitado y con más utilidades cuando es utilizado PHP como lenguaje de programación. Este programa dispone de opciones pensadas con acierto por personas que dominan como nadie la tecnología, de modo que este programa es el idóneo

1.5.6.3. Rational Rose.

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. **[21]**

Actualmente es la herramienta líder de su tipo en el mundo. Integra todos los elementos que propone la metodología RUP para cubrir el ciclo de vida de un proyecto. Además proporciona mecanismos para realizar Ingeniería Inversa y utiliza un proceso de desarrollo iterativo controlado que permite que haya varias personas trabajando a la vez, posibilitando que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

1.6. Conclusiones.

Con este capítulo se han esclarecido una serie de conceptos y elementos teóricos que son de gran importancia para comprender el contenido del trabajo. También se realizó un análisis de las herramientas y tecnologías vigentes para la construcción de aplicaciones sobre la Web, que permitió la selección de las mismas para el desarrollo.

Finalmente fue seleccionado como lenguaje de programación PHP y como SGBD PostgreSQL. Se escogió el Zend Studio como herramienta principal para la implementación y el Dreamweaver para el diseño de las interfaces de usuario. Se definió además como metodología de desarrollo para guiar el proceso a RUP y como herramienta para el modelado el Rational Rose.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

El presente capítulo identifica y fundamenta las características de la solución propuesta, especificando aspectos como el objeto de estudio y el flujo actual del proceso a informatizar. Se listan los requisitos funcionales y no funcionales que debe tener el sistema propuesto, permitiendo tener una concepción general del mismo, además de describirlo en términos de actores, casos de uso y sus relaciones.

2.2. Objeto de estudio.

La UCI como empresa productora de software debe velar por la calidad de los productos que se desarrollan en cada proyecto productivo. Para esto cuenta con la Dirección de Calidad de Software (DCS) la cual se encarga de estandarizar y asegurar el proceso de desarrollo para lograr un resultado optimo. El Grupo de Métricas (GM) de esta dirección tiene dentro sus principales directrices de trabajo la gestión y aplicación de las métricas de software en cada proyecto de la universidad, pudiendo así controlar lo que acontece en los mismos, además de poder mejorar los procesos y la calidad de sus productos.

Precisamente, el objeto de estudio de este trabajo lo constituye el proceso de gestión de métricas. Dicho proceso actualmente no se está realizando en la UCI. Del mismo modo la universidad no cuenta con un mecanismo informático que permita realizar dicho proceso de forma automática. Tampoco existe un espacio que mantenga a los proyectos actualizados constantemente sobre los cambios que ocurren en la concepción de las mediciones, lo que sin duda debuta en la desinformación de los implicados.

2.3. Descripción del proceso actual.

Actualmente en la UCI no se aplica un proceso de medición que arroje resultados y permita analizar los mismos para tomar decisiones y mejorar los procesos en cada proyecto productivo. Algunos equipos han realizado estimaciones y han aplicado métricas de forma individual para resolver un problema determinado, pero ninguno a seguido un mecanismo común, y otros nunca han aplicado una métrica, ni conocen o dominan los beneficios de esta terminología. De manera general esto ha provocado un desbalance a la hora de tomar decisiones en cada proyecto, corroborándose en la calidad de los resultados y en la incrementación de los riesgos de desarrollo.

2.4. Objeto de informatización.

Uno de los problemas de la estandarización del proceso de medición en la UCI radica en que el GM de la DCS no cuenta con una herramienta informática que le brinde la facilidad de tener un espacio para mantener informados a todos los proyectos de la universidad de forma simultanea. Por tal razón se le dificulta a este grupo hacerle llegar a cada equipo de desarrollo las últimas definiciones respecto a las mediciones, así como las particularidades que puede tener el proceso de medición.

En estos momentos la UCI se encuentra en un proceso de mejora para alcanzar el nivel dos de CMMI. CMMI (Capability Maturity Model Integration) es un modelo para la mejora de procesos que en cualquiera de sus representaciones, ya sea escalonada o continua, proporciona elementos esenciales para procesos eficaces. Una de las características de la representación escalonada es que clasifica las empresas en niveles de madurez, los cuales a su vez agrupan veintidós aéreas de proceso.

El proceso de medición se pretende aplicar según el área de proceso de Medición y Análisis (MA) del nivel dos; por lo que se hace necesario un control riguroso de las acciones a tomar para poder alcanzar dicho nivel.

MA de CMMI tiene como objetivo desarrollar y sostener una capacidad de medición que sea usada para ayudar a las necesidades de información de la gerencia. Se hace necesario entonces una aplicación informática que ayude a resolver la problemática planteada, formalice el proceso actual y de respuesta a las principales exigencias de MA, las cuales se sintetizan a continuación:

- Alinear las actividades de medición y análisis.
  - Establecer los objetivos de la medición
  - Especificar las métricas.
  - Especificar los procedimientos de recolección y almacenamiento de los datos.
  - Especificar los procedimientos del análisis.

- Proporcionar los resultados de la medición.
  - Recolectar los datos de la medición.
  - Analizar los datos de la medición.
  - Almacenar los datos y los resultados de la medición.
  - Comunicar los resultados.

Según CMMI la especificación explícita de los métodos de recolección ayuda a asegurar que los datos estén recogidos correctamente y también puede ayudar en el esclarecimiento de necesidades de información y los objetivos de la medición. Este criterio, tributa concretamente a especificar cómo los datos de la medición serán obtenidos y almacenados, lo que a su vez constituye una de las prácticas específicas de MA. La aplicación que se propone permite el cumplimiento de esta práctica, y con ella, estará disponible para los responsables de hacer el trabajo, la clara y concisa orientación sobre procedimientos correctos.

2.5. Propuesta del sistema.

Para dar solución a la situación problémica actual, se propone elaborar una aplicación web a la cual tendrán acceso para consultar la información disponible todos los proyectos de la universidad. El GM de la DCS será el encargado de administrar el mismo y mantenerlo actualizado. El sistema tendrá como objetivo primordial la gestión de métricas y contará con dos aplicaciones internas, una administrativa y otra informativa para el uso de los líderes de proyecto quienes serán los encargados de hacer utilización de dichas métricas para la evaluación posterior del software en desarrollo.

La parte administrativa se encargará de insertar, modificar y eliminar métricas, ya sean derivadas o bases. El sistema también garantizará la gestión de los tipos de proyectos a los cuales van estar dirigidos las métricas, la clasificación de las mismas y las metas a las cuales van a tributar una vez sean insertadas en el sistema de medición, además de gestionar todo el contenido que será visto por el usuario en la página principal de la aplicación informativa. Por último, pero no menos importante esta aplicación admitirá la gestión de los usuarios que sean necesarios para administrarla, en tanto la parte informativa además de mostrar las informaciones de la portada, permitirá a la comunidad consultar las métricas existentes en la base de datos, así como filtrarlas, de acuerdo su tipo, clasificación y las metas que satisfacen.

La arquitectura del sistema responderá al MVC, el cual estructura el sistema de una forma lógica para que responda a los requisitos funcionales para los cuales fue creado.

Todo el proceso de gestión anterior y el sistema en sí, tributarán de una forma u otra a las exigencias de MA del nivel de dos de CMMI, por lo que la aplicación tendrá un impacto positivo en los procesos de mejoras que se realizan en la producción de software de la UCI.

2.6. Especificación de los requerimientos de software.

Según RUP la captura de requisitos es el proceso de averiguar normalmente en circunstancias difíciles, lo que se debe construir. Los requisitos son las condiciones o capacidades que deben ser alcanzadas o poseídas para que un sistema satisfaga las necesidades para las cuales fue creado. En otras palabras estos definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema (es decir las condiciones o capacidades que el sistema debe cumplir) suficiente buena como para que pueda llegarse aun acuerdo entre el cliente (incluyendo a los usuarios) y a los desarrolladores sobre que debe hacer y que no debe hacer el sistema. **[22]**

2.6.1. Requerimientos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Para satisfacer los objetivos de este sistema el mismo debe ser capaz de:

RF1. Gestionar métrica.

1.1. Adicionar métrica derivada.

1.1.1. Los datos a adicionar son:

- Nombre.
- Tipo. (Producto, Proyecto, Proceso)
- Clasificación.
- Definición.
- Fórmula.
- Gráfico.
- Procedimiento de recolección.
- Métricas base necesarias.
- Metas asociadas.
- Tipos de proyecto asociados.

1.2. Adicionar métrica base.

1.2.1. Los datos a adicionar son:

- Nombre.
- Tipo. (Producto, Proyecto, Proceso)
- Clasificación.
- Definición.
- Fórmula.
- Gráfico.
- Procedimiento de recolección.
- Metas asociadas.
- Tipos de proyecto asociados.

1.3. Modificar métrica derivada.

1.3.1. Los datos a modificar son:

- Nombre.
- Tipo. (Producto, Proyecto, Proceso)
- Clasificación.
- Definición.
- Fórmula.
- Gráfico.
- Procedimiento de recolección.
- Métricas base necesarias.
- Metas asociadas.
- Tipos de proyecto asociados.

1.4. Modificar métrica base.

1.4.1. Los datos a modificar son:

- Nombre.
- Tipo. (Producto, Proyecto, Proceso)
- Clasificación.
- Definición.
- Fórmula.
- Gráfico.
- Procedimiento de recolección.
- Metas asociadas.
- Tipos de proyecto asociados.

1.5. Eliminar métrica.

RF2. Gestionar nomenclador metas.

2.1. Adicionar metas.

2.1.1. Los datos a adicionar son:

- Nomenclador.

2.2. Modificar metas.

2.2.1. Los datos a modificar son:

- Nomenclador.

2.3. Eliminar metas.

RF3. Gestionar nomenclador clasificación.

3.1. Adicionar clasificación.

3.1.1. Los datos a adicionar son:

- Tipo. (Producto, Proyecto, Proceso)
- Nomenclador.

3.2. Modificar clasificación.

3.2.1. Los datos a modificar son:

- Tipo. (Producto, Proyecto, Proceso)
- Nomenclador.

3.3. Eliminar clasificación.

RF4. Gestionar nomenclador tipos de proyecto.

4.1. Adicionar tipo de proyecto.

4.1.1. Los datos a adicionar son:

- Nomenclador.

4.2. Modificar tipo de proyecto.

4.2.1. Los datos a modificar son:

- Nomenclador.

4.3. Eliminar tipo de proyecto.

RF5. Gestionar usuarios.

5.1. Adicionar usuario.

5.1.1. Los datos a adicionar son:

- Nombre.
- Usuario.
- Contraseña.

5.2. Modificar usuario.

5.2.1. Los datos a modificar son:

- Nombre.
- Usuario.
- Contraseña.

5.3. Eliminar usuario.

RF6. Autenticar usuario.

RF7. Gestionar contenido informativo.

7.1. Adicionar contenido informativo.

7.1.1. Los datos a adicionar son:

- Título.
- Texto del contenido.

7.2. Modificar contenido informativo.

7.2.1. Los datos a modificar son:

- Título.
- Texto del contenido.

7.3. Eliminar contenido informativo.

RF8. Mostrar contenido informativo.

RF9. Mostrar métricas.

9.1. Mostrar últimas métricas.

9.2. Filtrar métricas por tipo, clasificación, meta y tipo de proyecto.

9.2.1. Los datos a introducir son:

- Tipo.
- Clasificación.
- Meta.
- Tipo de proyecto.

RF10. Buscar métricas.

10.1. El dato a introducir es:

- Nombre.

10.2. Los campos a mostrar son:

- Nombre.
- Tipo.
- Clasificación.

2.6.2. Requerimientos no funcionales.

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

2.6.2.1. Requerimientos de software.

- Para el funcionamiento del sistema en el servidor el mismo deberá contar con Sistema Operativo Microsoft Windows Server 2003 o superior; Sistema Operativo Linux en sus versiones servidores.
- Para el funcionamiento del sistema en las computadoras clientes, las mismas deberán contar Sistema Operativo Microsoft Windows XP o superior; Sistema Operativo Linux con ambiente grafico KDE o GNOME.
- Optimizado para Mozilla Firefox en sus versiones 3.0 o superior.

2.6.2.2. Requerimientos de hardware.

- Se requiere un mínimo de 256 MB de RAM, 512 MB recomendado, y 1.3 GHz de velocidad de procesamiento, recomendado 3.0 GHz.

2.6.2.3. Restricciones en el diseño y la implementación.

- La aplicación se desarrollará utilizando el lenguaje de programación PHP, Servidor Web Apache y como SGBD PostgreSQL.

2.6.2.4. Apariencia o interfaz externa.

- Interfaz con pocas imágenes para no demorar las respuestas al usuario.
- Diseño sencillo y claro, con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus funcionalidades.

2.6.2.5. Usabilidad.

- La información que brinda el sistema puede ser consultada por cualquier persona, que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
- El sistema podrá ser administrado solo por las personas que conozcan bien el negocio que el mismo implementa.

2.6.2.6. Rendimiento.

- Disponibilidad constante de trabajo en red contra el servidor.
- Rápido acceso a la información en tiempos relativamente cortos.
- El sistema debe requerir un consumo mínimo de recursos.

2.6.2.7. Mantenimiento.

- Se requiere que el producto reciba mantenimiento ante cualquier fallo que ocurra.

2.6.2.8. Seguridad.

- El sistema solo puede ser administrado por la persona que cuenta con los permisos necesarios para hacerlo.
- La información se valida tanto en el cliente como en el servidor, no obstante los usuarios acceden de manera rápida y operativa al sistema sin que los requerimientos de seguridad se conviertan en un retardo para ellos.

2.6.2.9. Confidencialidad.

- Toda la información está protegida del acceso no autorizado.

2.6.2.10. Disponibilidad.

- Se garantiza a los usuarios del sistema el acceso a la información solicitada en todo momento.

2.6.2.11. Confiabilidad.

- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

2.6.2.12. Portabilidad.

- El sistema es independiente de plataforma, podrá ser usado bajo los sistemas operativos Windows y Linux.

2.7. Modelo de casos de uso del sistema.

RUP plantea que un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. El modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre condiciones y capacidades que debe cumplir el

sistema. El mismo sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas.

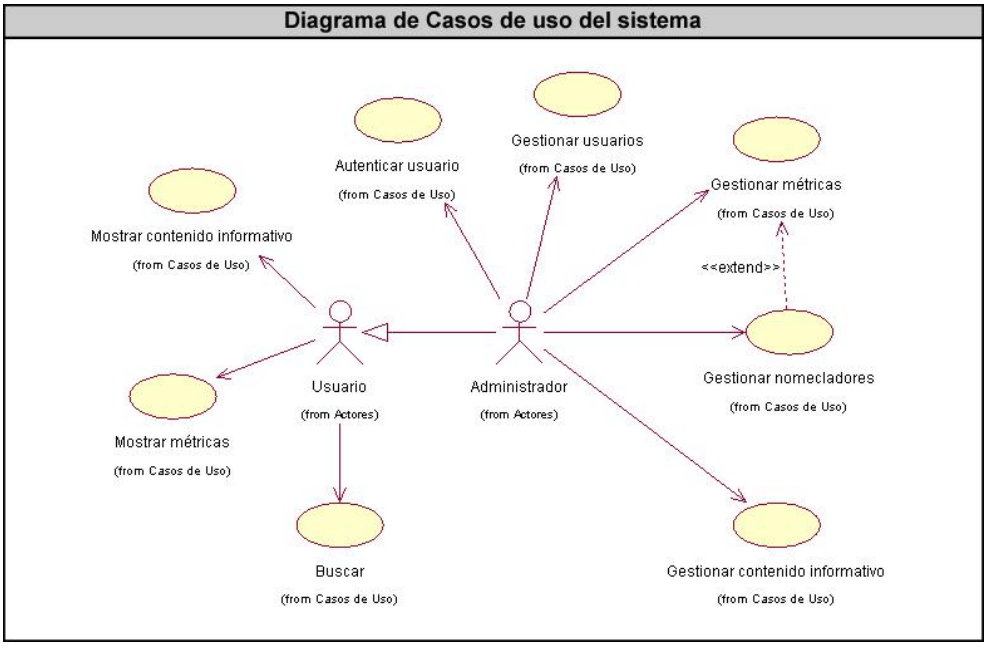
2.7.1. Definición de los actores.

Un actor es un rol de un usuario, que puede intercambiar información con el sistema o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina. Con el sistema interactúan dos actores que se definen a continuación:

Actores	Justificación
Usuario	Generaliza todas las personas que dentro de la universidad o más directamente desde un proyecto productivo pueden consultar el sistema para obtener la información requerida del mismo.
Administrador	Representa a una persona que configura y controla el comportamiento del sistema. Es el encargado de administrar todos los procesos de gestión que implementa el sistema, así como mantenerlo actualizado.

2.7.2. Diagrama de casos de uso.

El diagrama de casos de uso muestra la interacción de los actores y los casos de uso, y como se relacionan entre sí los casos de uso.



2.7.3. Casos de uso expandidos.

2.7.3.1. CU Autenticar usuario.

Caso de Uso	Autenticar usuario
Actores	Administrador
Propósito	El objetivo de este caso de uso es que el actor se autentique.
Resumen	El CU se inicia cuando el actor introduce sus datos para entrar al sistema.
CU asociados	_
Precondiciones	El actor debe encontrarse en la página de autenticación.
Poscondiciones	El actor accede o no al sistema, en dependencia de la validez de los datos
Referencias	RF6
Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor introduce su usuario y su contraseña y presiona el botón **Aceptar**.	2. El sistema verifica que las credenciales sean correctas y redirecciona al actor hacía la página de administración.
Cursos Alternos	
Línea 2: 1. Si los datos introducidos son incorrectos o nulos el sistema muestra un mensaje indicando que ha fallado la autenticación.	

2.7.3.2. CU Gestionar contenido informativo.

Caso de Uso	Gestionar contenido informativo.
Actores	Administrador
Propósito	El objetivo de este caso de uso es adicionar, modificar y eliminar los contenidos informativos que se muestran en la portada.
Resumen	El CU se inicia cuando el actor selecciona el menú Gestionar contenido informativo y se listan los artículos existentes, consiguiendo así adicionar, modificar o eliminar un artículo.
CU asociados	_

Precondiciones	Es necesario que el actor tenga los permisos necesarios para ejecutar el CU.
Poscondiciones	Quedarán actualizados los artículos de contenido de la portada.
Referencias	RF7

Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona el menú **Informaciones/Gestionar Artículos**.	2. El sistema lista todos los artículos existentes.
3. El actor selecciona el vínculo Nuevo. • Si el actor selecciona la opción Modificar (Ver **Sección I**) • Si el actor selecciona la opción Eliminar (Ver **Sección II**)	4. El sistema muestra una interfaz con los campos necesarios para adicionar un nuevo artículo.
5. El actor llena los campos con los datos correspondientes: • Título. • Texto del contenido. Presiona el botón **Guardar**.	6. El sistema muestra un mensaje indicando que el artículo ha sido guardado correctamente y que puede añadir un nuevo artículo.
Cursos Alternos	
Línea 1: 2. El actor selecciona el menú **Informaciones/Nuevo**. Línea 5: 3. El actor presiona el botón **Aplicar**, el sistema registra el artículo y lo carga en modo edición (Ver **Sección I**) 4. El actor presiona el vínculo **Cancelar**.	
Sección (I) Modificar	
Acción del Actor	**Respuesta del Sistema**

1. El actor selecciona la opción modificar del artículo que desea modificar o da clic sobre el título del mismo.	2. El sistema muestra una interfaz con los datos que posee el artículo.
3. El actor modifica el artículo y presiona el botón **Guardar**.	4. El sistema muestra un mensaje donde indica que el artículo fue actualizado satisfactoriamente.
5. El actor presiona el botón **Ver listado**.	6. El sistema actualiza la lista de artículos.
Cursos Alternos	
Línea 3: 5. En caso de que el actor no desee realizar por alguna razón la operación podrá presionar el vínculo **Ver listado**. 6. En caso de que el actor desee eliminar el artículo podrá presionar el vínculo **Eliminar**.	
Sección (II) Eliminar	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción eliminar del artículo que desea eliminar.	2. El sistema muestra un mensaje de confirmación.
3. El actor presiona el botón Aceptar.	4. El sistema muestra un mensaje indicando que el elemento fue borrado satisfactoriamente y actualiza la lista de artículos.
Cursos Alternos	
Línea 1: 7. En caso de que el actor desee eliminar más de un elemento, puede seleccionar los elementos en cuestión, escoger la opción **Eliminar** y presionar el botón **Aplicar**. Línea: 3: 8. El actor presiona el botón **Cancelar**.	

2.7.3.3. CU Gestionar métrica.

<table>
<tr><td>Caso de Uso</td><td colspan="2">Gestionar métrica</td></tr>
<tr><td>Actores</td><td colspan="2">Administrador</td></tr>
<tr><td>Propósito</td><td colspan="2">El objetivo de este caso de uso es adicionar, modificar y eliminar métricas.</td></tr>
<tr><td>Resumen</td><td colspan="2">El CU se inicia cuando el actor selecciona el menú Gestionar métricas ya sean base o derivadas y se listan las Métricas existentes, consiguiendo así adicionar, modificar o eliminar una métrica.</td></tr>
<tr><td>CU asociados</td><td colspan="2">Gestionar nomencladores (extend)</td></tr>
<tr><td>Precondiciones</td><td colspan="2">Es necesario que el actor tenga los permisos necesarios para ejecutar el CU.</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">Quedarán actualizadas las Métricas en el sistema.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF1</td></tr>
<tr><td colspan="3">Descripción</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El actor selecciona en el menú Métricas cualquiera del dos tipos ya sean base o derivadas, la opción Gestionar métricas.</td><td>2. El sistema lista las métricas existentes.</td></tr>
<tr><td colspan="2">3. El actor selecciona el vínculo Nuevo.
• Si el actor selecciona la opción Modificar (Ver Sección I)
• Si el actor selecciona la opción Eliminar (Ver Sección II)</td><td>4. El sistema muestra una interfaz con los campos necesarios para adicionar una nueva métrica.
• Si la métrica es base (Ver Sección III)
• Si la métrica es derivada (Ver Sección IV)</td></tr>
<tr><td colspan="3">Cursos Alternos</td></tr>
<tr><td colspan="3">Línea 1:
• El actor selecciona la opción Nuevo de cualquiera de los dos tipos de métricas.</td></tr>
<tr><td colspan="3">Sección (I) Modificar</td></tr>
</table>

Acción del Actor	Respuesta del Sistema
1. El actor selecciona la opción modificar de la métrica que desea modificar o da clic sobre el nombre de la misma.	2. El sistema muestra una interfaz con los datos que posee la métrica.
3. El actor modifica los datos de la métrica y presiona el botón **Guardar**.	4. El sistema muestra un mensaje donde indica que la métrica fue actualizada satisfactoriamente.
5. El actor presiona el botón **Ver listado**.	6. El sistema actualiza la lista de métricas.
Cursos Alternos	
Línea 3: • En caso de que el actor no desee realizar por alguna razón la operación podrá presionar el vínculo **Ver listado**. • En caso de que el actor desee eliminar la métrica podrá presionar el vínculo **Eliminar**.	
Sección (II) Eliminar	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción eliminar de la métrica que desea eliminar.	2. El sistema muestra un mensaje de confirmación.
3. El actor presiona el botón **Aceptar**.	4. El sistema muestra un mensaje indicando que el elemento fue borrado satisfactoriamente y actualiza la lista de métricas.
Cursos Alternos	
Línea 1: • En caso de que el actor desee eliminar más de un elemento, puede seleccionar los elementos en cuestión, escoger la opción **Eliminar** y presionar el botón **Aplicar**. Línea: 3: • El actor presiona el botón **Cancelar**.	
Sección (III) Adicionar métrica derivada	

<table>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El actor llena los campos con los datos correspondientes:
• Nombre.
• Tipo. (Producto, Proyecto, Proceso)
• Clasificación.
• Definición.
• Fórmula.
• Gráfico.
• Procedimiento de recolección.
• Métricas base necesarias.
• Metas asociadas.
• Tipos de proyecto asociados.
Presiona el botón Guardar.</td><td>2. El sistema muestra un mensaje indicando que la métrica ha sido guardada correctamente y que puede añadir una nueva métrica.</td></tr>
<tr><th colspan="2">Cursos Alternos</th></tr>
<tr><td colspan="2">Línea 1:
• El actor presiona el botón Aplicar, el sistema registra la métrica y la carga en modo edición (Ver Sección I)
• El actor presiona el vínculo Cancelar.</td></tr>
<tr><th colspan="2">Sección (IV) Adicionar métrica base</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El actor llena los campos con los datos correspondientes:
• Nombre.
• Tipo. (Producto, Proyecto, Proceso)
• Clasificación.
• Definición.
• Fórmula.</td><td>2. El sistema muestra un mensaje indicando que la métrica ha sido guardada correctamente y que puede añadir una nueva métrica.</td></tr>
</table>

• Gráfico. • Procedimiento de recolección.. • Metas asociadas. • Tipos de proyecto asociados. Presiona el botón **Guardar**.	
Cursos Alternos	
Línea 1: • El actor presiona el botón **Aplicar**, el sistema registra la métrica y la carga en modo edición (Ver **Sección I**) • El actor presiona el vínculo **Cancelar**.	

