# Universidad de las Ciencias Informáticas

FACULTAD 15


**Título:** Análisis y Diseño de una herramienta de configuración para cake bake.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

**Autor(es):**
Maelys Benítez Vega
Osnielky Roque Paz

**Tutor(es):**
Ing. Yosveny Escalona Escalona

Ciudad de la Habana, Junio del 2010

*"Muchos me dirán aventurero, y lo soy, sólo que de un tipo diferente y de los que ponen el pellejo para demostrar sus verdades."*

## DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Maelys Benítez Vega

_____

Firma del Autor

Osnielky Roque Paz

_____

Firma del Autor

Ing. Yosveny Escalona Escalona

_____

Firma del Tutor

**DATOS DE CONTACTO**

Síntesis del tutor:

**Ing.** Yosveny Escalona Escalona **(yescalona@uci.cu):** Graduado de Ingeniería en Ciencias Informáticas en el año 2006 en la Universidad de las Ciencias Informáticas. Profesor del Departamento de Programación de la Facultad 15.

## AGRADECIMIENTOS

*Agradecimientos Osnielky*

*Agradezco a Dios, a la revolución y al comandante por estos 5 años de buenos y malos momentos.*

*Agradezco a mis abuelos por quererme como un hijo, yo también los quiero como mis padres.*

*Agradezco a mis padres, en especial a mi mama, por seguir de cerca todos mis pasos.*

*Agradezco a toda mi familia, a mi tita, a todos en Cabaiguán en especial a Nico y Zena.*

*A mi novia por todo el apoyo incondicional y todo el amor entregado.*

*A todos los amigos que el tiempo ha traído y a los que se ha llevado en estos 5 años.*

*A la gente de mi barrio, Leti, Raúl, Papín, Alie, luisa, Tony, Ivey, mis amigos de la infancia, a Carlos por sus consejos que de mucho me han servido.*

## *Agradecimientos Maelys*

*A mi **mamá** por ser la mejor madre del mundo, por apoyarme en todos los problemas que la vida nos pone en nuestro camino, por su gran amor y confianza que siempre me ha demostrado, por ser mi guía y mi ángel...Te quiero….*

*A mi **papá** por su comprensión y confianza en mí...por ser mi mejor amigo, por ser el hombre que más quiero en la vida….por ayudarme cuando lo he necesitado… Tú eres mi guardián.*

*A mi **hermanito Ernesto** pq lo quiero con la vida, pq es parte de mi…por ser el mejor hermano del mundo…*

*A mi **abuela Milagro** por apoyarme, por ser como mi segunda madre….por cuidarme desde chiquita.*

*A mis otros **tres abuelos** (papa Antonio, mama Julia y papa José Ramón) donde quiera que estén, siempre los llevo en mí, ellos forman parte también de que sea la mujer que soy.*

*A **Yamil** por ser más que mi primo mi hermano… por saber que puedo contar con su apoyo…*

*A **mi novio** por estar a mi lado, por su apoyo, comprensión, por ayudarme en todo lo que he necesitado. Por hacerme diferente, por aguantar todas mis malas crianzas. Por hacerme Feliz de una forma u otra, por su dedicación, por lograr que esta etapa en mi vida nunca se borre de mi me mente, eres lo que más quiero en esta vida. Te Amo.*

*A mi **tía Nina**, ella es muy especial para mi…gracias por saber que puedo contar contigo.*

*A todos mis tíos que de una forma u otra me han brindado su ayuda y comprensión .En especial (Ruso, Tío Mime, Tata, Nene, tío Tony, Tía Nidia, Manolo, mi tía Esperanza, FEFA).*

*A todos mis **primos** espacialmente (Dayana, Nine, Nana, Adrian, Alexander, Eliany, Alicia, Alejandro, Yosvany, Livan Y William).*

*A **Susana, Pantaleon, Yamilka, Susser** .Ellos son mi otra familia, por su apoyo en estos 5 años de mi carrera y en mi vida…Susana eres mi madre postiza.*

*A **Regla, Yairin y Adalberto** porque me han ayudado en todo lo que ha estado a su alcance, pq siempre supe que podía contar con ellos.*

*A **Mara** por ser más que mi mejor amiga estos 5 años ser mi hermana, por apoyarme siempre que tuve algún problema, por su cariño.*