2.7.3.4. CU Gestionar nomencladores.

Caso de Uso	Gestionar nomencladores.
Actores	Administrador.
Propósito	El objetivo de este caso de uso es adicionar, modificar y eliminar un nomenclador.
Resumen	El CU se inicia cuando el actor selecciona la opción Gestionar de cualquiera de los nomencladores (clasificación, meta, tipo de proyecto) y se listan los nomencladores existentes, consiguiendo así adicionar, modificar o eliminar un nomenclador.
CU asociados	-
Precondiciones	Es necesario que el actor tenga los permisos necesarios para ejecutar el CU.
Poscondiciones	Quedarán actualizados los nomencladores en el sistema.
Referencias	RF2, RF3, RF4
Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona en el menú **Nomencladores** la opción	2. El sistema lista las métricas existentes.

Gestionar de cualquiera de los tres submenús que aparecen.	
3. El actor selecciona el vínculo **Nuevo**. • Si el actor selecciona la opción Modificar (Ver **Sección I**) • Si el actor selecciona la opción Eliminar (Ver **Sección II**)	4. El sistema muestra una interfaz con los campos necesarios para adicionar una nuevo nomenclador. • Si el nomenclador es una meta(Ver **Sección III**) • Si el nomenclador es una clasificación (Ver **Sección IV**) • Si el nomenclador es un tipo de proyecto (Ver **Sección V**)
Cursos Alternos	
Línea 1: • El actor selecciona la opción **Nuevo** de cualquiera de los tres submenús.	
Sección (I) Modificar	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción modificar del nomenclador que desea modificar o da clic sobre el nombre del mismo.	2. El sistema muestra una interfaz con los datos que posee el nomenclador.
3. El actor modifica los datos del nomenclador y presiona el botón **Guardar**.	4. El sistema muestra un mensaje donde indica que el elemento fue actualizado satisfactoriamente.
5. El actor presiona el botón **Ver listado**.	6. El sistema actualiza la lista de nomencladores.
Cursos Alternos	
Línea 3: • En caso de que el actor no desee realizar por alguna razón la operación podrá presionar el vínculo **Ver listado**. • En caso de que el actor desee eliminar la métrica podrá presionar el vínculo **Eliminar**.	
Sección (II) Eliminar	

Acción del Actor	Respuesta del Sistema
1. El actor selecciona la opción eliminar del nomenclador que desea eliminar.	2. El sistema muestra un mensaje de confirmación.
3. El actor presiona el botón **Aceptar**.	4. El sistema muestra un mensaje indicando que el elemento fue borrado satisfactoriamente y actualiza la lista de nomencladores.
Cursos Alternos	
Línea 1: • En caso de que el actor desee eliminar más de un elemento, puede seleccionar los elementos en cuestión, escoger la opción **Eliminar** y presionar el botón **Aplicar**. Línea: 3: • El actor presiona el botón **Cancelar**.	
Sección (III) Adicionar meta	
Acción del Actor	**Respuesta del Sistema**
1. El actor llena los campos con los datos correspondientes: • Nombre. Presiona el botón **Guardar**.	2. El sistema muestra un mensaje indicando que la meta ha sido guardada correctamente y que puede añadir una nueva meta.
Cursos Alternos	
Línea 1: • El actor presiona el botón **Aplicar**, el sistema registra la meta y la carga en modo edición (Ver **Sección I**) • El actor presiona el vínculo **Cancelar**.	
Sección (IV) Adicionar clasificación	
Acción del Actor	**Respuesta del Sistema**
1. El actor llena los campos con los datos correspondientes: • Tipo. • Nombre. Presiona el botón **Guardar**.	2. El sistema muestra un mensaje indicando que la clasificación ha sido guardada correctamente y que puede añadir una nueva clasificación.

Cursos Alternos	
Línea 1: • El actor presiona el botón **Aplicar**, el sistema registra la clasificación y la carga en modo edición (Ver **Sección I**) • El actor presiona el vínculo **Cancelar**.	
Sección (IV) Adicionar clasificación	
Acción del Actor	**Respuesta del Sistema**
1. El actor llena los campos con los datos correspondientes: • Nombre. Presiona el botón **Guardar**.	2. El sistema muestra un mensaje indicando que el elemento ha sido guardado correctamente y que puede añadir un nueva tipo de proyecto.
Cursos Alternos	
Línea 1: • El actor presiona el botón **Aplicar**, el sistema registra el tipo de proyecto y lo carga en modo edición (Ver **Sección I**) • El actor presiona el vínculo **Cancelar**.	

2.7.3.5. CU Gestionar usuarios.

Caso de Uso	Gestionar usuarios.
Actores	Administrador.
Propósito	El objetivo de este caso de uso es adicionar, modificar y eliminar usuarios.
Resumen	El CU se inicia cuando el actor selecciona el menú Gestionar usuarios y se listan las usuarios existentes, consiguiendo así adicionar, modificar o eliminar un usuario.
CU asociados	-
Precondiciones	Es necesario que el actor tenga los permisos necesarios para ejecutar el CU.
Poscondiciones	Quedarán actualizados los usuarios en el sistema.
Referencias	RF5
Descripción	
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El actor selecciona el menú **Usuarios/Gestionar usuarios**.	2. El sistema lista todos los usuarios existentes.
3. El actor selecciona el vínculo **Nuevo**. • Si el actor selecciona la opción Modificar (Ver **Sección I**) • Si el actor selecciona la opción Eliminar (Ver **Sección II**)	4. El sistema muestra una interfaz con los campos necesarios para adicionar un nuevo usuario.
5. El actor llena los campos con los datos correspondientes: • Nombre. • Usuario. • Contraseña. Presiona el botón **Guardar**.	6. El sistema muestra un mensaje indicando que el usuario ha sido guardado correctamente y que puede añadir un nuevo usuario.
Cursos Alternos	
Línea 1: • El actor selecciona el menú **Usuarios/Nuevo**. Línea 5: • El actor presiona el botón **Aplicar**, el sistema registra el usuario y lo carga en modo edición (Ver **Sección I**) • El actor presiona el vínculo **Cancelar**.	
Sección (I) Modificar	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción modificar del usuario que desea modificar o da clic sobre el nombre de usuario del mismo.	2. El sistema muestra una interfaz con los datos que posee el usuario.
3. El actor modifica el usuario y	4. El sistema muestra un mensaje donde

presiona el botón **Guardar**.	indica que el usuario fue actualizado satisfactoriamente.
5. El actor presiona el botón **Ver listado**.	6. El sistema actualiza la lista de usuarios.
Cursos Alternos	
Línea 3: • En caso de que el actor no desee realizar por alguna razón la operación podrá presionar el vínculo **Ver listado**. • En caso de que el actor desee eliminar el artículo podrá presionar el vínculo **Eliminar**.	
Sección (II) Eliminar	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción eliminar del usuario que desea eliminar.	2. El sistema muestra un mensaje de confirmación.
3. El actor presiona el botón Aceptar.	4. El sistema muestra un mensaje indicando que el elemento fue borrado satisfactoriamente y actualiza la lista de usuarios.
Cursos Alternos	
Línea 1: • En caso de que el actor desee eliminar más de un elemento, puede seleccionar los elementos en cuestión, escoger la opción **Eliminar** y presionar el botón **Aplicar**. Línea: 3: • El actor presiona el botón **Cancelar**.	

2.7.3.6. CU Mostrar contenido informativo.

Caso de Uso	Mostrar contenido informativo.
Actores	Usuario
Propósito	El objetivo de este caso de uso es mostrar en la portada los artículos de contenido que han sido añadidos al sistema.
Resumen	El CU se inicia cuando el actor entra al sitio y se muestra la portada del

	mismo.
CU asociados	
Precondiciones	Debe existir al menos un artículo de contenido en el sistema.
Poscondiciones	_
Referencias	RF8

Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor accede a la página principal del sitio.	2. El sistema muestra el último artículo de contenido que ha sido añadido al sistema y una lista del resto de las informaciones que han sido publicadas.
3. El actor selecciona cualquiera de los artículos de la lista.	4. El sistema muestra el artículo.

2.7.3.7. CU Mostrar métricas.

Caso de Uso	Mostrar métricas
Actores	Usuario
Propósito	El objetivo de este caso de uso es mostrar las métricas que han sido añadidas al sistema.
Resumen	El CU se inicia cuando el actor selecciona cualquiera de las opciones del menú Mostrar.
CU asociados	_
Precondiciones	Debe existir al menos una métrica en el sistema.
Poscondiciones	_
Referencias	RF9

Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción Últimas métricas.	2. El sistema lista las últimas cinco métricas añadidas al sistema.

• Si el actor selecciona la opción Filtrar métricas (Ver **Sección I**)	
3. El actor da clic sobre la métrica deseada.	4. El sistema muestra la métrica seleccionada con todos sus detalles.
Cursos Alternos	
Línea 4: • En caso de que el actor no desee ver los detalles de la métrica, puede dar clic en el botón **Cerrar**.	
Sección (I) Filtrar métricas.	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona los filtros que desea para la búsqueda: • Tipo. • Clasificación. • Meta. • Tipo de proyecto.	2. El sistema realiza la búsqueda y muestra las métricas correspondientes.
3. El actor da clic sobre la métrica deseada.	4. El sistema muestra la métrica seleccionada con todos sus detalles.
Cursos Alternos	
Línea 4: • En caso de que el actor no desee ver los detalles de la métrica, puede dar clic en el botón **Cerrar**.	

2.7.3.8. CU Buscar.

Caso de Uso	Buscar
Actores	Usuario
Propósito	El objetivo de este caso de uso permitir al usuario realizar una búsqueda al azar, introduciendo por teclado una cadena de caracteres.
Resumen	El CU se inicia cuando el actor selecciona escribe la cadena en el campo de texto del buscador y presiona el botón **Buscar**.
CU asociados	_

Precondiciones	_
Poscondiciones	_
Referencias	RF10
Descripción	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor escribe una cadena de caracteres en el campo de texto del buscador y presiona el botón **Buscar**.	2. El sistema muestra una lista con los resultados de la búsqueda agrupando las métricas en base y derivadas, mostrando su nombre, tipo y clasificación.
Cursos Alternos	
Línea 2: • En caso de no existir ningún resultado el sistema muestra un mensaje indicando que no existen resultados asociados a la búsqueda.	

2.8. Conclusiones.

En este capítulo quedaron fundamentadas las características que deberá tener el sistema según las necesidades del proceso que se necesita informatizar. Se realizó además la propuesta de solución, obteniéndose a partir de un análisis las funcionalidades que debe tener el sistema, las cuales se representaron mediante un Diagrama de casos de uso y finalmente se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Con estos resultados, se puede comenzar a construir el sistema, dándole cumplimiento a todos los requerimientos establecidos en este capítulo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1. Introducción.

RUP define el análisis y el diseño, como un único flujo de trabajo en el que se realizan desde el inicio del desarrollo del software. El presente capítulo está dedicado a este flujo de trabajo, donde se traducen los requisitos funcionales a una especificación que describe como implementar la solución propuesta. Su objetivo es definir los diagramas de clases del análisis especificando qué clases toman parte del caso de uso y las relaciones entre ellas. Además se muestran los diagramas de clases del diseño que describen la realización física de los casos de uso y los diagramas de secuencia como una descripción gráfica de la interacción entre los actores y el sistema. Por último se estructura la información que se desea persista a través del diseño de la base de datos y se tratan los principios de diseño de la aplicación.

3.2. Modelo de análisis.

Durante el análisis, se analizan los requisitos que se describieron en la captura de requisitos, refinándoles y estructurándolos. El objetivo de hacerlo es conseguir una comprensión mas precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura. **[22]**

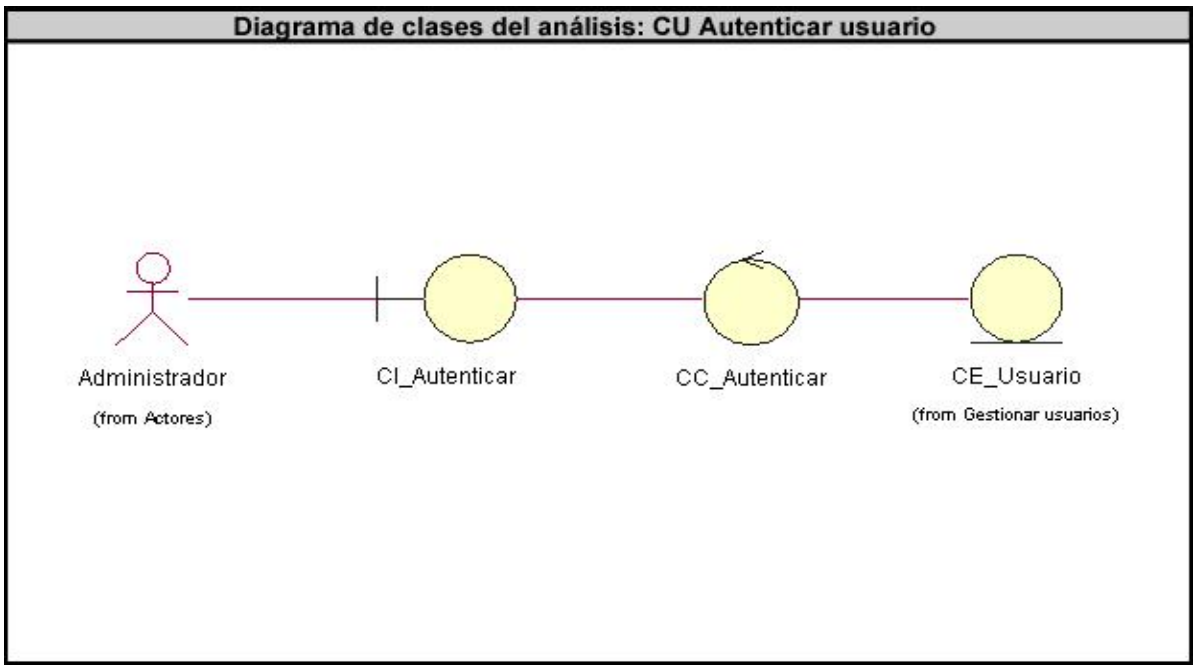
El modelo de análisis RUP lo define como la vista interna del sistema descrita en el lenguaje del desarrollador. Estructurado por clases y paquetes estereotipados, este proporciona la estructura interna del sistema y es utilizado fundamentalmente por los desarrolladores para comprender como debería darse forma al sistema, es decir, como debería ser diseñado e implementado. Además define las realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

3.2.1. Diagrama de clases del análisis.

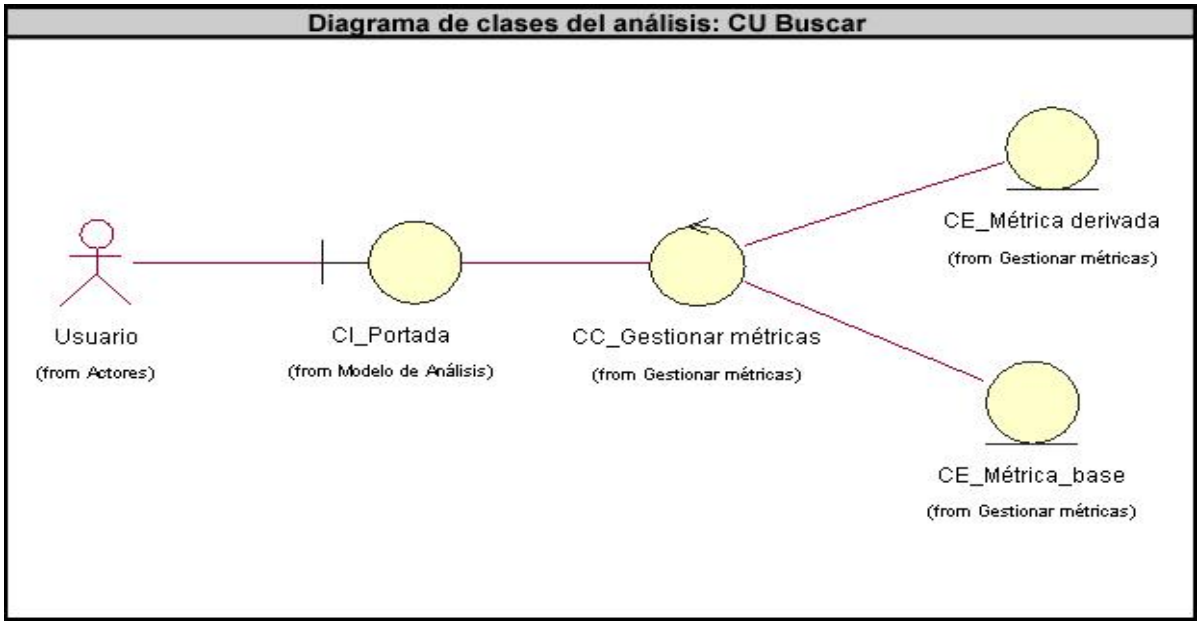
Dentro del modelo de análisis, los casos de uso se describen mediante clases del análisis y sus objetos. Esto se representa mediante colaboraciones dentro del modelo de análisis que llamamos realizaciones casos de uso-análisis. **[22]**

Para manejar todo esto se utilizan diagramas de clases conectados a una realización casos de uso, mostrando sus clases participantes y sus relaciones.

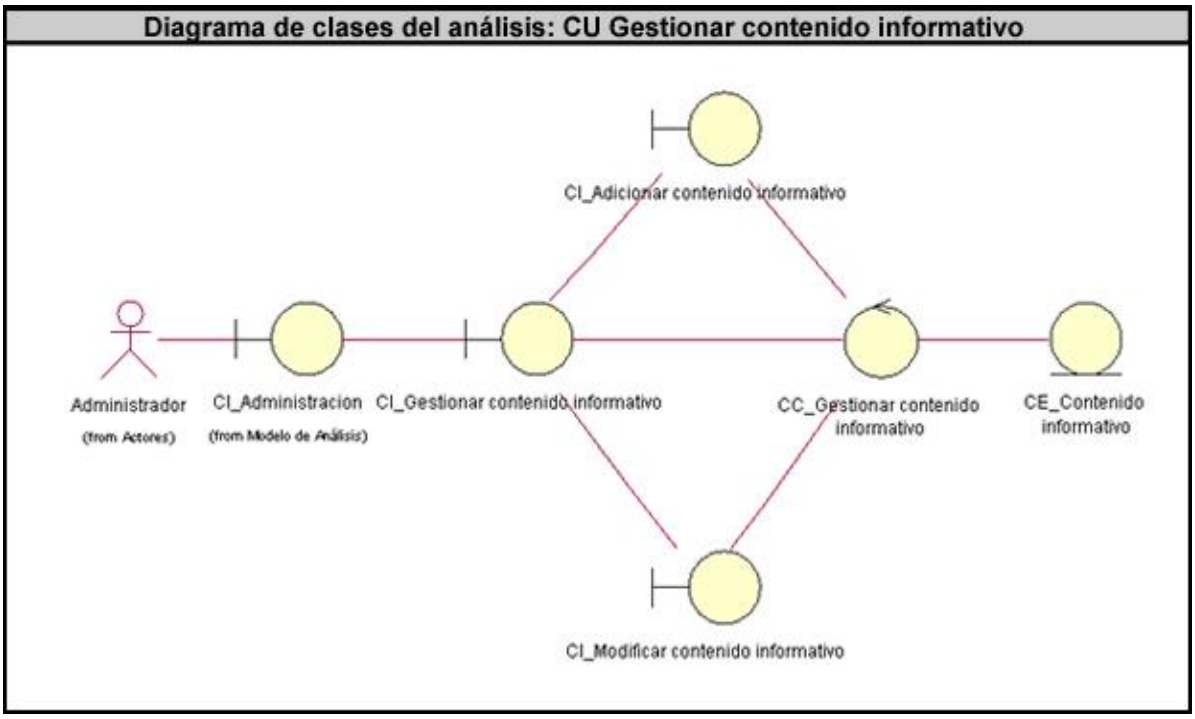
3.2.1.1. Diagrama de clases del CU "Autenticar usuario".



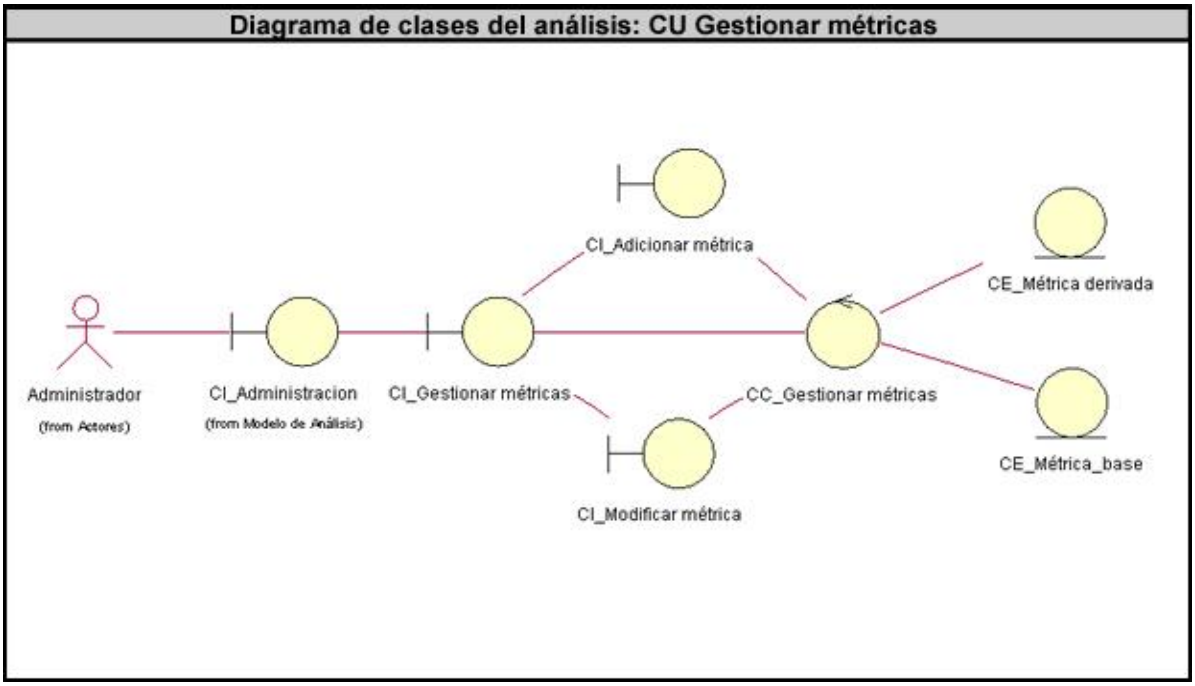
3.2.1.2. Diagrama de clases del CU "Buscar".



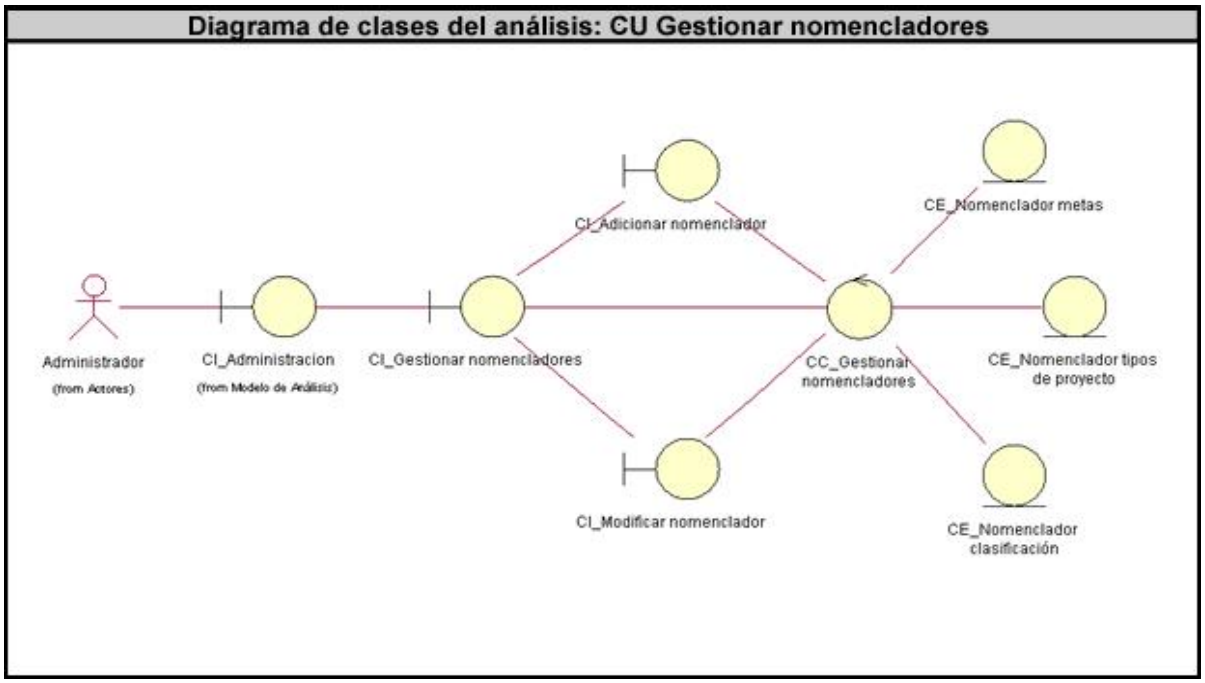
3.2.1.3. Diagrama de clases del CU “Gestionar contenido informativo”.



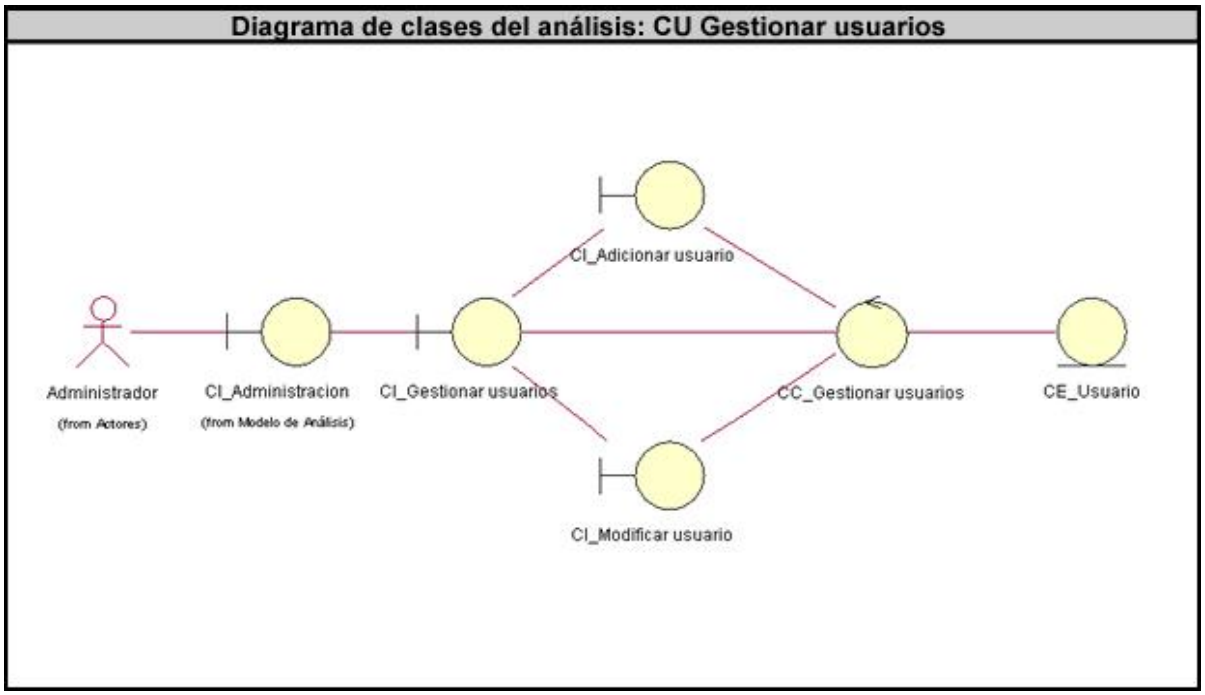
3.2.1.4. Diagrama de clases del CU “Gestionar métrica”.



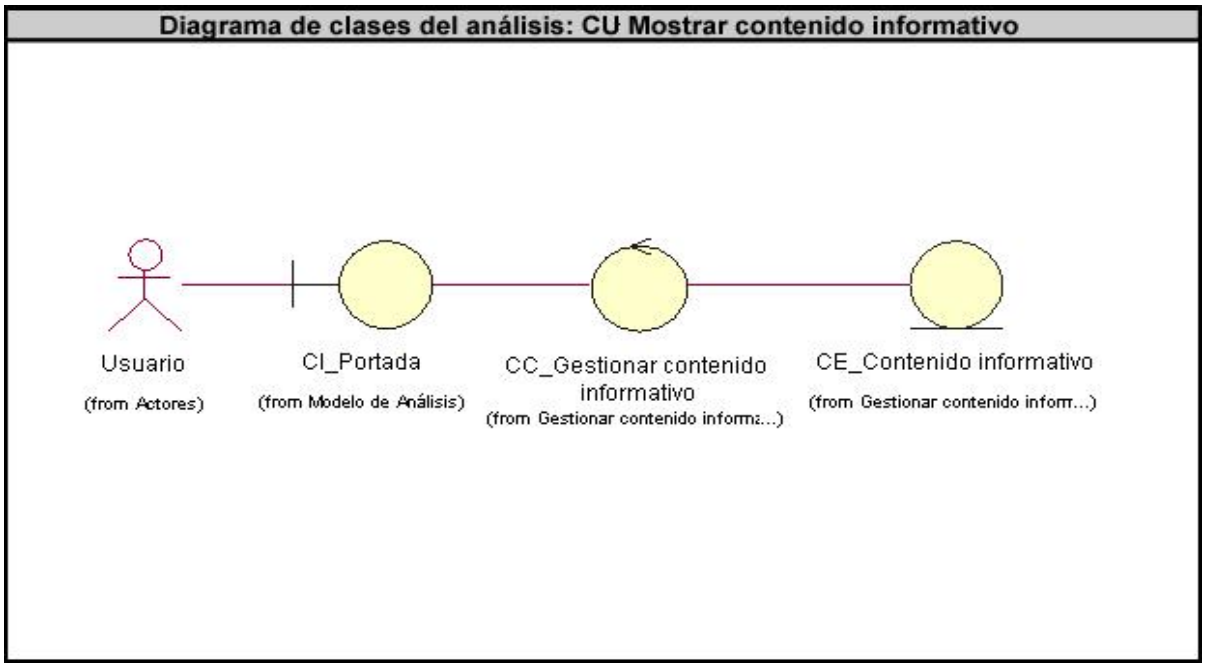
3.2.1.5. Diagrama de clases del CU “Gestionar nomencladores”.



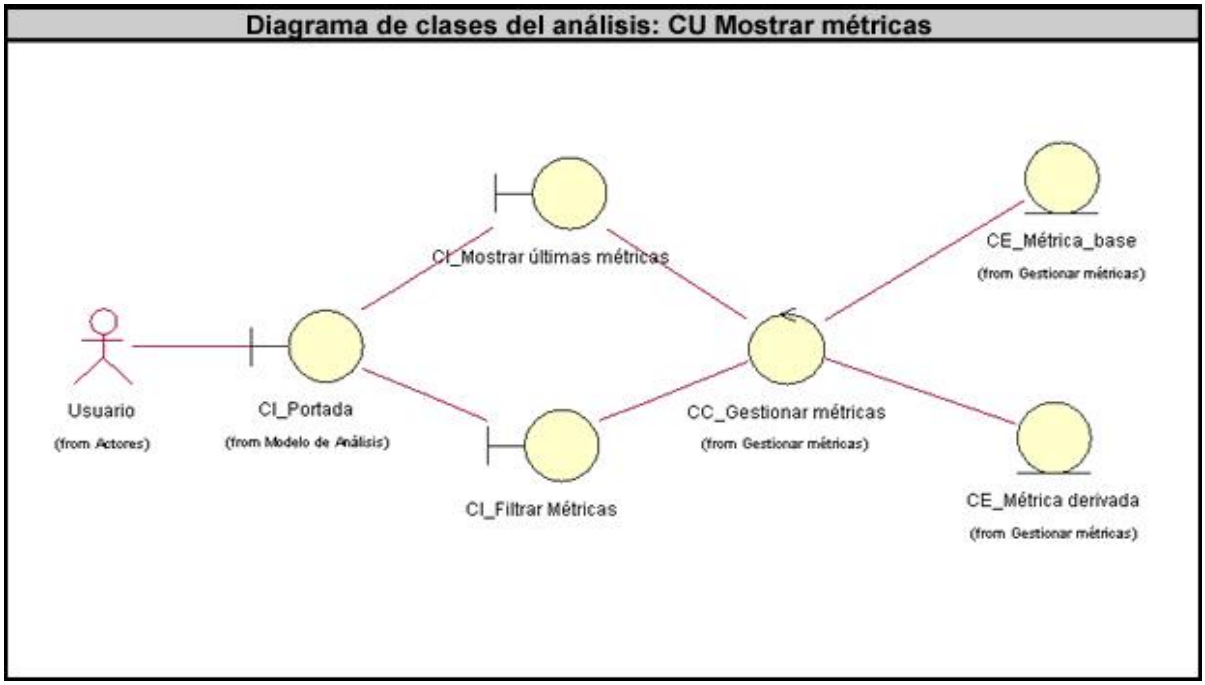
3.2.1.6. Diagrama de clases del CU “Gestionar usuarios”.



3.2.1.7. Diagrama de clases del CU "Mostrar contenido informativo".



3.2.1.8. Diagrama de clases del CU "Mostrar métricas".



3.3. Modelo de diseño.

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones que se imponen. Entre los propósitos que RUP propone que se sigan con el diseño se encuentran:

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación.
- Crear una entrada apropiada y un punto de partida para actividades de implementación capturando los requisitos, interfaces y clases. **[22]**

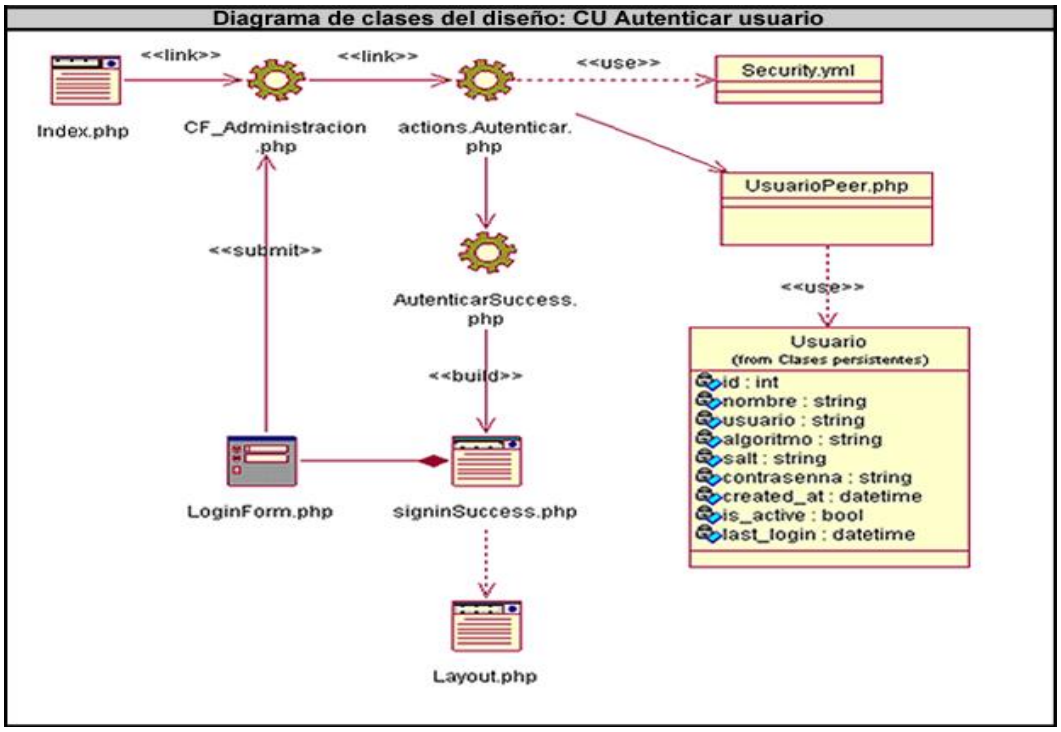
El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen un impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. **[22]**

3.3.1. Diagrama de clases del diseño.

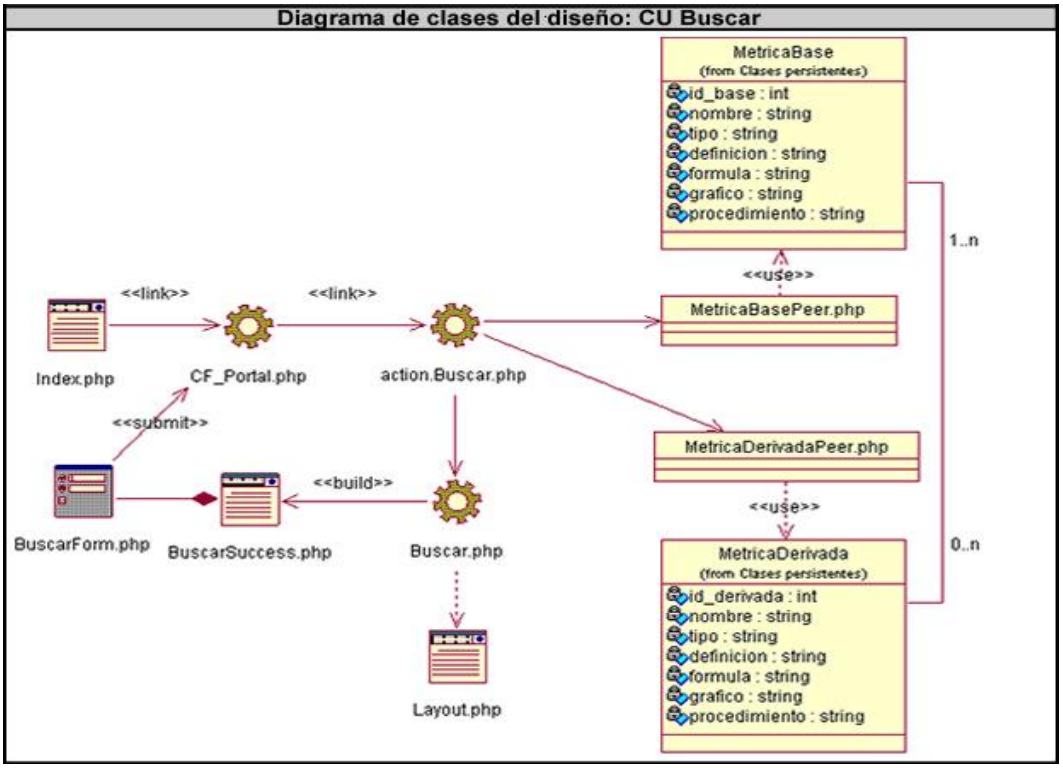
En el modelo de diseño, los casos de uso son realizados por las clases y los objetos del diseño. Esto se representa por colaboraciones en el modelo de diseño y denota una realización de casos uso-diseño, esto ligado a algunas operaciones, atributos y asociaciones sobre una clase específica es manipulado por los diagramas de clases que conectados a una realización casos de uso, muestran sus clases participantes y sus relaciones.

Para realizar los diagramas de clases del diseño se usó la misma lógica con la que funciona el patrón MVC. De este modo las *páginas cliente* y los *formularios* constituyen la vista, el *controlador frontal* y las clases *actions* son el controlador, y las clases de la *lógica de negocio* y el *acceso a datos* representan el modelo. (Ver anexo 2)

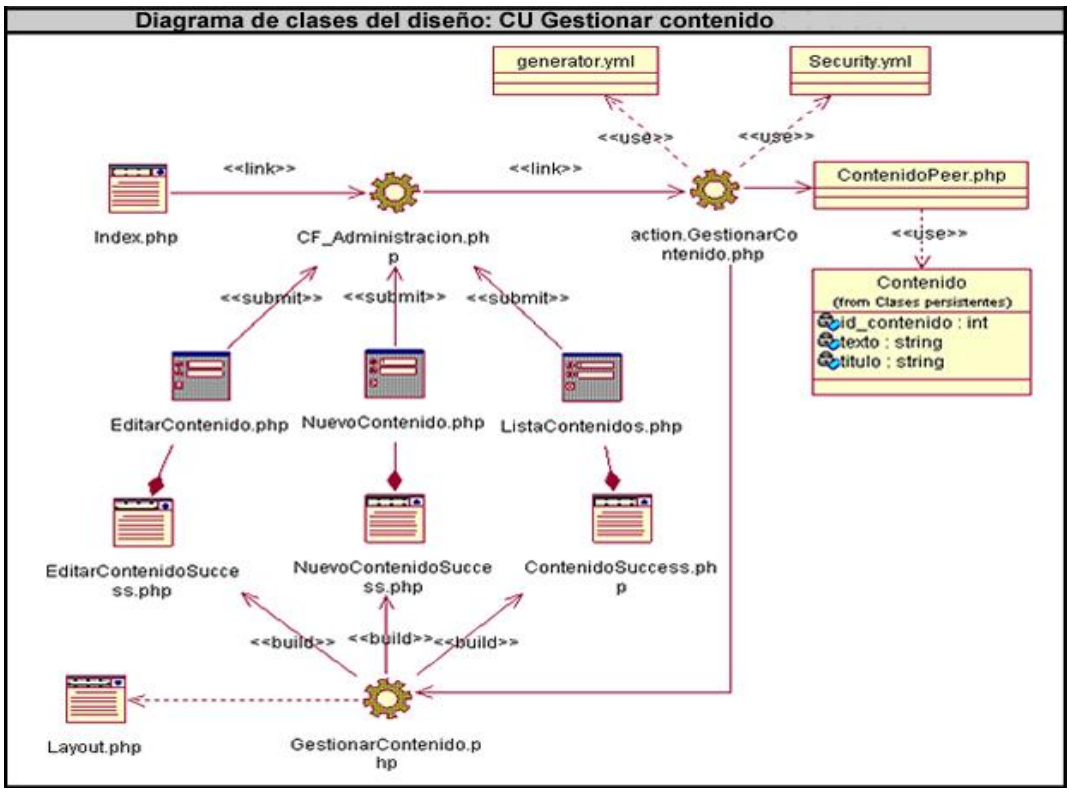
3.3.1.1. Diagrama de clases del CU "Autenticar usuario".



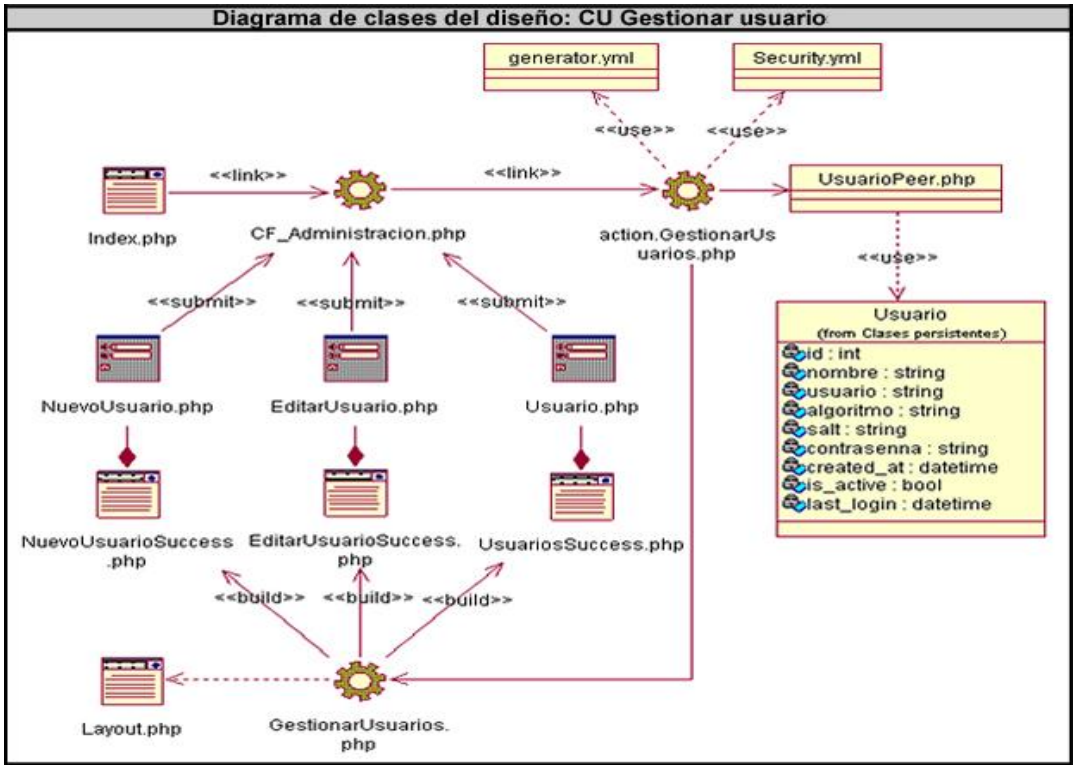
3.3.1.2. Diagrama de clases del CU "Buscar".