*A **Maylin y Vilmary** por ser mis mejores amigas…por todos sus buenos consejos que me han dado y que me han ayudado a salir adelante.*

*A todos **mis amigos** de la universidad ,por su apoyo en la realización de este trabajo a Odiel, Liset, ,Sahily,Iliana,Gretter,Andriel,Yunior,Kerlin,Lisy,Yenier,Liem,Keila,Elda,Katia, Daimi.*

*A todos **mis vecinos en** especial a Aricel, Omar y a Arelys a ella la aprecio mucho siempre me acuerdo de ti….*

## *DEDICATORIA*

### *Dedicatoria Maelys*

*Dedico este trabajo a mi mamá y mi papá pq sin ustedes este sueño no se me hubiera hecho realidad, gracias por su apoyo y comprensión…Los quiero.*
*A mi hermano Ernesto por ser mi personita más querida.*
*Y a toda mi Familia y compañeros que me apoyaron en estos 5 años de mi carrera.*

### *Dedicatoria Osnielky*

*Dedico esta tesis a mis abuelos, en especial a Nico, la dedico también a mi hijo a mis hermanos, los que están y los que están por venir.*

***RESUMEN***

El uso de los diferentes frameworks para desarrollo web se encuentra muy extendido en el mundo de la informática. Tal es el caso del framework CakePHP en el cual se utiliza la herramienta cake bake para creación y configuración de toda la estructura de los proyectos realizados en dicho framework. Cake bake es una herramienta en la cual se puede crear un proyecto perfectamente funcional. Con todas sus virtudes, dicha herramienta no muestra una interfaz con características visuales acordes a las exigencias de la actualidad y los parámetros de facilidad y usabilidad que buscan todos los posibles clientes. VisualBake pretende ser una herramienta que asuma las virtudes de la consola y combinarlas con una interfaz amigable y amena para el desarrollador.

**Palabras Claves:** CakePHP, VisualBake, frameworks, cake bake.

# ÍNDICE

## INTRODUCCIÓN

En un considerable corto período de tiempo, Internet se ha convertido en una de las herramientas de desarrollo y comunicación más usadas a través de la historia de la humanidad. Desde su surgimiento en los Estados Unidos de América (EUA), como parte de un proyecto militar, hasta la actualidad, alrededor de 694 millones de personas en el mundo tienen acceso a Internet. Muchas son las aplicaciones prácticas que han llevado a esta herramienta a tan altos niveles de interacción con los usuarios de todo el mundo, entre ellas se destacan, compartir recursos, y uno de sus objetivo es hacer que todos los programas, datos y equipo estén disponibles para cualquiera de la red que así lo solicite, sin importar la localización física del recurso y del usuario, también se ha convertido en poco tiempo en una de las oportunidades de negocio más importantes para aquellas empresas que incluyen y fomentan el comercio electrónico dada la situación peculiar en un mundo donde parece que si no estás en Internet no existes, situación a la que no escapan las empresas de nuestro país. La World Wide Web es tal vez el punto más visible de Internet y hoy en día el más usado junto con el correo electrónico, aunque también es de los más recientes. Originalmente denominado Proyecto WWW fue desarrollado en Suiza a principio de los 90. El aspecto exterior de la WWW son las conocidas páginas Web, en las cuales una ventana muestra al usuario la información que desea, en forma de texto y gráficos, con los enlaces marcados en diferente color y subrayados. La Web es el lugar de Internet que más crecimiento está experimentando últimamente: se calcula que hay más de 50 millones de páginas Web en la Red, y su número crece a un ritmo vertiginoso. El desarrollo inmenso, en cuanto a calidad y cantidad de páginas Web, está sustentado por nuevas tecnologías, lenguajes y herramientas que posibilitan un desarrollo Web cada vez más organizado y especializado. Existen numerosos lenguajes de programación empleados para el desarrollo de Aplicaciones Web, entre los que destacan: PHP, ASP, ASP.NET, Java, con sus tecnologías Java Servlets y JavaServer Pages (JSP). Uno de los lenguajes más populares y por consiguiente más usado en el mundo de la Web es PHP (acrónimo de PHP: Hypertext Pre-processor). La gran mayoría de los que desean comenzar a programar aplicaciones Web, optan por el lenguaje que es más popular y en el que todos los servidores ofrecen soporte, se trata de PHP. Este lenguaje ofrece soluciones rápidas, pero a costa de eso es un lenguaje desorganizado, no separando la lógica del negocio de la presentación. Con la aparición de PHP5 y su Orientación a Objetos, surgieron muchos Frameworks**,** entre los que se encuentra CakePHP, el cual usa el paradigma MVC. Estos entornos de trabajo que usan el Modelo Vista-

Controlador[1], separan la lógica de la presentación, salvando pues a PHP de ser tachado como un lenguaje desorganizado. CakePHP es libre y de código abierto, pero ni aún en su última versión estable ha logrado librarse de la consola en la que se crea toda la estructura de los proyectos (MVC). El trabajo de consola es muy poco atractivo para aquellos programadores que se inician, a medida que crecen los proyectos más numerosa es la cantidad de modelos, vistas y controladores que debe ser creados uno a uno mediante comandos que interpreta la consola de CakePHP, el uso de la consola supone un costo en tiempo adicional ya que el programador necesita aprender comandos para su posterior uso.

Ante la situación descrita se plantea el siguiente **problema a resolver**: Como suplir las deficiencias que presenta la consola cake bake.

Por tanto el **objeto de estudio** de esta investigación será el framework CakePHP.

De ello se deriva que el **campo de acción** donde está enmarcado será la herramienta de configuración Cake bake.

Se plantea como **objetivo general** realizar el análisis y diseño de una herramienta que permita la creación de las clases controladoras, los modelos y las vistas a través de una aplicación web.

Como **tareas investigativas** que se han trazado:

1. Caracterizar el framework CakePHP como objeto de estudio.
2. Caracterizar cake bake como parte del funcionamiento del framework CakePHP.
3. Identificar requisitos funcionales y no funcionales para su futura implementación.
4. Realizar el diseño para la herramienta de configuración de cake bake.
5. Aplicar Técnicas para la validación del diseño.

[1] Fue descrito por primera vez por el autor de “the group Gang of Four”, Dean Helman.

Como **Posibles resultados:**

- Análisis y Diseño que facilite la implementación de una herramienta web para cake bake que permita la creación de los controladores, modelos y vistas en los proyectos desarrollados en CakePHP además de su configuración.

Los **métodos teóricos** para el desarrollo de la investigación fueron los siguientes:

**Método Analítico-Sintético:** proporciona una vía para realizar un análisis detallado de toda la bibliografía necesaria para realizar la investigación, que sea necesaria en la solución del problema planteado.

**Método Histórico-Lógico:** permitirá realizar una investigación detallada de la trayectoria histórica real del framework CakePHP, su evolución y desarrollo, así como estudiar todos los problemas existentes, y encontrar la solución más eficiente.

# CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En el presente capítulo se realizó un estudio acerca de cómo se utiliza en el mundo el framework CakePHP así como sus principales potencialidades en el desarrollo Web. En el mismo se analizará la herramienta Cake Bake, identificando sus ventajas y desventajas, además de sus tendencias y formas de uso actuales se realizará una comparación con aquellos frameworks semejantes, en vistas de ofrecer mejoras a la herramienta dicha.

## 1.2 Estado del Arte

En 2005, Michal Tatarynowicz [2] escribió una mínima versión de un Framework Rápido para Aplicaciones en PHP. El encontró que ese fue el principio de un muy buen framework. Michal publicó el framework bajo la licencia MIT, mezclando Cake, y abriéndose a una comunidad de desarrolladores, quienes ahora mantienen Cake bajo el nombre CakePHP, en su versión 1.1. Después de un largo período de desarrollo de más de 2 años finalmente se anunció así la liberación de la versión final y estable de CakePHP 1.2 en la cual se notan considerables cambios, entre los que se destacan las mejoras en cuanto a las validaciones. Si bien la antigua versión daba la oportunidad de validar fechas, direcciones de correo, campos vacíos y números, la versión 1.2 supo integrar a esto, validaciones de teléfonos, alfanumérica, dirección de correo, máximo de caracteres y url. Sorprendentes son los cambios en el script bake.php, usado desde la consola, de la primera versión a la 1.2. Básicamente lo que hace el script bake.php es crear la estructura de archivos de los modelos, controladores y vistas (a esto normalmente se le conoce cómo baking). La nueva consola lleva esto al siguiente nivel permitiéndote no sólo hacer bake (auto-generar código) sino para automatizar diferentes tareas. Pasando desde la versión 1.2.0, hasta la 1.2.6 que es la última entrega estable que se conoce, no se han producido cambios radicales en cuanto a principios y estructura en el desarrollo Web, sino que ha disfrutado de necesarias mejoras, hechas o sugeridas por la gran comunidad mundial con que cuenta este software. Recientemente salió al público la versión 1.3, la cual se encuentra en estado de pruebas, esta versión es muy inestable y no cuenta todavía con el consentimiento de la comunidad para su uso. Aún en su última versión estable, el trabajo organizacional y de creación de clases, ya sean modelos, vistas o controladores, se hace a través de la