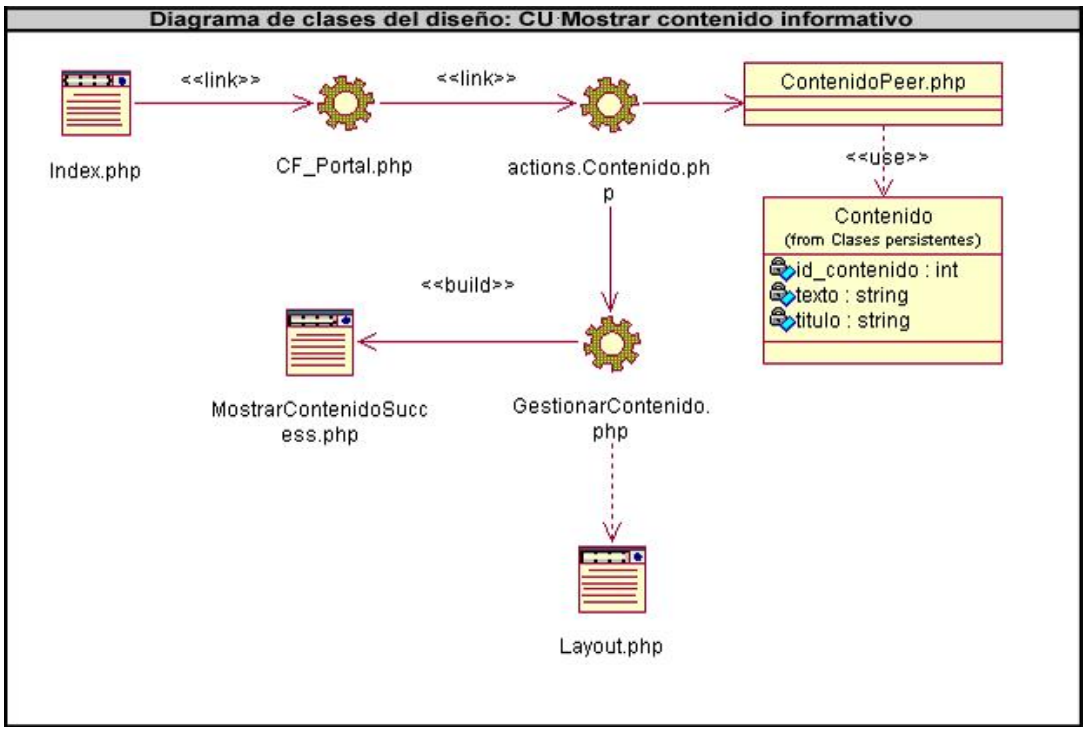
3.3.1.3. Diagrama de clases del CU "Gestionar contenido".



3.3.1.4. Diagrama de clases del CU "Gestionar usuario".

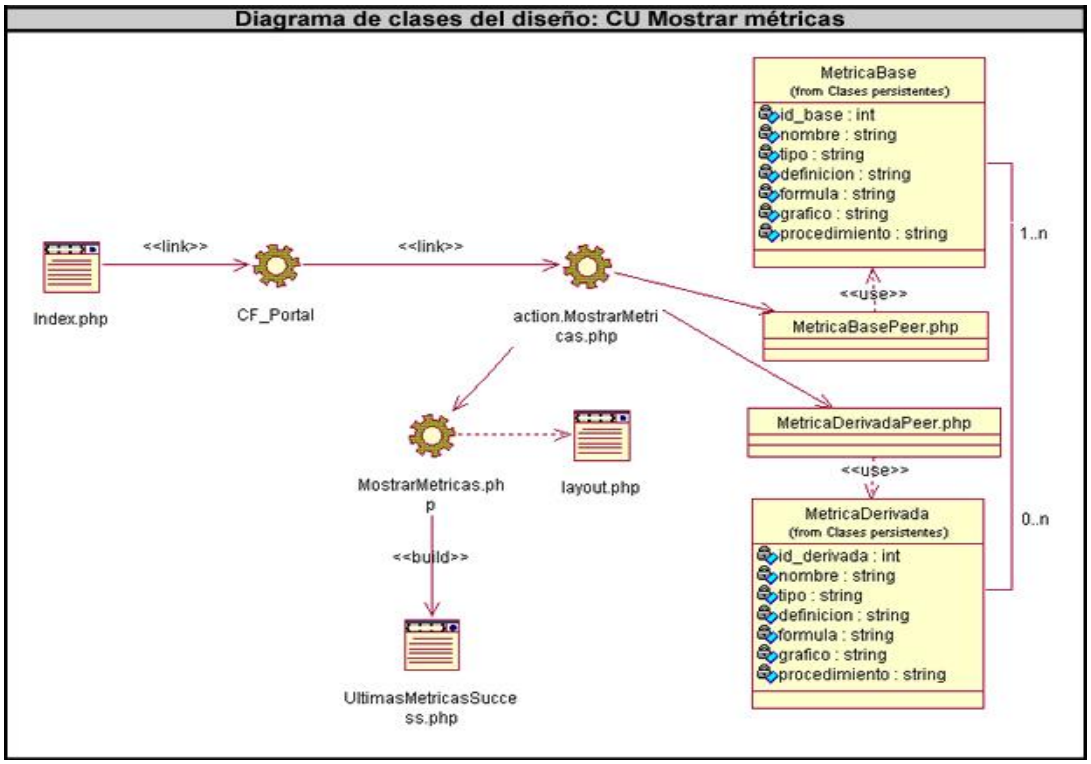


3.3.1.5. Diagrama de clases del CU "Mostrar contenido informativo".

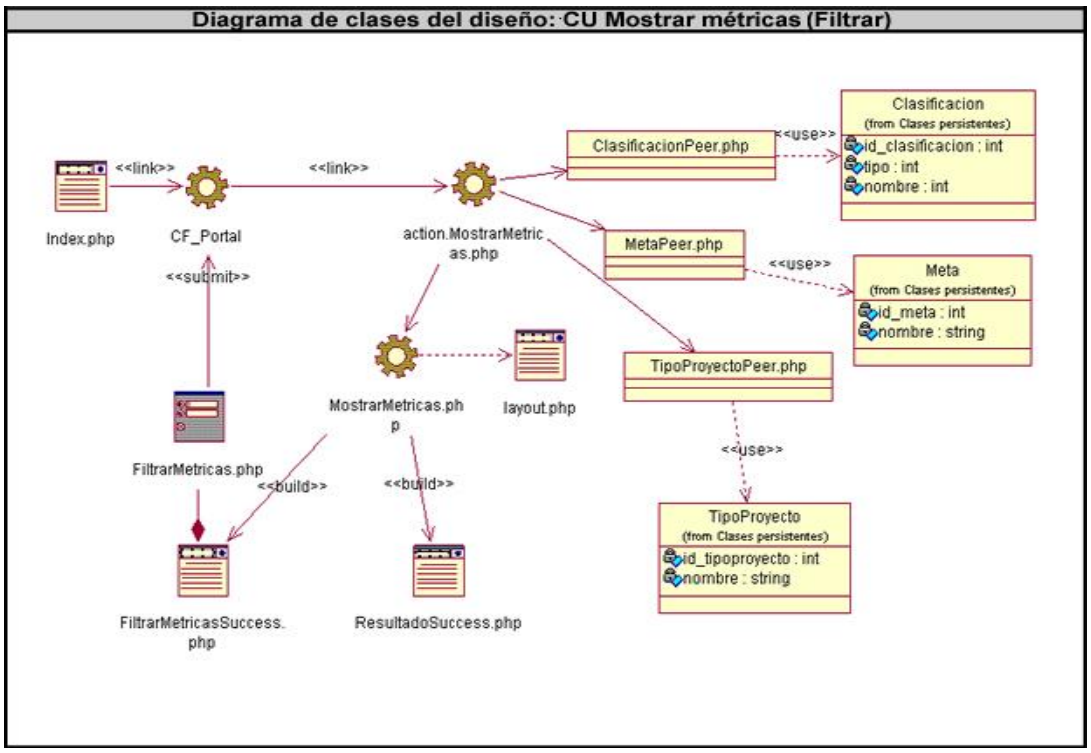


3.3.1.6. Diagramas de clases del CU "Mostrar métricas".

3.3.1.6.1. Mostrar métricas.

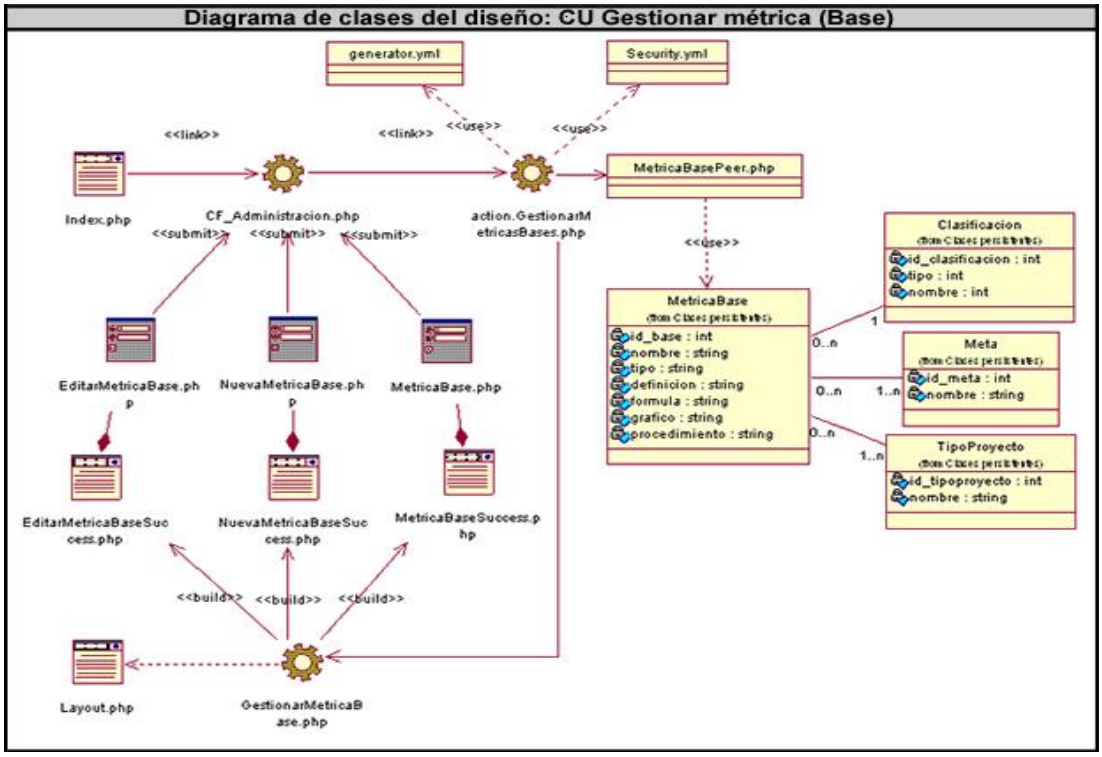


3.3.1.6.2. Filtrar métricas.

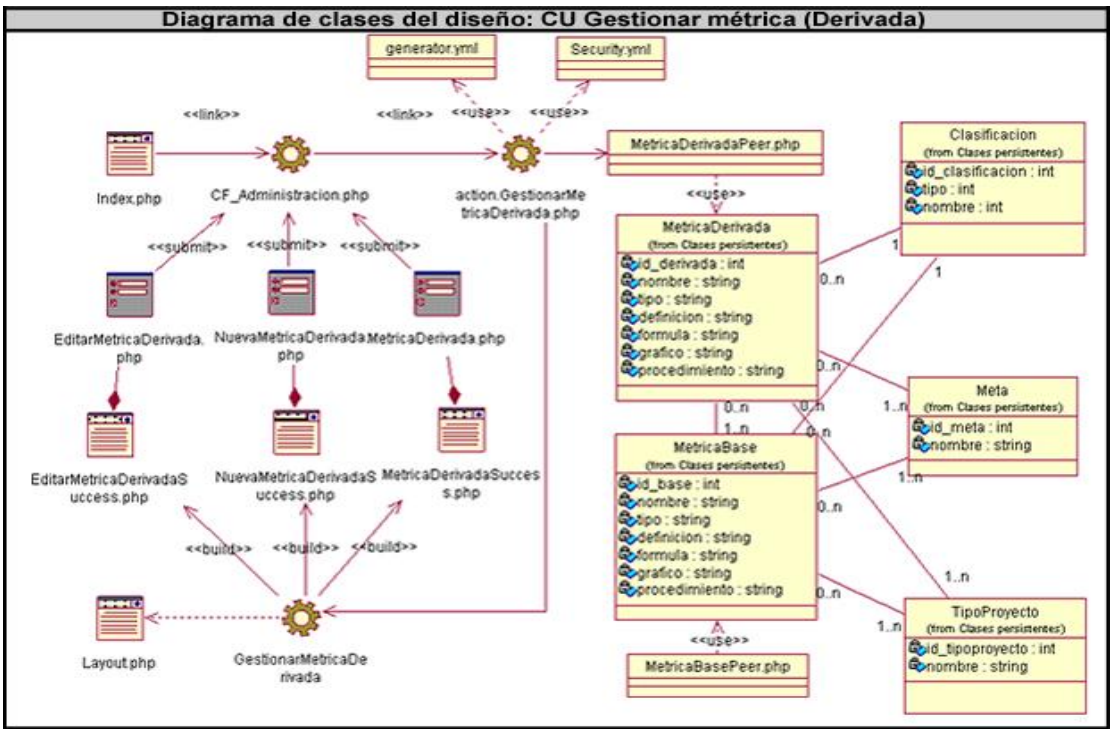


3.3.1.7. Diagrama de clases del CU “Gestionar métrica”.

3.3.1.7.1. Gestionar métrica base.

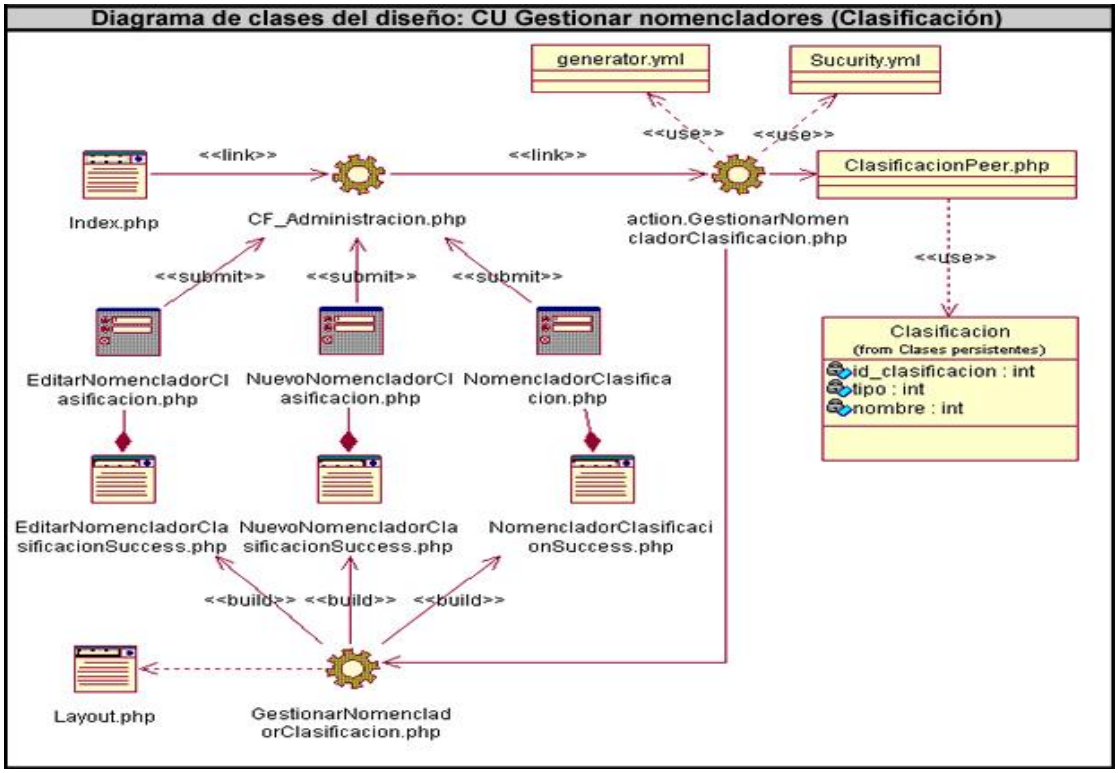


3.3.1.7.2. Gestionar métrica derivada.

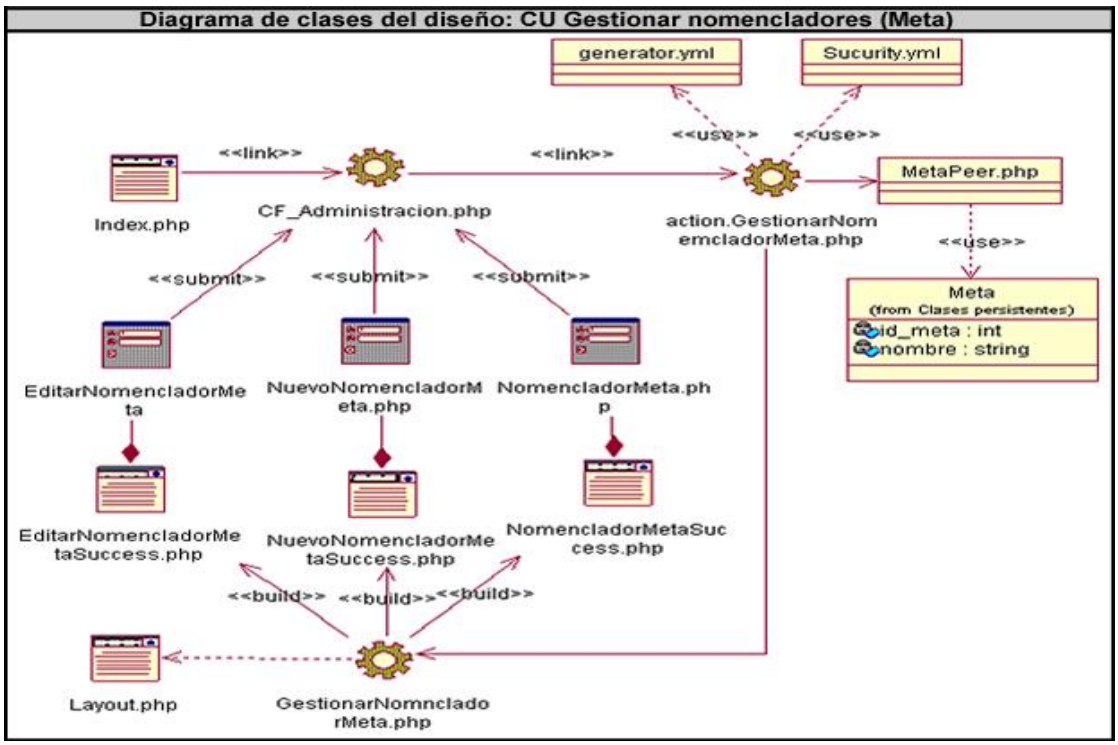


3.3.1.8. Diagrama de clases del CU “Gestionar nomencladores”.

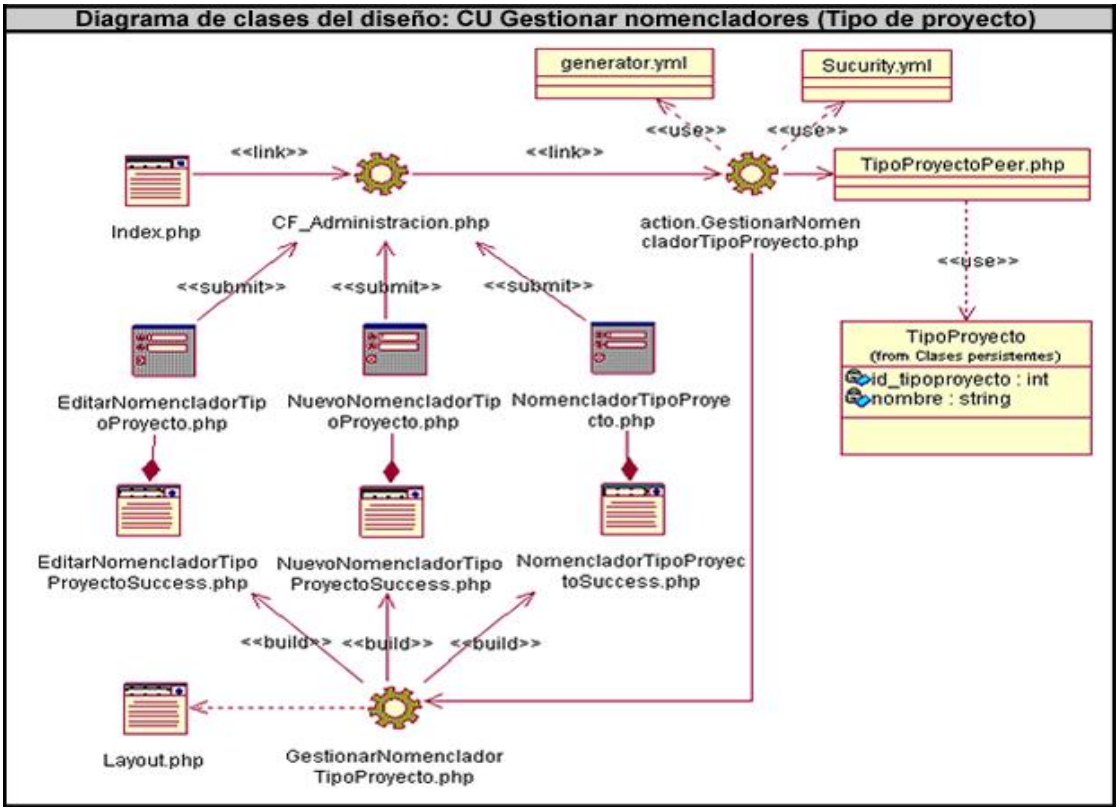
3.3.1.8.1. Nomenclador clasificación.



3.3.1.8.2. Nomenclador meta.



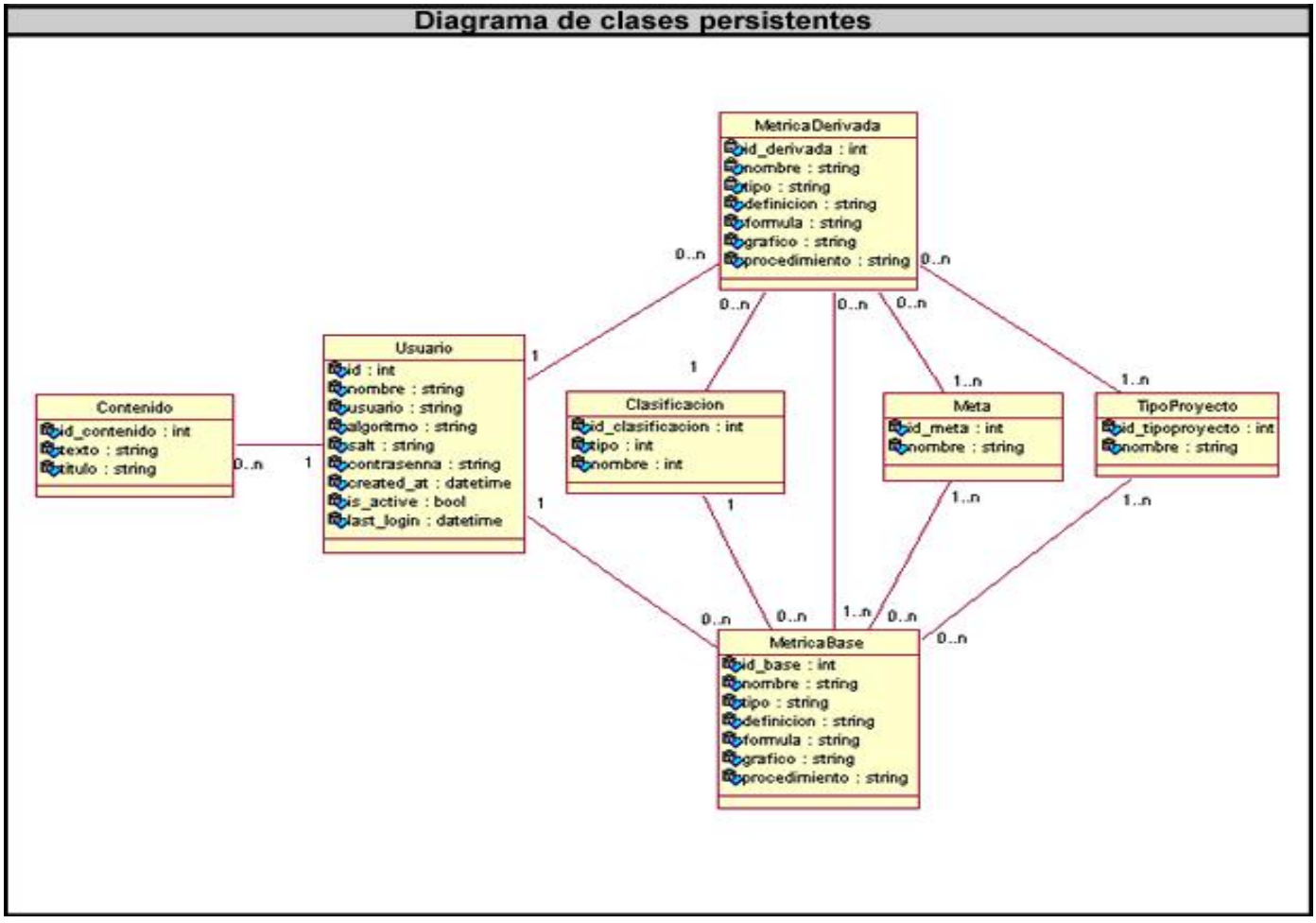
3.3.1.8.3. Nomenclador tipo de proyecto.