[2] Creador del framework CakePHP

consola (Cake Bake). En esta consola se realiza un gran por ciento del trabajo, en términos de desarrollo Web, llevado a cabo para crear aplicaciones, y aunque Cake Bake se ha nutrido de mejoras de una versión a otra, el trabajo sigue siendo de consola, en donde se ejecutan todas las tareas de creación, a través de comandos, lo cual resulta tedioso, engorroso y poco práctico para aquellos programadores que buscan rapidez en el trabajo.

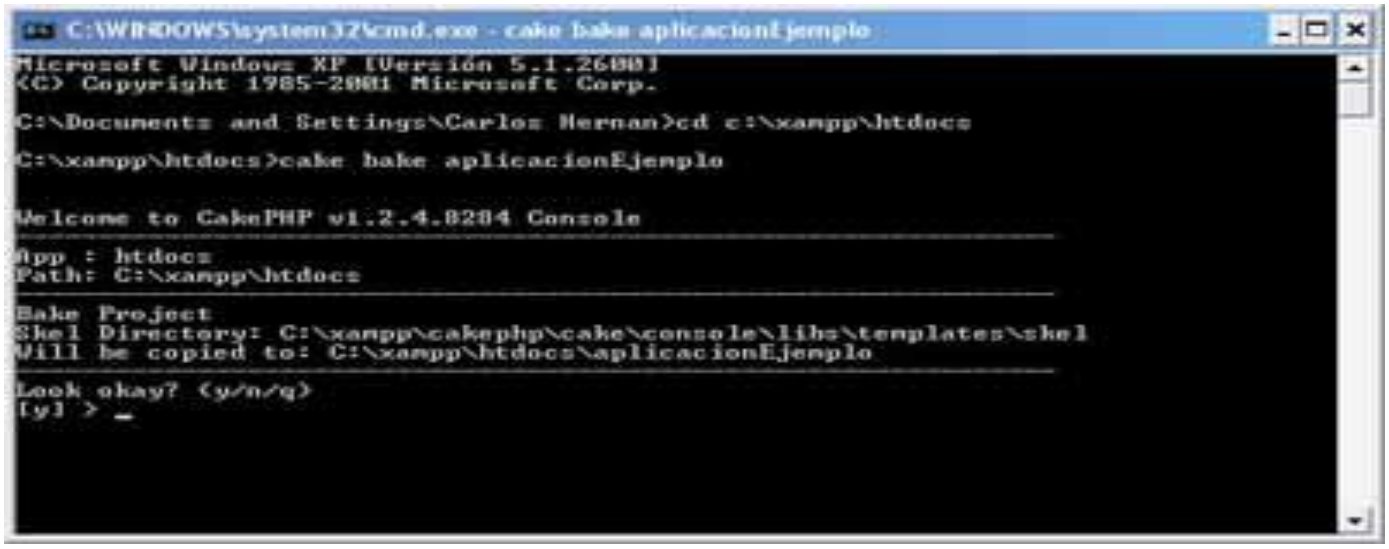
### 1.2.1 Como crean otros frameworks los modelos, las vistas y los controladores

Las vistas modelos y controladores son creados en CodeIgniter de forma manual, se crean archivos en directorios específicos para cada tipo de estructura, ya sean vistas modelos o controladores, dentro de estos archivos se crean las clases representativas de cada estructura siempre con la extensión php (1). De esta manera de construye una aplicación funcional en CodeIgniter, no siendo así en el tan usado y famoso Symfony, el cual hace este trabajo mediante comandos desde una consola, sin embargo una ventaja que aporta este software y es que en el no hay necesidad de crear modelos pues estos se generan automáticamente desde las tablas que existen en la base de datos del proyecto. (2)

## 1.3 Cake Bake

La consola Bake es otro esfuerzo para tener CakePHP corriendo rápido. La consola Bake puede crear cualquiera de los ingredientes básicos de CakePHP: modelos, vistas y controladores. Y no estamos hablando sólo de clases estructurales: Bake puede crear una aplicación completamente funcional en sólo unos minutos. De hecho, Bake es el paso natural que toman las aplicaciones una vez que han pasado por la etapa de scaffolding. [3] (3)

[3] Método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos framework del tipo MVC en el cuál el programador escribe una especificación que describe cómo debe ser usada la base de datos.




**Fig#1.1**: Aspecto visual de la consola cake bake

## 1.4 Patrón

Un patrón es una unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto. El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones. Pueden referirse a distintos niveles de abstracción, desde un proceso de desarrollo hasta la utilización eficiente de un lenguaje de programación. Un buen patrón debería: (4)

- Solucionar un problema
- Ser un concepto probado
- La solución no es obvia
- Describe participantes y relaciones entre ellos
- Tiene un componente humano alto: estética y utilidad

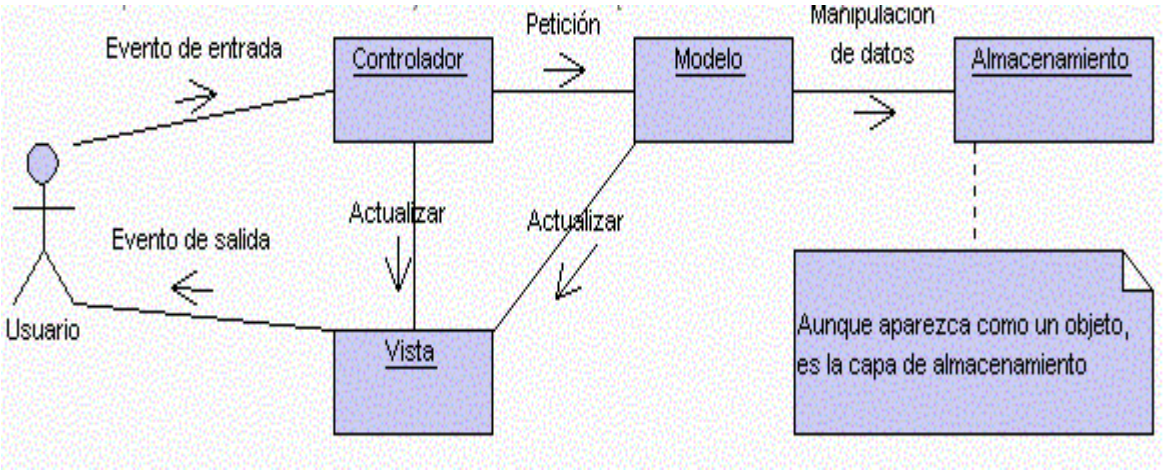
### 1.4.1 Modelo Vista Controlador (MVC).

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

Modelo: datos y reglas de negocio

- Vista: muestra la información del modelo al usuario
- Controlador: gestiona las entradas del usuario




**Fig#1.2**: Ejemplo de Modelo Vista Controlador

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Veamos cada componente:

El **modelo** es el responsable de:

Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El **controlador** es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar ()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las **vistas** son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes). (18)

### 1.5 PHP (acrónimo de "PHP: Hypertext Pre-processor")

PHP (Hypertext Pre-processor) es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de Aplicaciones Web muy robustas.

**Ventajas.**

- Es un lenguaje multiplataforma, funciona tanto para Unix (con Apache) como para Windows (con Microsoft Internet Information Server).
- Similar en sintaxis a C y a PERL.
- Es un lenguaje Orientado a Objetos a partir de la versión 4. Clases y herencia.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Evitando que el usuario tenga que separar las variables y sus valores.
- Se puede incrustar código PHP con etiquetas HTML.
- Excelente soporte de acceso a base de datos.
- La comprobación de que los parámetros son válidos se hace en el servidor y no en el cliente (como se hace con javascript) de forma que se puede chequear que no se reciban solicitudes adulteradas.
- Es software libre.
- Se puede obtener en la Web y su código está disponible bajo la licencia GPL.