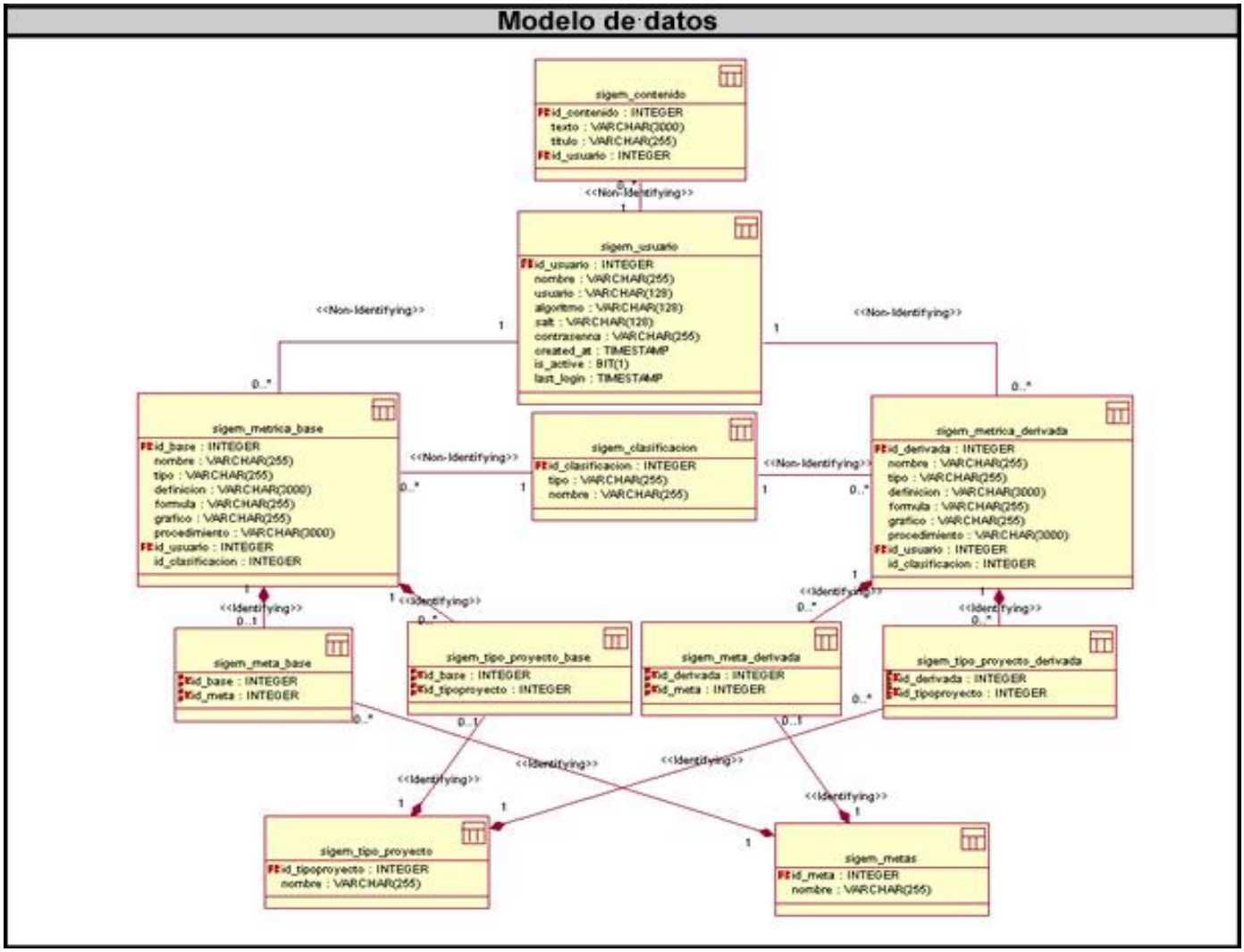
3.4. Diseño de la base de datos.

Todas las clases identificadas en el dominio del análisis y le diseño no son persistentes. La persistencia es la capacidad que tiene un objeto de mantener su valor en el espacio y en el tiempo. Algunas clases representan los datos que se obtienen y almacenan durante los procesos de la aplicación, estas se pueden modelar a través de un Diagrama de clases persistentes, lo que permitirá ver la relación entre los datos, y completará el moldeamiento de la lógica de la aplicación. Para el correcto diseño de la base de datos RUP propone la elaboración del Diagrama de clases persistentes para mediante este, generar el Modelo de datos que no es más que la descripción lógica y física de los datos en el sistema.

3.4.1. Diagrama de clases persistentes.



3.4.2. Modelo de datos.



3.5. Diagramas de interacción.

En los diagramas de interacción son representadas las interacciones entre las diferentes clases que interactúan en la realización de un caso de uso. Hay dos tipos de diagrama de interacción: secuencia y colaboración, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular.

Los diagramas de colaboración muestran una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre ellos. En tanto los diagramas de secuencia, que fueron los utilizados para modelar la interacción de las clases del sistema, muestran una interacción ordenada según la secuencia temporal de eventos. (Ver anexo 3)

3.6. Principios del diseño universal.

Los siete Principios del Diseño Universal, se centran en el diseño utilizable universalmente. Estos principios generales del diseño, son aplicables y de hecho se aplican en la arquitectura, la ingeniería y, por supuesto, las páginas y aplicaciones Web, entre otros campos de aplicación. El objetivo del diseño debe basarse en el usuario, razón por la cual este sistema utiliza e implementa según las necesidades identificadas seis de estos principios.

Uso equiparable: Donde las características de privacidad, garantía y seguridad estén igualmente disponibles para todos los usuarios, y se proporcionen las mismas maneras de uso para todos los usuarios: idénticas cuando es posible, equivalentes cuando no lo es. Que el diseño sea atractivo para todos los usuarios.

Uso flexible: Donde se ofrezcan posibilidades de elección en los métodos de uso, que facilite al usuario la exactitud y precisión, y se adapte al paso o ritmo del usuario.

Simplicidad e intuición: Donde se elimine la complejidad innecesaria y que sea consistente con las expectativas e intuición del usuario.

Información perceptible: Donde se usen diferentes modos para presentar de manera redundante la información esencial (gráfica y verbal), se proporcione contraste suficiente entre la información esencial y sus alrededores, se amplíe la legibilidad de la información esencial, y que diferencie los elementos en formas que puedan ser descritas.

Tolerancia al error: Donde se dispongan los elementos para minimizar los riesgos y errores, por ejemplo utilizando elementos comunes; y los elementos peligrosos eliminados, aislados o tapados, que se proporcionen advertencias sobre peligros y errores. Posibilitar el descubrimiento interactivo y el aprendizaje ensayo-error, y posibilitar la reversibilidad de las acciones.

Esfuerzo de acceso y uso: Que minimicen las acciones repetitivas, y que proporcione una línea de visión clara hacia los elementos importantes para todos los usuarios.

3.7. Tratamiento de errores.

En el sistema propuesto el tratamiento de errores se realiza con el fin de evitar, minimizar y tratar los posibles errores, garantizando de esta forma la integridad y confiabilidad de la información que se registra

y se muestra al usuario. Antes de realizar cualquier acción se validan los datos y antes de eliminar, modificar o adicionar se exige confirmación al usuario. El usuario es informado cuando ha ejecutado o no correctamente las acciones, por ejemplo el dejar campos requeridos en un formulario o al introducir datos inválidos. De forma general siempre se notifica al usuario de los posibles errores tanto en la navegación como en la administración del sistema.

3.8. Interfaz usuario.

La calidad de la interfaz de usuario puede ser uno de los factores que conduzca a una aplicación al éxito o al fracaso, razón por la cual su consistencia es considerada como uno de los aspectos más relevantes.

Las páginas de la interfaz del sistema están diseñadas con un estilo común garantizando que exista uniformidad y definiendo de forma única la manera de presentar los contenidos. Todo ello fue posible gracias al uso de hojas de estilo, lo que reduce el uso de imágenes que puedan demorar el tiempo de respuesta de las aplicaciones.

De igual modo la aplicación de administración brinda al usuario una interfaz amigable con menús desplegables y acceso a determinadas acciones por más de una vía, lo que sin duda mejora la usabilidad del sistema.

3.9. Conclusiones.

Al concluir este capítulo queda realizado satisfactoriamente el flujo de trabajo Análisis y Diseño. Como los resultados más importantes se encuentra la realización de los diagramas de clases del análisis, la realización de los diagramas de clases del diseño y el modelo de datos del sistema. Se abordaron también los principios de diseño seguidos en el desarrollo de la interfaz de usuario, además de explicar como el sistema maneja el tratamiento de errores.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1. Introducción.

RUP propone que el mayor desarrollo en el flujo de trabajo de Implementación se realice una vez concluido el análisis y el diseño del sistema. En la implementación se comienza precisamente con los resultados del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. La mayor parte de la arquitectura del sistema es capturada durante el diseño, siendo el propósito fundamental de la implementación el desarrollar la arquitectura y el sistema como un todo.

RUP plantea además que las actividades de prueba se realicen durante todo el desarrollo del software, pero que estas se incrementen durante y una vez concluida la implementación del sistema, para de esta forma verificar los resultados probando cada construcción, así como las versiones del sistema a ser entregadas al usuario final.

El propósito de este capítulo es describir como los elementos presentes en el diseño se implementan en términos de componentes y como el sistema se organiza de acuerdo a los nodos específicos en el modelo de despliegue, utilizando para ello los diagramas de componentes y el diagrama de despliegue. Además se hace una caracterización del ambiente de implantación del producto y se valida mediante pruebas de aceptación, el sistema que será entregado al usuario.

4.2. Modelo de despliegue.

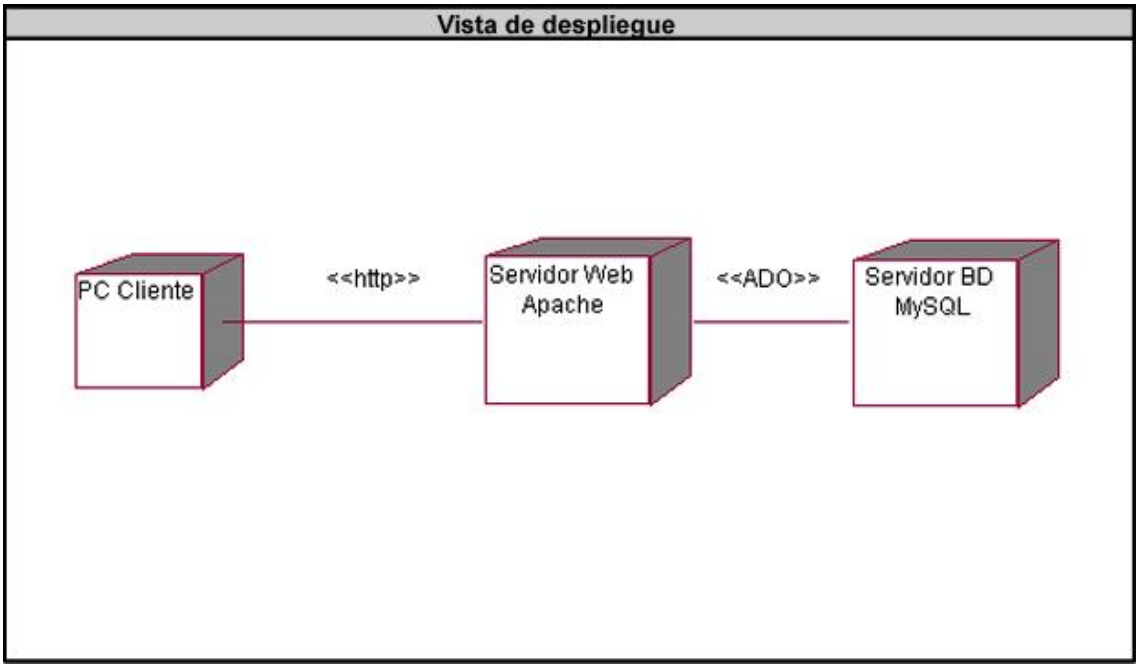
El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de como se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

Se puede observar lo siguiente sobre el modelo de despliegue:

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como internet, intranet, bus, y similares.

- El modelo de despliegue puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulaciones. **[22]**

4.2.1. Vista de despliegue.



4.3. Ambiente de implantación del producto.

Para el correcto despliegue del sistema en su entorno real de implantación, se hace necesario las siguientes demandas:

- Para la instalación del sistema se necesita una computadora servidor con Sistema Operativo Microsoft Windows Server 2003 o superior, o Sistema Operativo Linux. Además en el servidor se necesitan correctamente configuradas las siguientes tecnologías, Servidor Web Apache y Gestor de Bases de Datos PostgreSQL. Para el óptimo rendimiento del servidor, el mismo debe poseer como mínimo una memoria RAM de 512 MB y un procesador de 3.00 GHZ.
- Las computadoras clientes que serán las que harán consumo del sistema, pueden contar con Sistema Operativo Windows o Linux siempre que el mismo tenga instalado y correctamente configurado el navegador web Mozilla Firefox en sus versiones 2.0 o superior. Para el óptimo rendimiento de las mismas se recomienda como mínimo 256 MB de memoria RAM y 1.3 GHz de velocidad de procesamiento.

- En el caso de las personas que se van a encargar de la administración del sistema, deben poseer conocimientos informáticos, así como conocer los temas referentes a las métricas del software y sus elementos compositivos. Por otro lado las personas que consumirán información del sistema no deben tener necesariamente amplios conocimientos informáticos, sino un conocimiento básico en el manejo de la computadora y de un ambiente Web en sentido general. Estas últimas para poder comprender la información que se brinda en el sistema, tienen que dominar los conceptos de métrica de software y su importancia en el desarrollo de software. En caso de que estos usuarios finales no dominen del todo la terminología que aborda el sistema, es responsabilidad de la DCS hacerles llegar los elementos necesarios para su comprensión.

4.4. Modelo de implementación.

El modelo de implementación describe como los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes uno de los otros. **[22]**

4.4.1. Componentes de implementación.

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Algunos estereotipos estándar de componentes son los siguientes:

- “executable”: Es un programa que puede ser ejecutado en un nodo.
- “file”: Es un fichero que contiene código fuente o datos.
- “library”: Es una librería estática o dinámica.
- “table”: Es una tabla de base de datos.
- “document”: Es un documento. **[22]**

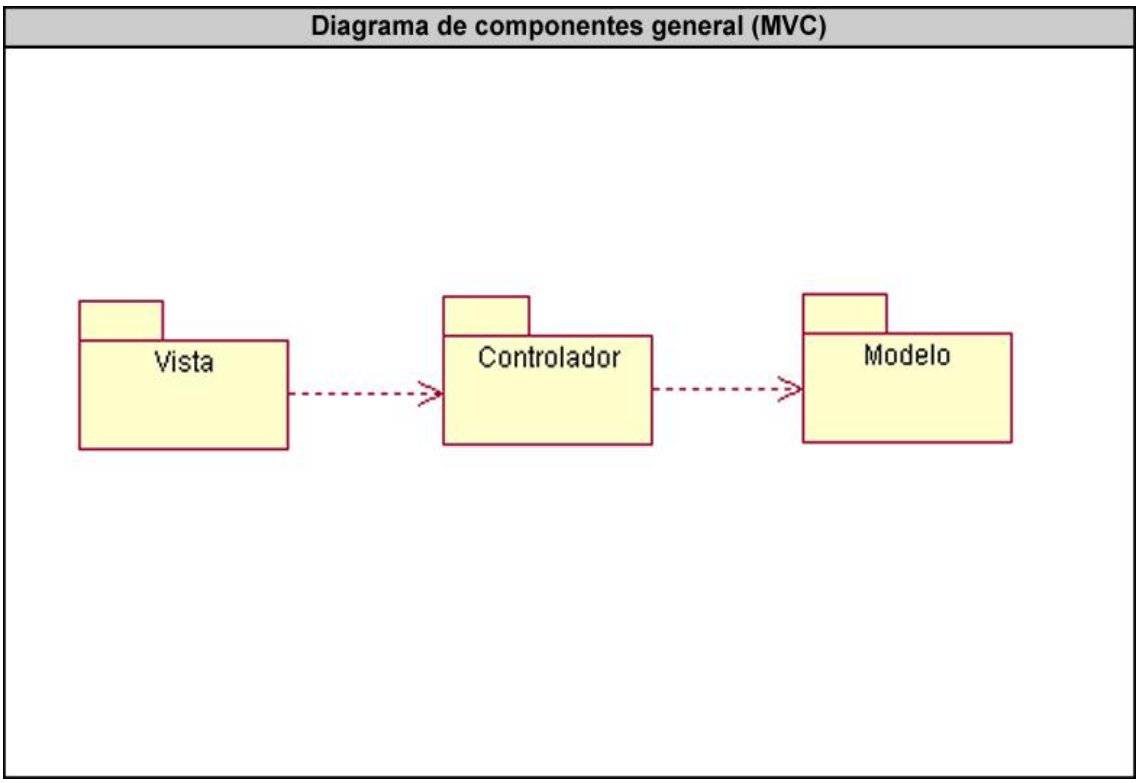
4.4.2. Diagrama de componentes.

Según RUP los diagramas de componentes estructuran el modelo de implementación en términos de subsistemas de implementación y muestran las relaciones entre los elementos de implementación. Estos modelan la vista estática del sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes, sean estos de código fuente, librerías, ejecutables, etc.

Para modelar la implementación del sistema se tuvo en cuenta que el mismo contaba con dos aplicaciones internas y que el patrón arquitectónico utilizado era el Modelo Vista Controlador (MVC). Esto llevó a organizar los componentes en paquetes, de acuerdo con su funcionalidad y sus características, además de tener en cuenta los módulos con que cuenta cada una de las aplicaciones.

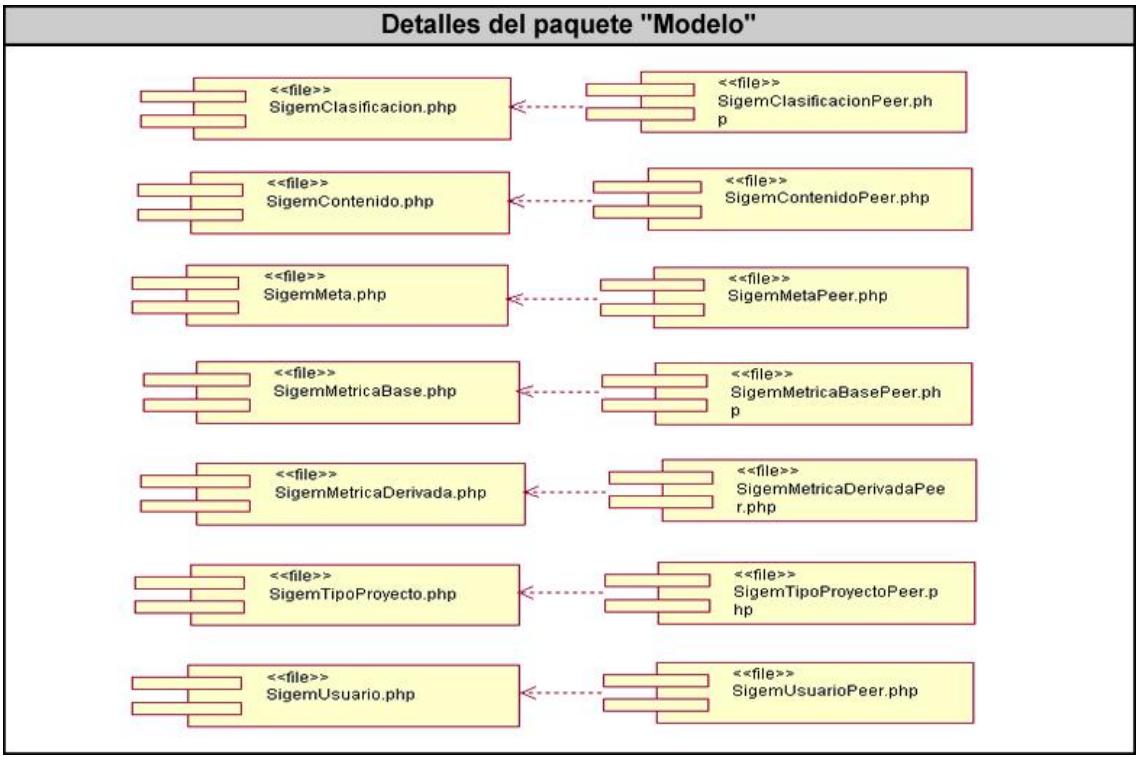
4.4.2.1. Vista general de componentes.

La vista general de componentes es la misma para las dos aplicaciones, pues ambas usan el mismo patrón MVC.