**Desventajas:**

- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP. (5)

### 1.6 Técnicas y tecnologías del lado del cliente seleccionadas:

Para el desarrollo de dicha aplicación se decidió utilizar las siguientes herramientas: HTML, CSS y JavaScript; dicha elección fue realizada teniendo en cuenta las ventajas que estas tecnologías nos pueda proporcionar en el desarrollo de la aplicación. Por ejemplo: El HTML es el lenguaje estándar utilizado para

representar documentos en la WWW (World Wide Web), otra característica muy importante de este lenguaje es que es portable, es decir, se pueden visualizar las páginas con cualquier sistema operativo y, por supuesto también crearlas. Contamos además con CSS (Lenguaje de Hojas de estilo) que brinda entre sus principales ventajas, mejoría en la accesibilidad del documento, reduce la complejidad de su mantenimiento, permite visualizar el mismo documento en infinidad de dispositivos diferentes. Otra elección indispensable fue JavaScript, lenguaje especializado en las validaciones de datos y que no requiere de tiempo de compilación, posee características de interfaz, que son gestionados por el navegador y por el código HTML. (6)

## 1.7 Lenguaje del lado del servidor seleccionado (PHP)

Se ha arribado a la decisión de que el lenguaje a utilizar en la implementación de la aplicación propuesta será PHP; ya que es el lenguaje propio del Framework CakePHP, el cual será usado en la implementación del presente trabajo y es al mismo tiempo el objeto de estudio.

## *1.8* IDE (Entorno de Desarrollo Integrado) seleccionado.

Un entorno de desarrollo integrado (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. (7)

### 1.8.1 NetBeans

NetBeans IDE es un entorno de desarrollo visual para aplicaciones programadas mediante Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java, es uno de los lenguajes de programación más poderosos del momento.

Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.

Está escrito completamente en Java usando la plataforma NetBeans pero soporta el desarrollo de todos los tipos de aplicación Java (J2SE, Web, EJB y aplicaciones móviles), también puede servir para cualquier otro lenguaje de programación.

Aparte de la filosofía de distribución y desarrollo que respalda a NetBeans, el IDE ofrece a los desarrolladores numerosas ventajas, en la creación de nuevas aplicaciones multiplataforma. (8)

### 1.9 Gestor de Base Datos seleccionado (MySQL)

Para el desarrollo de la herramienta se realizó un detallado estudio de los Gestores de Base de Datos, el cual permitió que se pudiera realizar una elección certera. En dicho estudio se llegó a la conclusión de que el Gestor que se utilizará es MySQL, por las siguientes razones:

Es un sistema gestor de bases de datos “Open Source”.

Presenta múltiples motores de almacenamiento, permitiendo al usuario escoger el que sea más adecuado para cada tabla de la base de datos.

Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

- MySQL está escrito en una mezcla de C y C++.
- Además de las facilidades antes mencionadas se decidió escoger este gestor pues es muy usado en la universidad por ser muy rápido en aplicaciones de este tipo, además es compatible y muy utilizado junto al framework CakePHP y al lenguaje de programación seleccionado.(9)

### 1.10 Frameworks.

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto (10).

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al

código fuente, donde el desarrollador es obligado a crear código más legible y más fácil de mantener, lo cual permite la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (11)

### 1.10.1 Framework seleccionado (CakePHP).

Para el desarrollo de la herramienta se determinó utilizar el Framework CakePHP en su última versión estable, pues sus características de rapidez, estabilidad, una amplia comunidad mundial y el uso del Modelo Vista Controlador (MVC) proporcionan razones suficientes para su uso. CakePHP fue elegido para la implementación de la herramienta ya que es el objeto de investigación y desarrollo del presente trabajo, reduciendo así, tiempo valioso en aprendizaje e implementación de la herramienta a desarrollar. Pero tal vez lo más significativo de esta decisión sea que por sus características, la aplicación a desarrollar viajara con el Framework como un proyecto terminado que nos brinda la funcionalidad alterna a la consola cake bake. (13)

#### 1.10.1.1 Descripción de la disposición de archivos de Cake PHP

Cuando se desempaqueta Cake PHP en el servidor se encontraran 3 carpetas principales:
- App
- Cake
- Vendors

La carpeta Cake es el lugar para las bibliotecas base de CakePHP. La carpeta App es el lugar donde estarán las carpetas y archivos específicos de su aplicación. La separación entre la carpeta Cake y la carpeta App hace posible para usted tener muchas carpetas App compartiendo un solo conjunto de bibliotecas de Cake PHP. La Carpeta vendors se utiliza para guardar en ella bibliotecas de terceros.