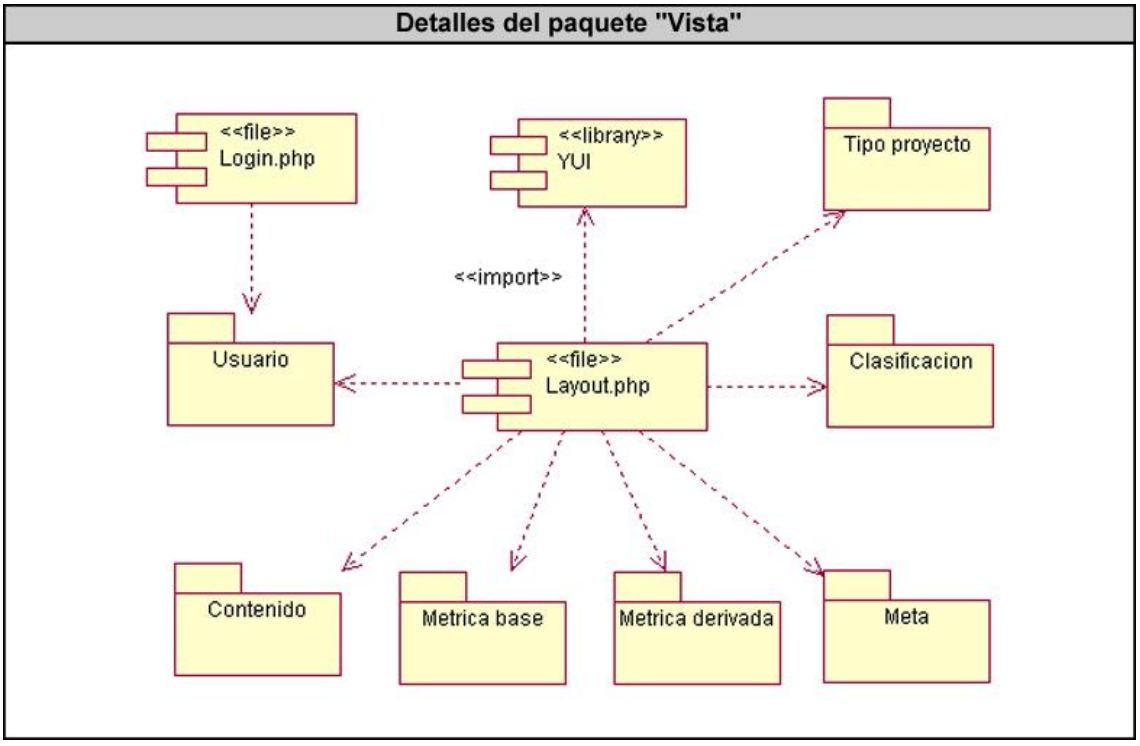
4.4.2.2. Detalles del paquete “Modelo”.

Dado que ambas aplicaciones usan el mismo modelo y acceden a los mismos datos ya sea de una u otra forma, el paquete “Modelo” es el mismo para las dos.

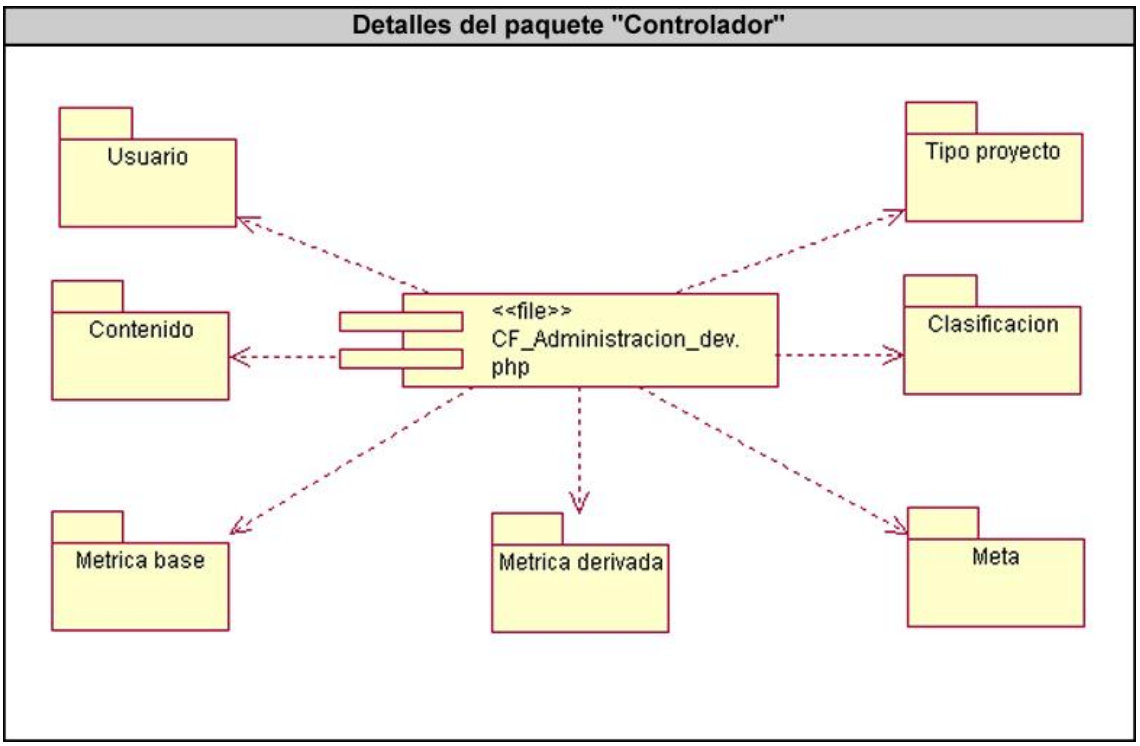


4.4.2.3. Diagramas de componentes de la aplicación administrativa.

4.4.2.3.1. Detalles del paquete “Vista”.

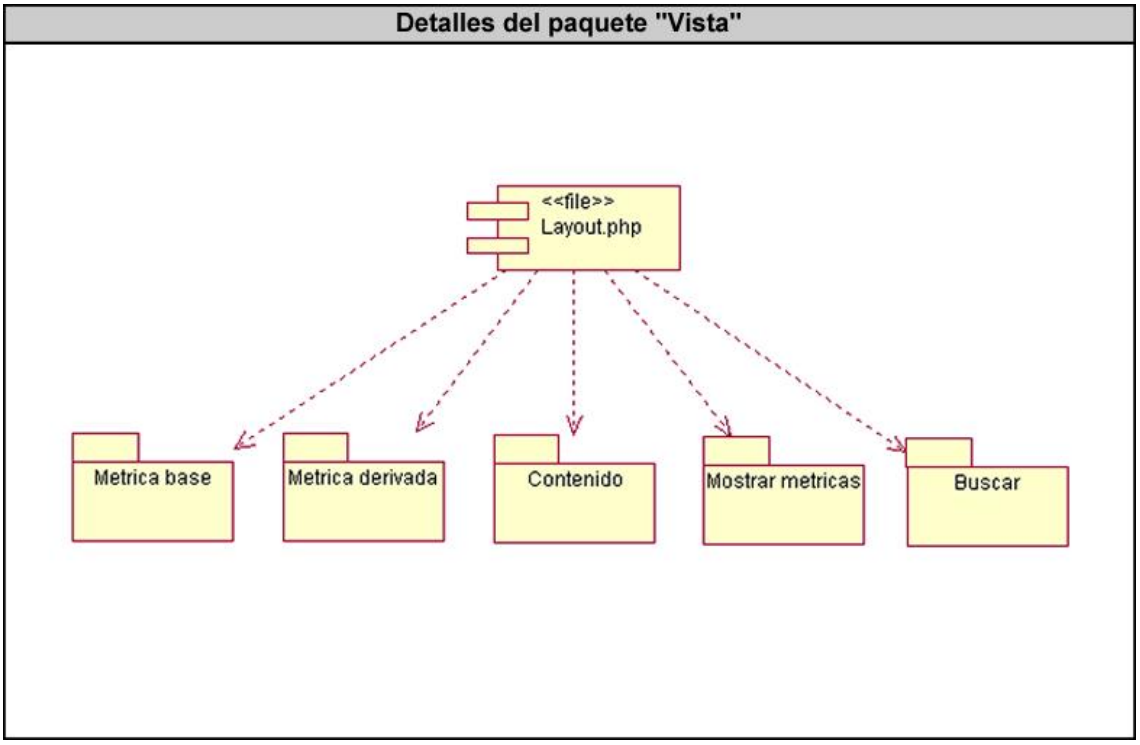


4.4.2.3.2. Detalles del paquete “Controlador”.

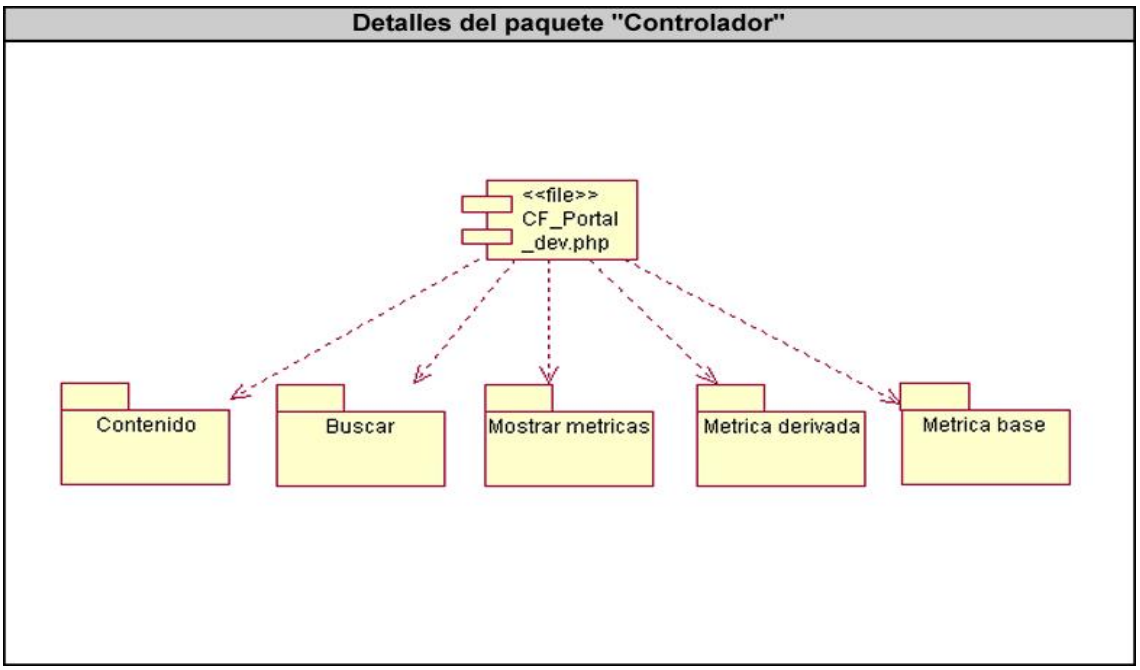


4.4.2.4. Diagramas de componentes de la aplicación informativa.

4.4.2.4.1. Detalles del paquete “Vista”.

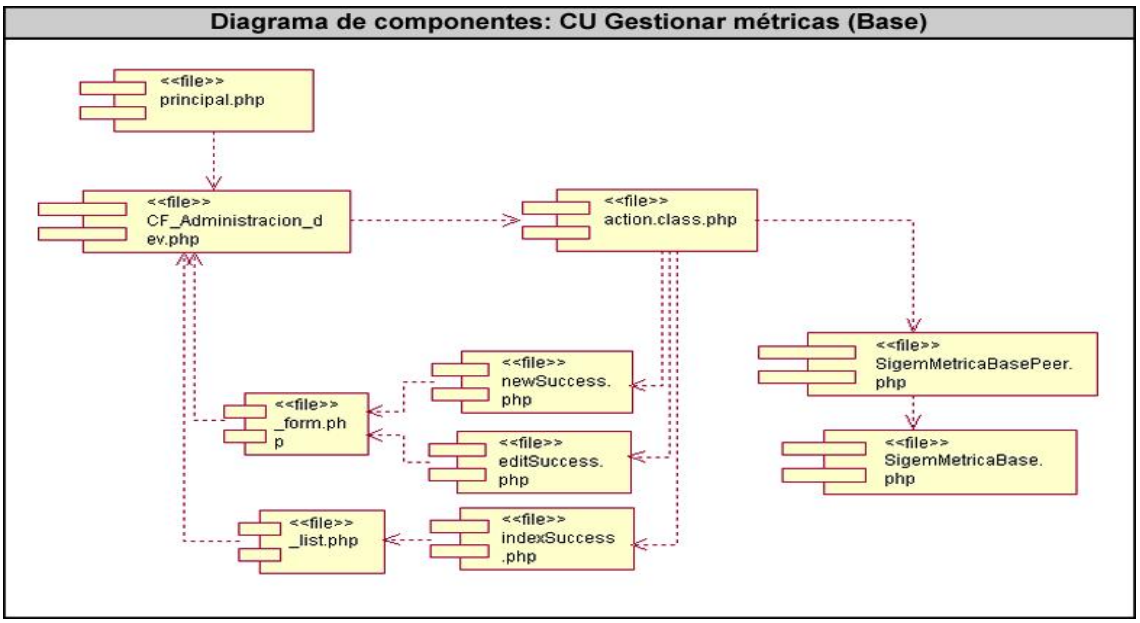


4.4.2.4.2. Detalles del paquete “Controlador”.



4.4.2.5. Realización de los CU.

La realización de los CU mediante los diagramas de componentes permitió ver de una forma más clara como realmente se comportará el sistema en cuanto a su funcionamiento. A continuación se muestra uno de los diagramas de componentes del CU Gestionar métricas.



Para consultar el resto de las realizaciones del los CU ver Anexo 4.

4.5. Pruebas.

El desarrollo de sistemas de software implica una serie de actividades de producción en la que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma erronea o imperfecta, asi como en posteriores pasos de diseño y desarrollo. Debido a la imposiblidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad. Las pruebas del software son un elemento crítico para la garantia de la caldiad del software y representan una revision final de las especificaciones, del diseño y de la codificación. **[5]** Estas no son más que la ejecución de un sistema bajo ciertas condiciones con la intención de descubrir errores.

4.5.1. Pruebas de Caja Negra.

El método de prueba a utilizar para comprobar la funcionalidad del sistema son las pruebas de caja negra. Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. **[5]**

Las pruebas de caja negra intentan encontrar errores en las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.

4.5.2. Diseño de casos de prueba.

Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultados con la que se ha de probar y las condiciones bajo las que ha de probarse. **[22]**

El objetivo del diseño de casos de prueba es comprobar el sistema utilizando el método de pruebas de caja negra, el cual cuentan con varias técnicas de pruebas, pero entre las más conocidas por los probadores se encuentran:

- **Partición equivalente:** La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de

prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que de otro modo requerían la ejecución de muchos casos antes de detectar el error genérico.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados validos o no validos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

- **Análisis de valores límite:** Por razones que no están del todo claras, los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. **[5]**

4.5.3. Pruebas de aceptación.

Las pruebas de aceptación no son más que pruebas de caja negra. Estas las realiza el usuario final en lugar del responsable del desarrollo del sistema, una prueba de aceptación puede ir desde un informal "paso de prueba" hasta la ejecución sistemática de una series de prueba bien planificadas. **[5]**

Para la comprobación y evaluación final del sistema con el usuario, se realizó una prueba piloto como también son llamadas las pruebas de aceptación. En esta prueba mediante la técnica de diseños de casos de prueba se probaron todos los requisitos funcionales del sistema.

4.5.3.1. Pruebas de aceptación para el requisito Autenticar usuario.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Autenticar usuario
Propósito	Probar que el usuario se autentica correctamente en el sistema.
Condiciones de ejecución	El usuario debe encontrarse en la página de autenticación.
Entrada	Se intenta autenticar un usuario con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el usuario se autentica, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.2. Pruebas de aceptación para el requisito Buscar.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Buscar
Propósito	Probar que el usuario puede realizar una búsqueda en el sistema.
Condiciones de ejecución	El usuario debe encontrarse en la interfaz principal del sistema.
Entrada	Se intenta buscar información en el sistema con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema muestra el resultado de la búsqueda, si las entradas son inválidas el sistema emite un mensaje informando que no existen resultados de la búsqueda.
Resultado de la prueba	

4.5.3.3. Pruebas de aceptación para el requisito Adicionar usuario.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar usuario
Propósito	Probar que se puede adicionar un nuevo usuario al sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Usuarios/Gestionar usuarios** o menú **Usuarios/Nuevo**.
Entrada	Se intenta adicionar un nuevo usuario con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona el usuario, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.4. Pruebas de aceptación para el requisito Modificar usuario.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar usuario
Propósito	Probar que se puede modificar un usuario en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Usuarios/Gestionar usuarios**.
Entrada	Se intenta modificar un usuario con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica el usuario, si las entradas son inválidas el sistema emite un mensaje informándolo.

Resultado de la prueba	

4.5.3.5. Pruebas de aceptación para el requisito Eliminar usuario.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar usuario
Propósito	Probar que se puede eliminar un usuario en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Usuarios/Gestionar usuarios**. Deben existir usuarios en el sistema.
Entrada	Se intenta eliminar un usuario.
Resultado esperado	Se elimina el usuario del sistema o se cancela la acción.
Resultado de la prueba	

4.5.3.6. Pruebas de aceptación para el requisito Adicionar contenido informativo.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar contenido informativo
Propósito	Probar que se puede adicionar un nuevo artículo de contenido al sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Informaciones/Gestionar artículos** o menú **Informaciones/Nuevo.**
Entrada	Se intenta adicionar un nuevo artículo de contenido con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona el artículo de contenido, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.7. Pruebas de aceptación para el requisito Modificar contenido informativo.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar contenido informativo
Propósito	Probar que se puede modificar un artículo de contenido en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Informaciones/Gestionar artículos**. Deben existir artículos en el sistema.

Entrada	Se intenta modificar un artículo de contenido con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica el artículo de contenido, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.8. Pruebas de aceptación para el requisito Eliminar contenido informativo.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar contenido informativo
Propósito	Probar que se puede eliminar un artículo de contenido en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Informaciones/Gestionar artículos**. Deben existir artículos en el sistema.
Entrada	Se intenta eliminar un artículo de contenido.
Resultado esperado	Se elimina el artículo de contenido del sistema o se cancela la acción.
Resultado de la prueba	

4.5.3.9. Pruebas de aceptación para el requisito Mostrar contenido informativo.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Mostrar contenido informativo
Propósito	Probar que se muestra el contenido informativo en el sistema.
Condiciones de ejecución	El usuario se debe encontrar en la interfaz principal del sistema.
Entrada	Se intenta visualizar el contenido informativo del sistema.
Resultado esperado	Se muestra el contenido informativo del sistema.
Resultado de la prueba	

4.5.3.10. Pruebas de aceptación para el requisito Filtrar métricas.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Filtrar métricas
Propósito	Probar que se puede filtrar métricas según un criterio de búsqueda.
Condiciones de ejecución	El usuario debe encontrarse en la interfaz principal del sistema.

Entrada	Se intenta buscar información sobre las métricas filtrando con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema muestra el resultado de la búsqueda, si las entradas son inválidas el sistema emite un mensaje informando que no existen resultados de la búsqueda.
Resultado de la prueba	

4.5.3.11. Pruebas de aceptación para el requisito Mostrar últimas métricas.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Mostrar últimas métricas
Propósito	Probar que se muestran las últimas cinco métricas introducidas en el sistema.
Condiciones de ejecución	El usuario debe encontrarse en la interfaz principal del sistema.
Entrada	Se intenta visualizar las últimas cinco métricas introducidas en el sistema.
Resultado esperado	Se muestran las últimas cinco métricas introducidas en el sistema.
Resultado de la prueba	

4.5.3.12. Pruebas de aceptación para el requisito Adicionar métrica base.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar métrica base
Propósito	Probar que se puede adicionar una métrica base al sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Métricas/Métricas base/Gestionar métricas**.
Entrada	Se intenta adicionar una métrica base al sistema con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona la métrica. Si las entradas son inválidas el sistema muestra un mensaje informándolo.
Resultado de la prueba	

4.5.3.13. Pruebas de aceptación para el requisito Adicionar métrica derivada.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar métrica derivada
Propósito	Probar que se puede adicionar una métrica derivada al sistema.
Condiciones	El usuario debe tener los permisos necesarios para ejecutar el requisito.

de ejecución	Se debe seleccionar el menú **Métricas/Métricas derivada/Gestionar métricas**.
Entrada	Se intenta adicionar una métrica derivada al sistema con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona la métrica. Si las entradas son inválidas el sistema muestra un mensaje informándolo.
Resultado de la prueba	

4.5.3.14. Pruebas de aceptación para el requisito Modificar métrica base.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar métrica base
Propósito	Probar que se puede modificar una métrica base en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Métricas/Métricas base/Gestionar métricas**. Deben existir métricas en el sistema.
Entrada	Se intenta modificar una métrica base con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica la métrica, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.15. Pruebas de aceptación para el requisito Modificar métrica derivada.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar métrica derivada
Propósito	Probar que se puede modificar una métrica derivada en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Métricas/Métricas derivada/Gestionar métricas**. Deben existir métricas en el sistema.
Entrada	Se intenta modificar una métrica derivada con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica la métrica, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.16. Pruebas de aceptación para el requisito Eliminar métrica.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar métrica
Propósito	Probar que se puede eliminar una métrica en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Métricas/Métrica base o Métrica derivada/Gestionar métricas**. Deben existir métricas en el sistema.
Entrada	Se intenta eliminar una métrica.
Resultado esperado	Se elimina la métrica del sistema o se cancela la acción.
Resultado de la prueba	

4.5.3.17. Pruebas de aceptación para el requisito Adicionar clasificación.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar clasificación
Propósito	Probar que se puede adicionar una clasificación al sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Clasificaciones/Gestionar clasificaciones**.
Entrada	Se intenta adicionar una clasificación al sistema con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona la clasificación. Si las entradas son inválidas el sistema muestra un mensaje informándolo.
Resultado de la prueba	

4.5.3.18. Pruebas de aceptación para el requisito Modificar clasificación.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar clasificación
Propósito	Probar que se puede modificar una calcificación en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el **Nomencladores/Clasificaciones/Gestionar clasificaciones**. Deben existir clasificaciones en el sistema.

Entrada	Se intenta modificar una clasificación con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica la clasificación, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.19. Pruebas de aceptación para el requisito Eliminar clasificación.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar clasificación
Propósito	Probar que se puede eliminar una clasificación en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Clasificaciones/Gestionar clasificaciones**. Deben existir clasificaciones en el sistema.
Entrada	Se intenta eliminar una clasificación.
Resultado esperado	Se elimina la clasificación del sistema o se cancela la acción.
Resultado de la prueba	

4.5.3.20. Pruebas de aceptación para el requisito Adicionar meta.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar meta
Propósito	Probar que se puede adicionar una meta al sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Metas/Gestionar metas**.
Entrada	Se intenta adicionar una meta al sistema con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona la meta. Si las entradas son inválidas el sistema muestra un mensaje informándolo.
Resultado de la prueba	

4.5.3.21. Pruebas de aceptación para el requisito Modificar meta.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar meta
Propósito	Probar que se puede modificar una meta en el sistema.

Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el **Nomencladores/Metas/ Gestionar metas**. Deben existir metas en el sistema.
Entrada	Se intenta modificar una meta con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica la meta, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.22. Pruebas de aceptación para el requisito Eliminar meta.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar meta
Propósito	Probar que se puede eliminar una meta en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Metas /Gestionar metas**. Deben existir metas en el sistema.
Entrada	Se intenta eliminar una meta.
Resultado esperado	Se elimina la meta del sistema o se cancela la acción.
Resultado de la prueba	

4.5.3.23. Pruebas de aceptación para el requisito Adicionar tipo de proyecto.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Adicionar tipo de proyecto
Propósito	Probar que se puede adicionar un tipo de proyecto en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Tipos de proyecto/Gestionar tipos**.
Entrada	Se intenta adicionar un tipo de proyecto con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema adiciona el tipo de proyecto. Si las entradas son inválidas el sistema muestra un mensaje informándolo.
Resultado de la prueba	

4.5.3.24. Pruebas de aceptación para el requisito Modificar tipo de proyecto.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Modificar tipo de proyecto

Propósito	Probar que se puede modificar un tipo de proyecto en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el **Nomencladores/Tipos de proyecto/Gestionar tipos**. Deben existir tipos de proyectos en el sistema.
Entrada	Se intenta modificar un tipo de proyecto con entradas válidas e inválidas.
Resultado esperado	Si las entradas son válidas el sistema modifica el tipo de proyecto, si las entradas son inválidas el sistema emite un mensaje informándolo.
Resultado de la prueba	

4.5.3.25. Pruebas de aceptación para el requisito Eliminar tipo de proyecto.

Pruebas de aceptación: Diseño de casos de prueba.	
Requisito	Eliminar tipo de proyecto
Propósito	Probar que se puede eliminar un tipo de proyecto en el sistema.
Condiciones de ejecución	El usuario debe tener los permisos necesarios para ejecutar el requisito. Se debe seleccionar el menú **Nomencladores/Tipos de proyecto /Gestionar tipos**. Deben existir tipos de proyecto en el sistema.
Entrada	Se intenta eliminar un tipo de proyecto.
Resultado esperado	Se elimina el tipo de proyecto del sistema o se cancela la acción.
Resultado de la prueba	

4.6. Conclusiones.

En este capítulo se definió en términos de componentes de implementación la organización de las clases y objetos que componen el sistema, organizando los mismos en diagramas de componentes según la arquitectura definida. También se describió mediante el diagrama de despliegue la distribución física del sistema, representando como se organiza su funcionalidad entre los nodos de cómputos. Además se caracterizó el ambiente de implantación del sistema y se realizaron las pruebas de aceptación con el cliente para validar la propuesta.

CONCLUSIONES

Luego de haber culminado la investigación realizada para el desarrollo del Sistema para la Gestión de Métricas y teniendo en cuenta que el mismo estuvo guiado por las tareas de la investigación, como colofón de este trabajo queda lo siguiente:

- Se realizó un estudio acerca del uso de las métricas, su composición y clasificación, además de un análisis relacionado con los procesos de medición.
- Se hizo un análisis de las tecnologías vigentes para desarrollar este tipo de sistema, lo que permitió seleccionar la metodología, plataforma y herramientas a utilizar en el desarrollo del software.
- Se modeló el sistema, obteniendo así todos los artefactos necesarios para la implementación del mismo.
- Se realizó el diseño e implementación de un sistema que permite la gestión de las métricas definidas por el GM de la DCS, así como el diseño de la base de datos que lo soportará.
- Se validó la propuesta con los Usuarios Finales, lo que permitió hacer las recomendaciones necesarias para un futuro enriquecimiento de la misma.

Por tales razones se puede concluir en que se alcanzó satisfactoriamente el objetivo propuesto: desarrollar un sistema para la gestión de las métricas que permita mantener la información referente a las mismas actualizada para la consulta de los interesados.

RECOMENDACIONES

Al terminar este trabajo es necesario realizar algunas recomendaciones, las cuales podrán tenerse en cuenta en la futura utilización del sistema y porque no en un futuro desarrollo e incremento de sus funcionalidades.

- Poner el sistema a prueba durante un período de tiempo determinado, para comprobar su correcto funcionamiento.
- Implantar el sistema para su uso por el GM.
- Continuar profundizando en el estudio de las necesidades existentes con respecto al uso de las métricas, con el objetivo de añadir nuevas funcionalidades.

BIBLIOGRAFÍA

1. **Calero. C.** Métricas del software: Conceptos básicos, definición y formalización. [En línea] Octubre de 2006. [Citado el: 15 de enero de 2009] http:// eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Metricas4.pdf
2. **Zulueta, Y.** Introducción de técnicas del Personal Software Process desde los primeros años en la formación del ingeniero informático., [En línea] 14 de mayo de 2007, [Citado el 15 de enero de 2009] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/pelaez_r_jj/
3. **Pressman, R.** *Ingeniería de software. Un enfoque Práctico,* 2005.
4. **Gracia, J.** *CMM - CMMI Nivel 2.* [En línea] 26 de noviembre de 2005, [Citado el: 16 de enero de 2009] http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php
5. **Morales, N y González, S.** *Procedimiento para realizar la medición y análisis en un proyecto.* [Citado el: 16 de enero de 2009] http://www.monografias.com/trabajos64/procedimiento-medicion-analisis-proyecto/procedimiento-medicion-analisis-proyecto.shtml
6. **Asociación Española de Métricas de Sistemas Informáticos (AEMES)**, *Utilización del repositorio ISBSG para la estimación y benchmarking de proyectos de TI.,* [En línea] 2008, [Citado el 16 de enero de 2009] http://www.aemes.org/eventos/seminarios/2008/ISBSG_para_AEMES.pdf
7. *Programación por capas*., [Citado el 19 de enero de 2009] http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles
8. **Galli, R.,** *Desarrollo Web Extremo*., [En línea] 16 de julio de 2001, [Citado el 19 de enero de 2009] http://bulma.net/body.phtml?nIdNoticia=734
9. Sistema de gestión de base de datos., [Citado el 19 de enero de 2009] http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos
10. **Welling, L., Thomson, L.**, *Desarrollo Web con PHP y MySQL*., Anaya Multimedia, Agosto de 2003.
11. **Cameron, Nadia.** ComputerWorld the voice of it management. *ComputerWorld the voice of it management.* [En línea] 20 de Enero de 2003. [Citado el: 19 de Enero de 2009.] http://www.computerworld.com.au/index.php?id=760310963.
12. **Institute, S. E.**, *Capability Maturity Model Integration.,* 2002.
13. *Programación por capas*., [Citado el 19 de enero de 2009] http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles

BIBLIOGRAFÍA

14. **Masip, David.** *Desarrollo Web.com.* [En línea] 2006, [Citado el: 19 de Enero de 2009] http://www.desarrolloweb.com/articulos/840.php
15. Lenguaje de programación Java., [Citado el 21 de febrero de 2009] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java
16. Qué es Oracle., [Citado el 21 de febrero de 2009] http://www.desarrolloweb.com/articulos/840.php
17. Oracle., [Citado el: 21 de febrero de 2009] http://es.wikipedia.org/wiki/Oracle
18. Microsoft SQL Server., [Citado el 21 de febrero de 2009] http://www.microsoft.com/latam/sqlserver/default.aspx
19. **Jacobson, I.; Booch; G.; Rumbaugh, J.** "*El Proceso Unificado de Desarrollo de Software.*" AddisonWesley, 2000.
20. **Hernández, A.** "*Un método para el diseño de la base de datos a partir del modelo orientado a objetos."* [Citado el 15 de abril de 2009] http://redalyc.uaemex.mx/redalyc/pdf/615/61570402.pdf
21. Principios del Diseño Universal o Diseño para Todos., [Citado el 15 de abril de 2009] http://www.ascun.org.co/eventos/seminariocampus/principiosdisenouniv.pdf

REFERENCIAS BIBLIOGRÁFICAS

[1] **Westfall, L.** *Software metrics that meet your information needs*, 1995.

[2] **Kan, S. H.** "*Metrics and Models in Software Quality Engineering"*, AddisonWesley, 2000.

[3] **Pérez, I.** *Métricas para el control de proyectos de software.* Trabajo de Diploma para optar por el título de Ingeniero Informático. Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2002.

[4] **Humphrey, W. S.** "*A discipline for software engineering"*, AddisonWesley, 1995.

[5] **Pressman, R.** *Ingeniería de software. Un enfoque Práctico,* 2005

[6] **Román, M.** *Mediciones prácticas de software y sistemas (PSM): Una propuesta para la producción de software en la UCI.* Universidad de las Ciencias Informáticas (UCI), 2007.

[7] **Zulueta, Y.** Introducción de técnicas del Personal Software Process desde los primeros años en la formación del ingeniero informático., [En línea] 14 de mayo de 2007, [Citado el 15 de enero de 2009] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/pelaez_r_jj/

[8] *Repositorio*., [Citado el 16 de enero de 2009] http://es.wikipedia.org/wiki/Repositorio

[9] **Gracia, J.** *CMM - CMMI Nivel 2.* [En línea] 26 de noviembre de 2005, [Citado el: 16 de enero de 2009] http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php

[10] **Fahnle, P.** *ASP en Castellano. ASP en Castellano.* [En línea] Diciembre de 2000. [Citado el: 18 de enero de 2008.] http://www.programacion.com/asp/articulo/aspnet_quees/.

[11] **Dondo, A.** PHP en Castellano. *PHP en Castellano.* [En línea] 03 de febrero de 2005. [Citado el: 15 de enero de 2008.] http://www.programacion.com/php/articulo/porquephp/.

[12] **Gallego Vázquez, José A..** *Desarrollo Web con PHP y MySQL.*, Anaya Multimedia, 2003.

[13] *Usabilidad y arquitectura del software*., [En línea] 09 de septiembre de 2004] , [Citado el 10 de marzo de 2009] http://www.desarrolloweb.com/articulos/1622.php

[14] *Symfony.*, [Citado el 10 de marzo de 2009] http://es.wikipedia.org/wiki/Symfony

[15] *Ventajas de PostgreSQL.*, [En Línea] 31 de mayo de 2003, [Citado el: 19 de enero de 2009] http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html

[16] *Microsoft SQL Server.*, [Citado el 21 de febrero de 2009] http://es.wikipedia.org/wiki/MS_SQL

[17] **Sitio Web Linalco, especialistas en Linux y software libre.** *Servicios y soluciones linalco.* [Citado el 19 de enero de 2009] http://www.linalco.com/soluciones.html

REFERENCIAS BIBLIOGRÁFICAS

[18] **Canós, José H.; Letelier, P.; Penadés, María C.** *Metodologías Ágiles en el Desarrollo de Software.* DSIC -Universidad Politécnica de Valencia [Citado el 1 de abril de 2009] www.willydev.net/descargas/prev/TodoAgil.pdf

[19] **Hernández, Pedro V.** *El Proceso Unificado de Racional (RUP) y su relación con las técnicas y métodos de la ingeniería y usabilidad del software.*, [En línea] 2004-2005. [Citado el 20 de enero de 2009] http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Hernández.pdf

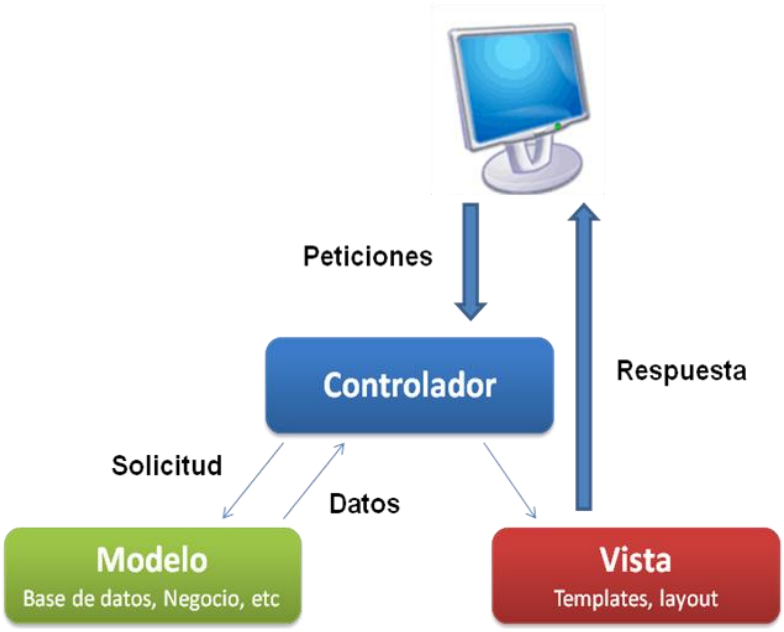
[20] **Álvarez, M.** *Evaluando Zend Studio***.** [Citado el 20 de enero de 2009] http://www.maestrosdelweb.com/editorial/zendstudio/

[21] **Moreno, G.** Monografía: Ingeniería de Software UML. [Citado el 20 de enero de 2009] http://www.monografias.com/trabajos5/insof/insof.shtml

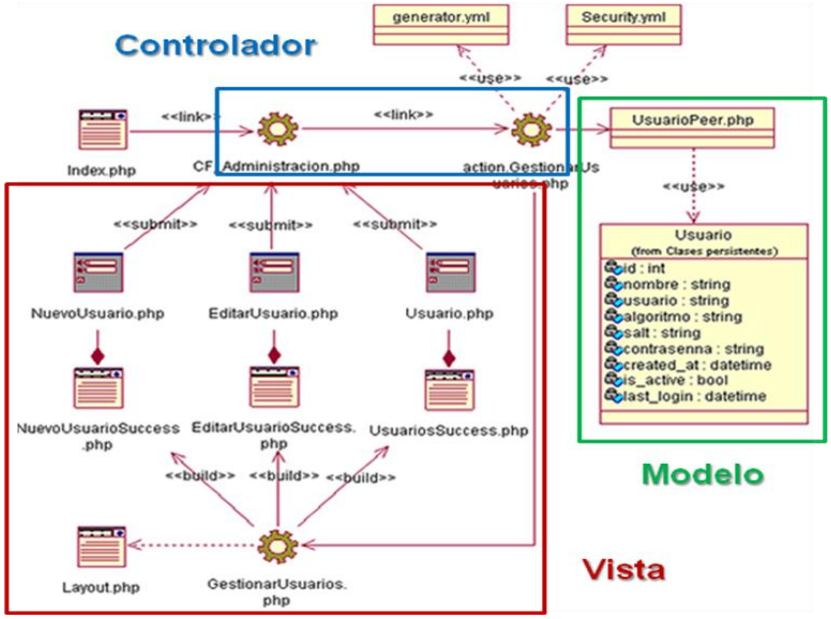
[22] **Jacobson, I.; Booch; G.; Rumbaugh, J.** "*El Proceso Unificado de Desarrollo de Software.*" AddisonWesley, 2000.

ANEXOS

Anexo 1. Patrón Modelo Vista Controlador (MVC).

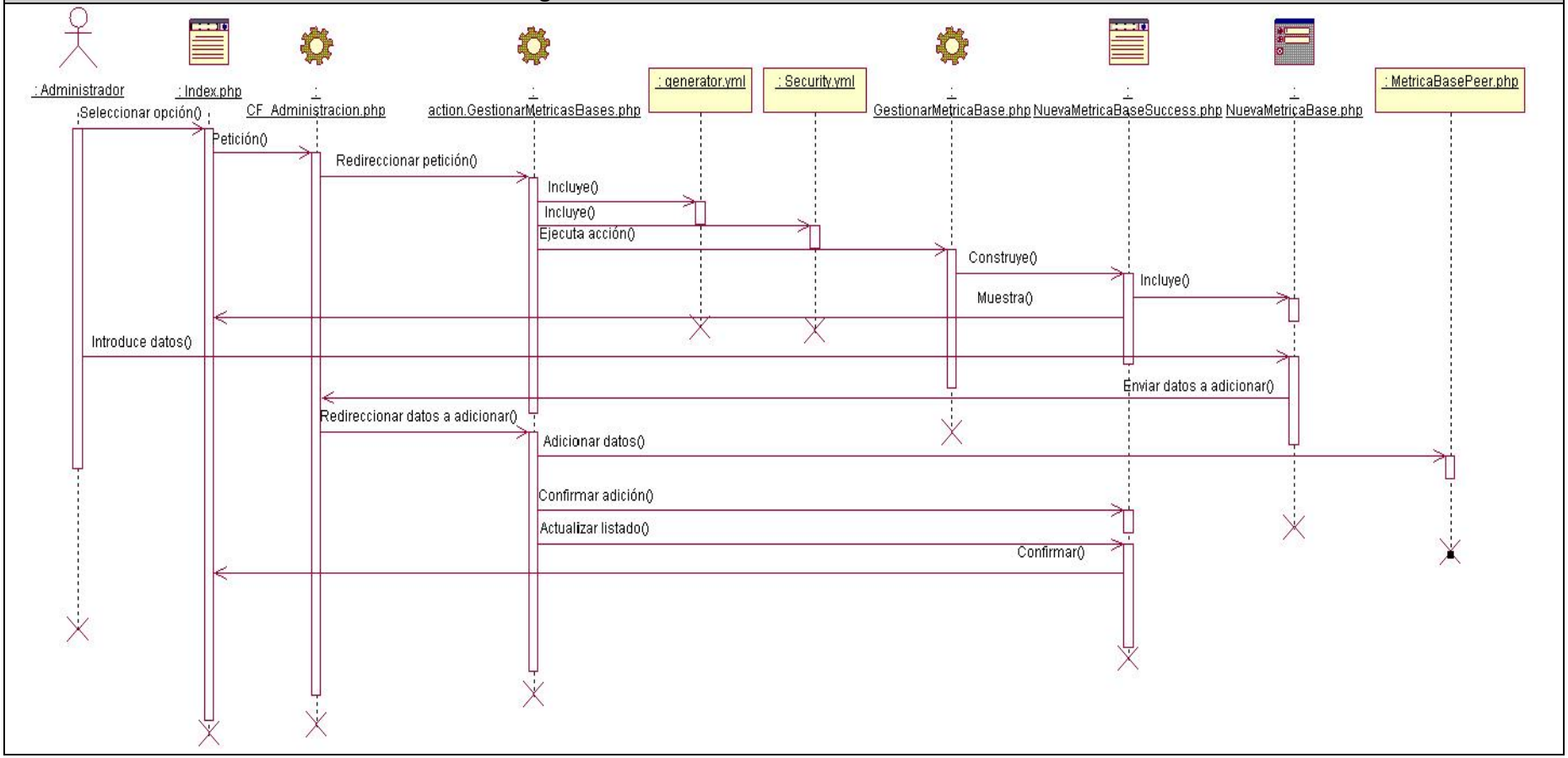


Anexo 2. Uso del patrón MVC en los diagramas de clases del diseño.



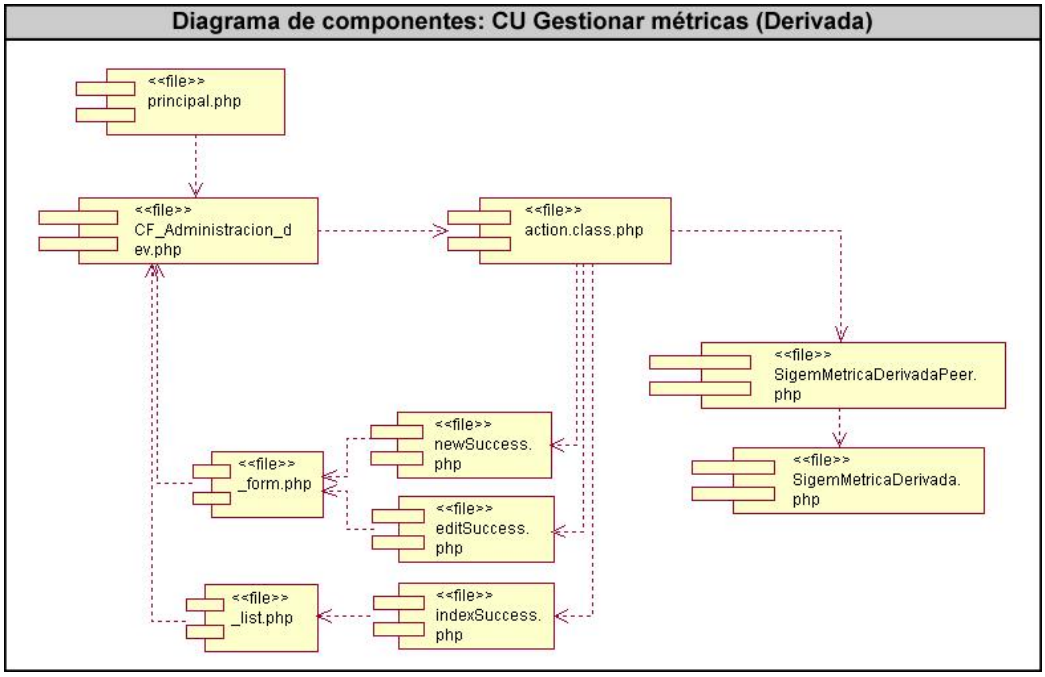
Anexo 3. Ejemplo de diagrama de secuencia.

Diagrama de secuencia: Adicionar métrica base.

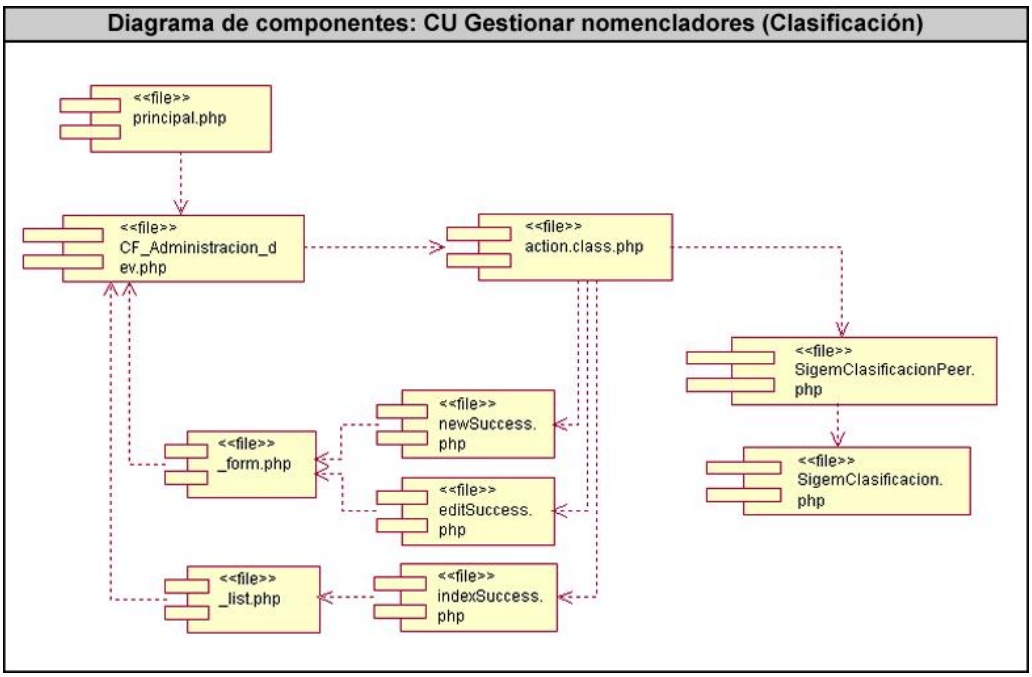


Anexo 4. Realización de CU mediante diagramas de componentes.

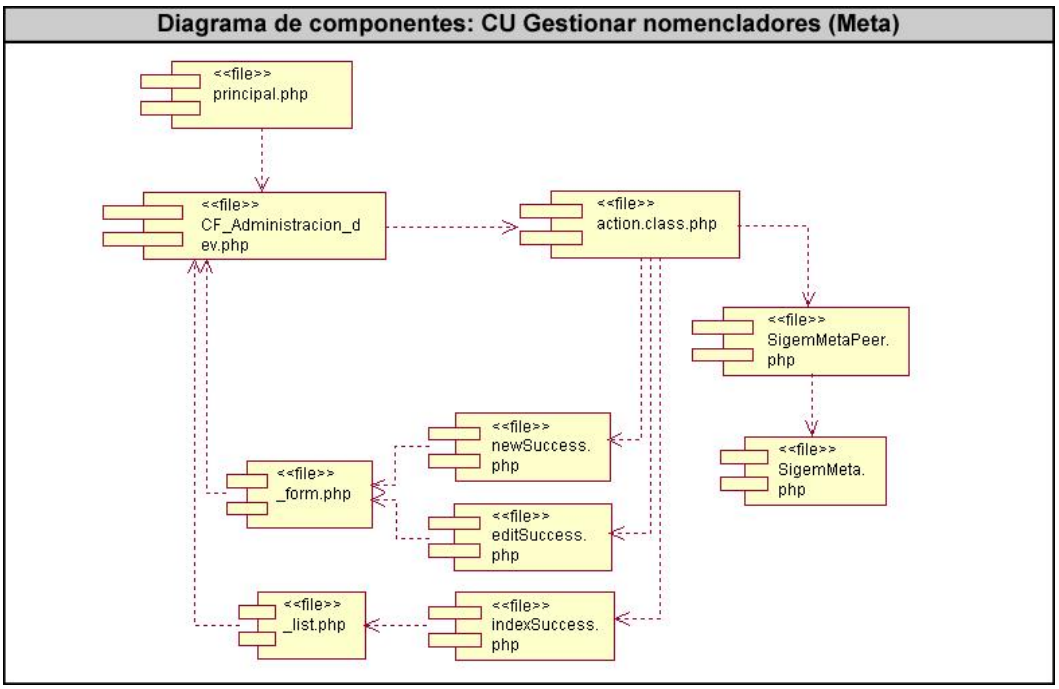
Anexo 4.1. CU Gestionar métricas (Derivada).



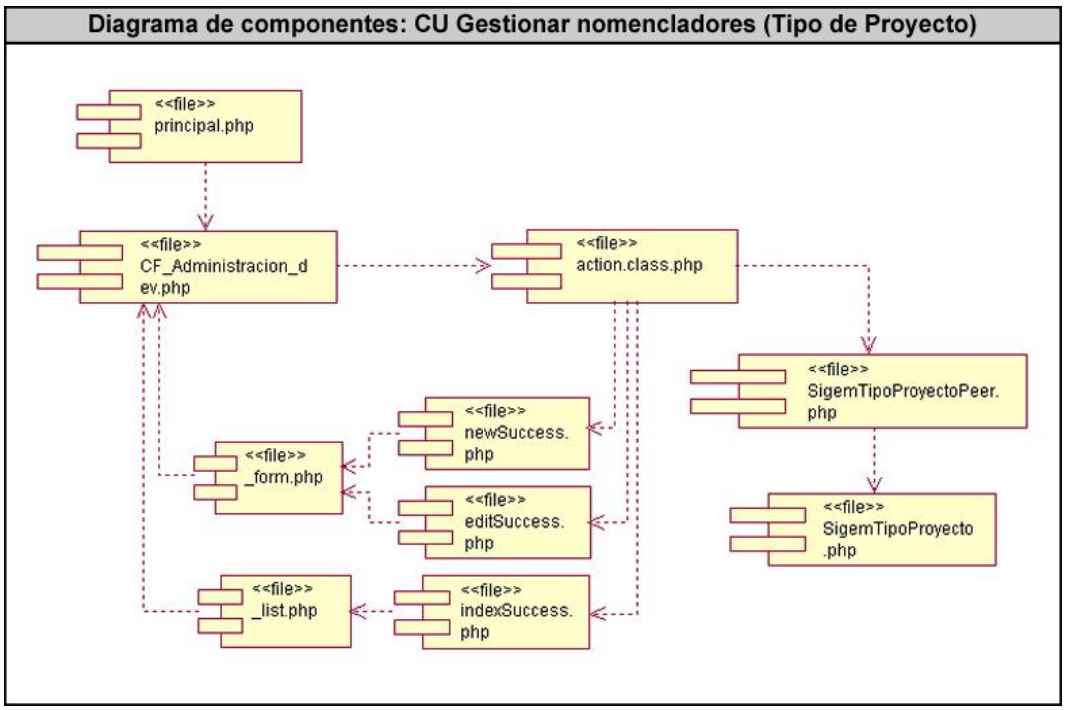
Anexo 4.2. CU Gestionar nomencladores (Clasificación).



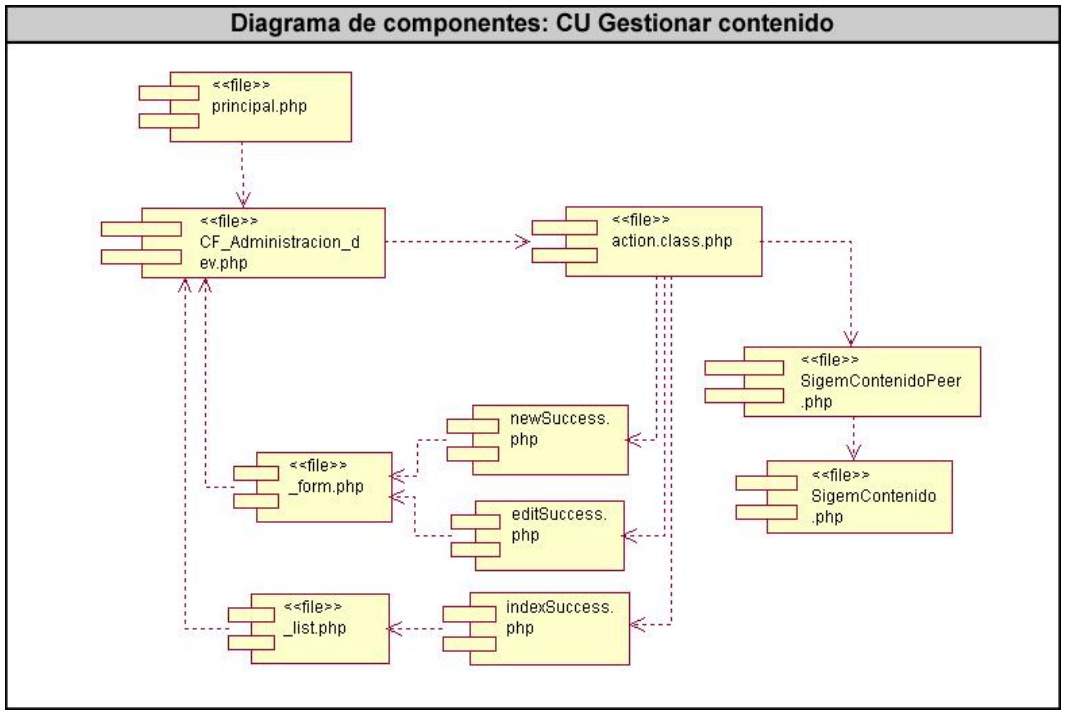
Anexo 4.3. CU Gestionar nomencladores (Meta).



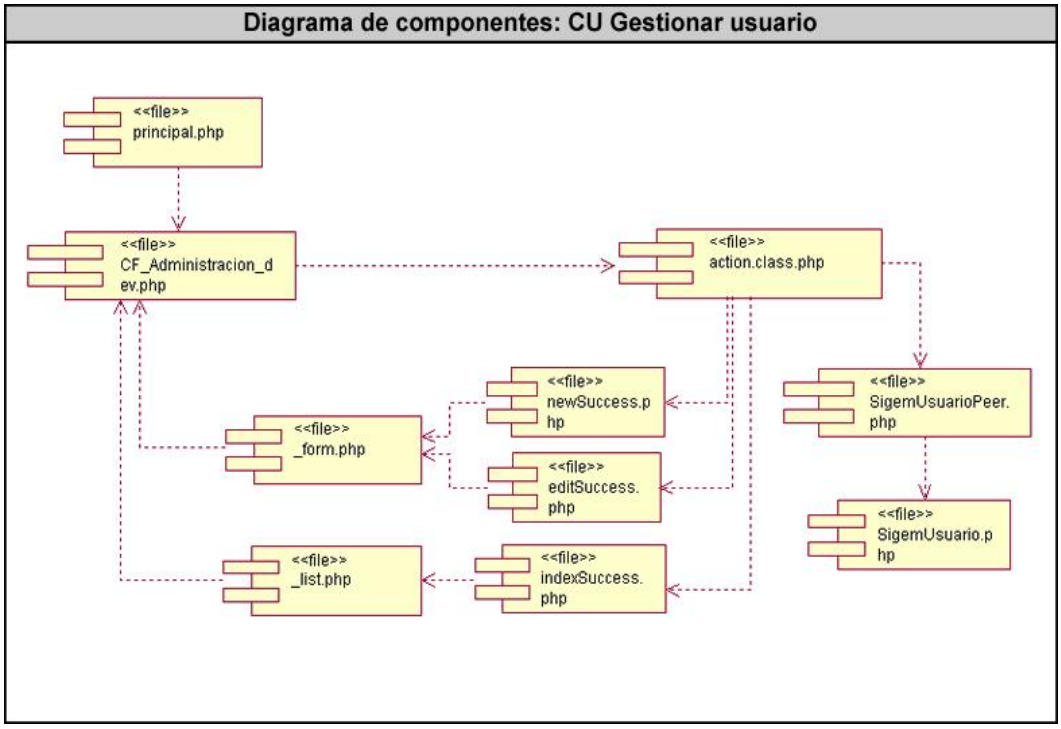
Anexo 4.4. CU Gestionar nomencladores (Tipo de proyecto).



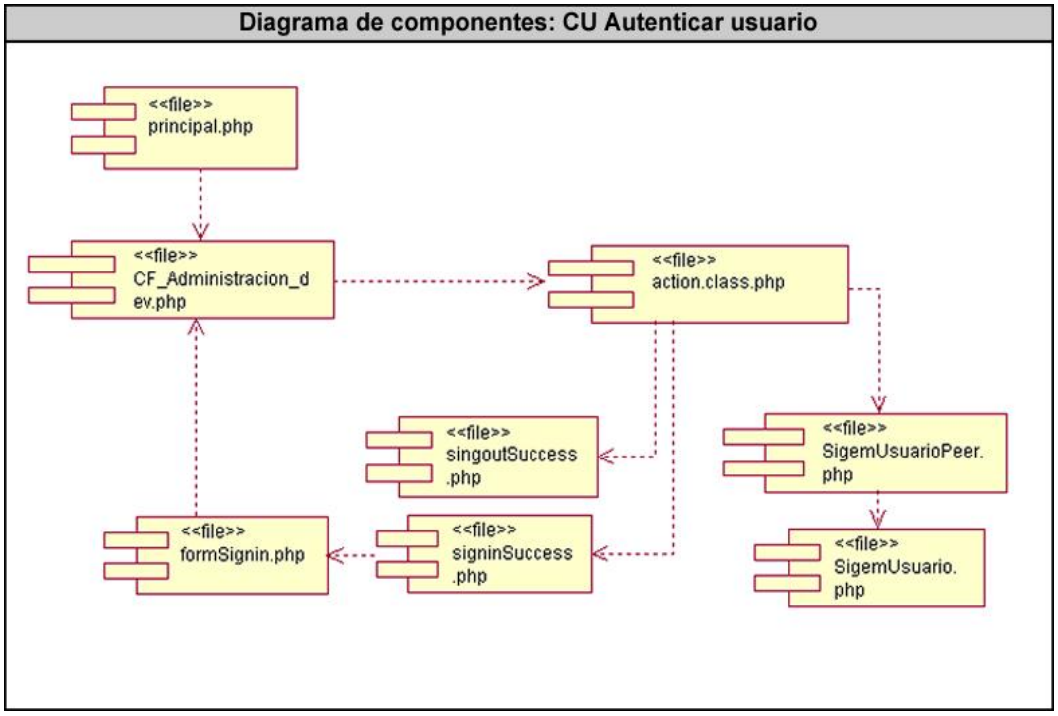
Anexo 4.5. CU Gestionar contenido.



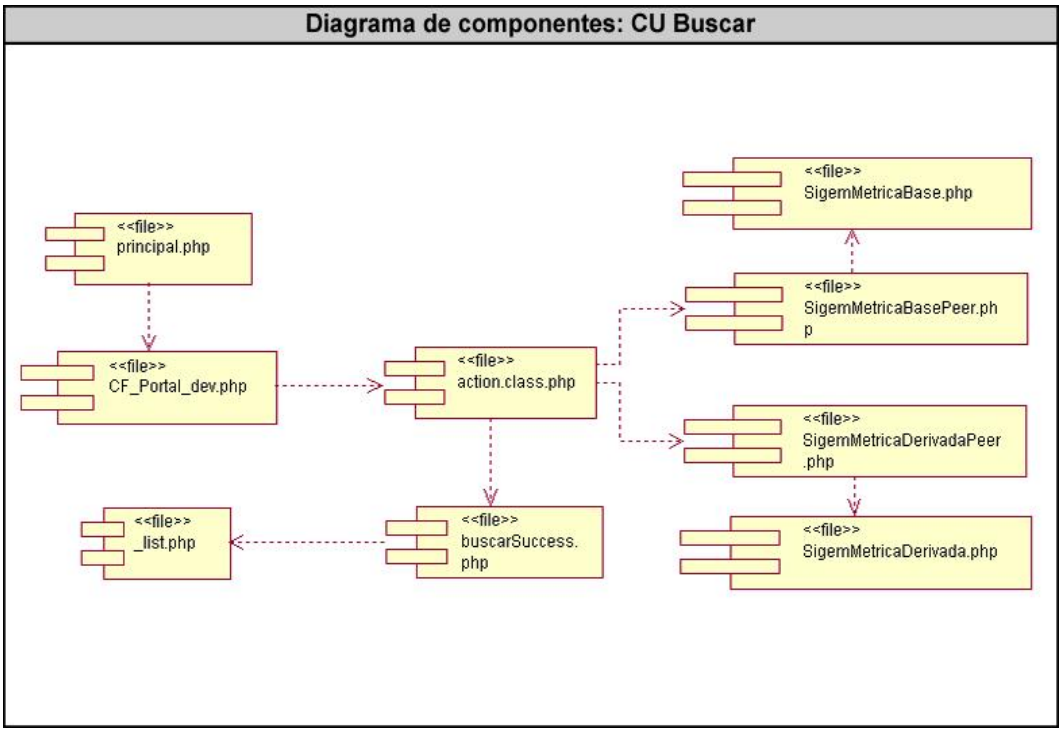
Anexo 4.6. CU Gestionar usuario.



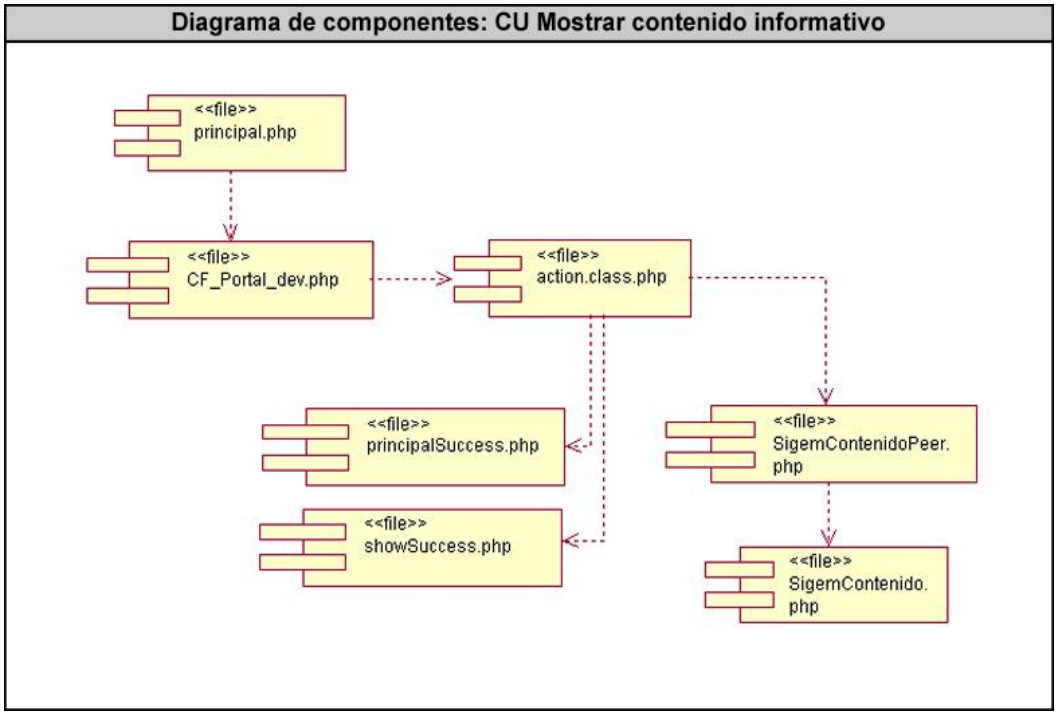
Anexo 4.7. CU Autenticar usuario.



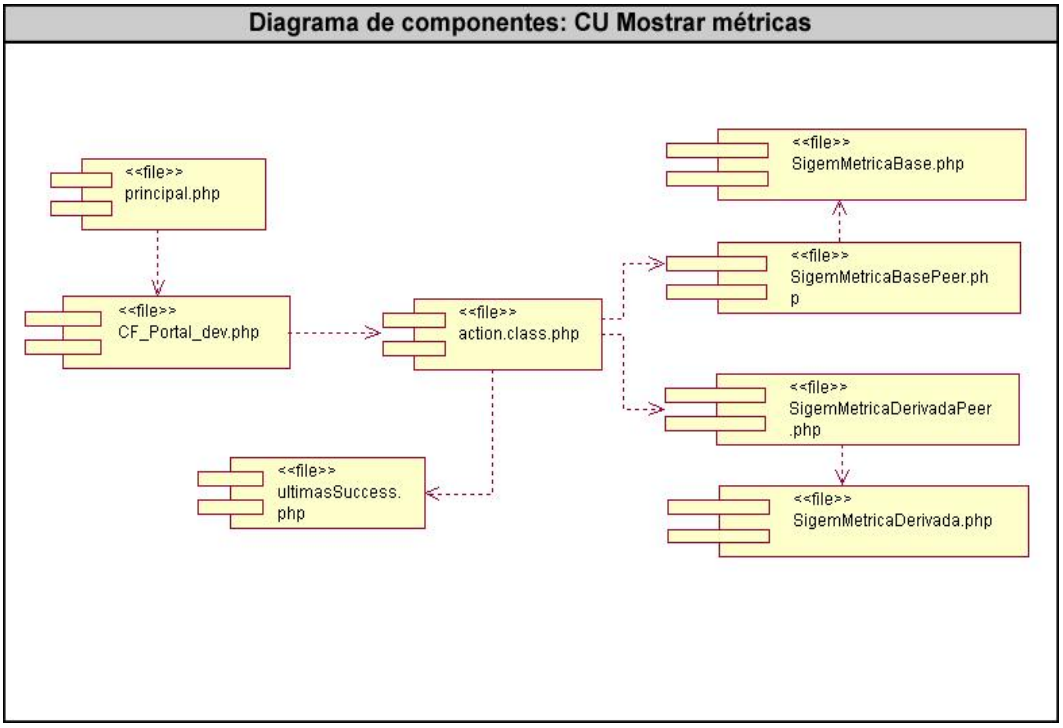
Anexo 4.8. CU Buscar.



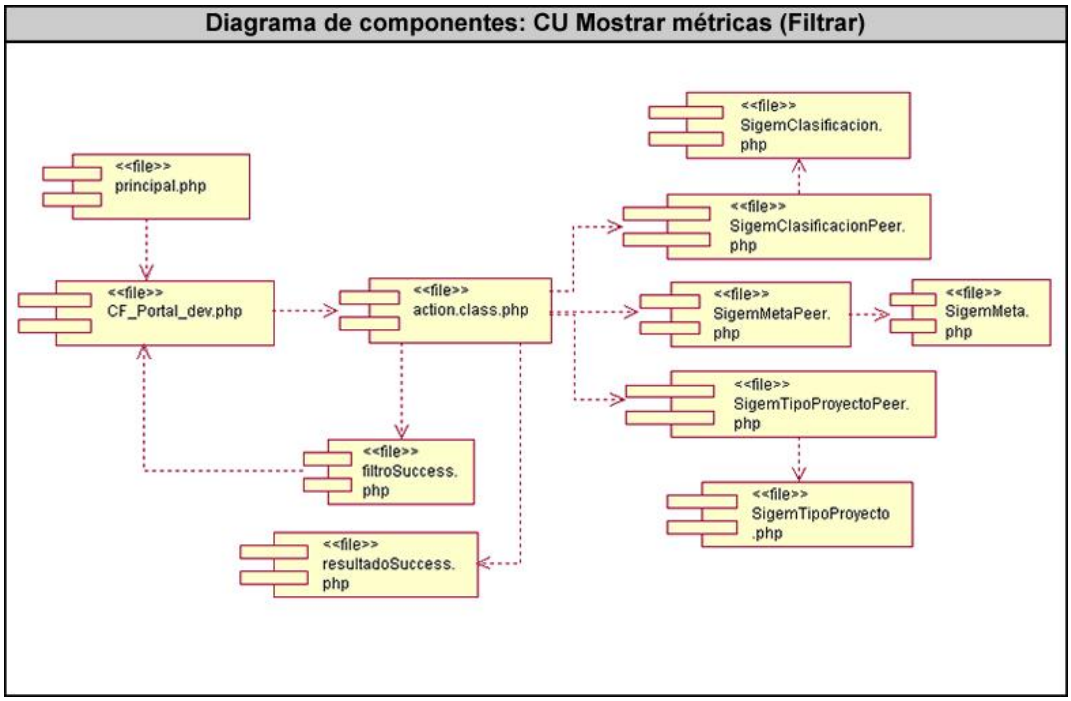
Anexo 4.9. CU Mostrar contenido informativo.



Anexo 4.10. CU Mostrar métricas.



Anexo 4.11. CU Mostrar métricas (Filtrar).



GLOSARIO

API: *Application Programming Interface.* En español: Interfaz de Programación de Aplicaciones. Es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

ASP: *Active Server Pages*. Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). Con ASP se pueden combinar páginas HTML, scripts y objetos COM.

CASE: *Computer Aided Software Engineering.*

CGI: *Common Gateway Interface.* Estándar para transferir datos entre el cliente y el programa.

CMMI: *Capability Maturity Model Integration.* En español: Modelo Integrado de Madurez y Capacidad.

DCS: Dirección de Calidad de Software de la Universidad de las Ciencias Informáticas.

GM: Grupo de Métricas perteneciente a la DCS.

GNOME: *GNU Network Object Model Environment. E*s un entorno de escritorio e infraestructura de desarrollo para sistemas operativos Unix/Linux, compuesto enteramente de software libre.

GPL: Licencia Pública de GNU.

HTML: *Hypertext Markup Language.* Lenguaje de marcas usado para escribir documentos para servidores World Wide Web. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

HTTP: *Hypertext Transfer Protocol.* Es un protocolo diseñado para transferir hipertextos, páginas Web o páginas HTML.

GLOSARIO

ICSW: Industria Cubana del Software.

Informix: SGBD adquirido por IBM en el 2001 a una compañía del mismo nombre.

ISAPI: *Internet Server API.* La API para servidores de Internet.

ISO/IEC 15939: Ingeniería de software - Proceso de medición de Software. Norma de la Organización Internacional para la Estandarización.

Java: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems.

JSP: *Java Server Pages*. La tecnología JSP, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página web. Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático.

KDE: *K Desktop Environment.* En español: Entorno de Escritorio K. Es un entorno de escritorio e infraestructura de desarrollo para sistemas Unix/Linux.

MA: Medición y Análisis. Área de proceso del nivel 2 CMMI.

MSSQL: *Microsoft SQL Server.* Es un sistema de gestión de bases de datos relacionales.

MVC: *Modelo Vista Controlador.* Patrón de arquitectura de software.

MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.

Oracle: Sistema de gestión de base de datos relacional desarrollado por Oracle Corporation.

PERL: *Practical Extraction and Report Language.* Es un lenguaje de programación desarrollado por Larry Wall inspirado en otras herramientas de UNIX como son: sed, grep, awk, c-shell.

GLOSARIO

PHP: *Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y scripts.

PostgreSQL: Es un SGBD Objeto-Relacionales (ORDBMS) libre.

PSM: *Practical Software and Systems Measurement*. En español: Mediciones Prácticas de Software y Sistemas.

RUP: *Rational Unified Process*. En español: Proceso Unificado de Desarrollo. Metodología para el desarrollo de Software.

SGBD: Sistemas de gestión de base de datos.

TIC: Tecnologías de la Información y las Comunicaciones.

UCI: Universidad de las Ciencias Informáticas.

UML: *Unified Modeling Language.* Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.