## 1.11 Metodología, herramienta para la modelación del software.

Todo desarrollo de software se torna riesgoso y difícil de controlar, pero si no se utiliza una metodología, existe una mayor exposición a obtener clientes y desarrolladores insatisfechos con el resultado. Para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique el desarrollo del mismo, así como escoger una metodología adecuada para el desarrollo del software que sirva como guía para realizar de forma disciplinada y eficiente el producto deseado. Actualmente existen diversas metodologías de desarrollo de software, algunas de las más importantes y utilizadas son: Extreme Programing (XP), y Rational Unified Process (RUP).

### 1.11.1 Metodología de desarrollo seleccionada: Proceso Unificado de Modelado (RUP)

El Proceso Unificado es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos. Dicho proceso divide en 4 fases el desarrollo del software:

- ✓ **Inicio:** El Objetivo en esta etapa es determinar la visión del proyecto.
- ✓ **Elaboración:** En esta etapa el objetivo es determinar la arquitectura básica.
- ✓ **Construcción:** En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ **Transmisión:** El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de todas las iteraciones precedentes.

**Disciplina de Desarrollo:**

- ✓ Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- ✓ Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- ✓ Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.

- ✓ Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- ✓ Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado está presente.

**Disciplina de Soporte:**

- ✓ Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- ✓ Administrando el proyecto: Administrando horarios y recursos.
- ✓ Ambiente: Administrando el ambiente de desarrollo.
- ✓ Distribución: Hacer todo lo necesario para la salida del proyecto

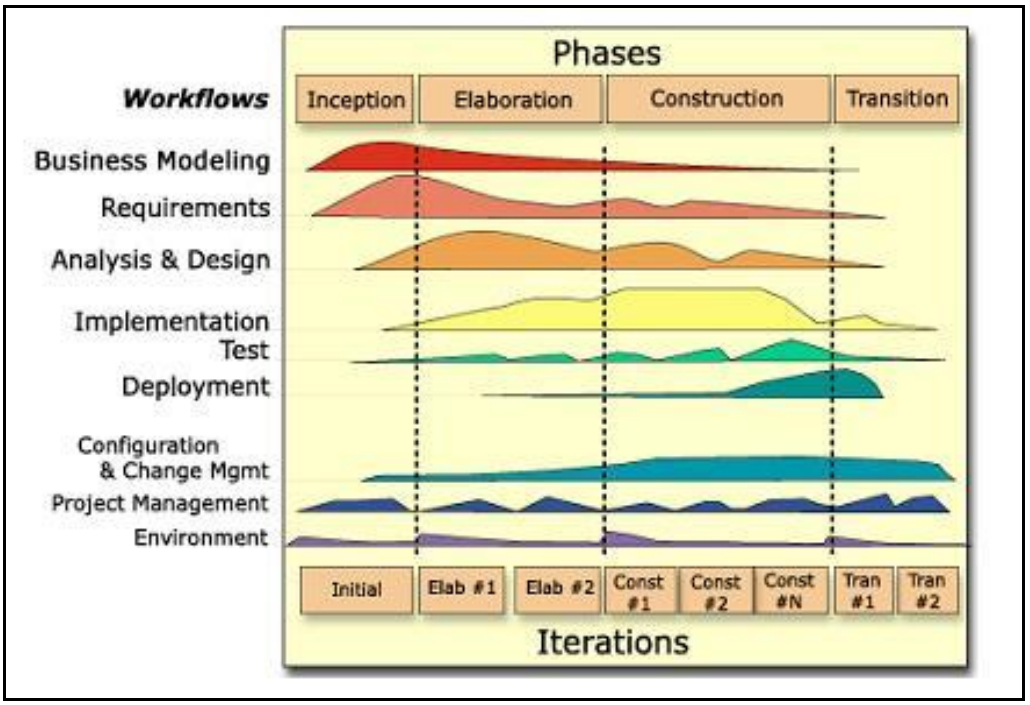



**Fig. #1.3 Fases e Iteraciones de la Metodología RUP**

Los elementos del RUP son:

- ✓ **Actividades**: Son los procesos que se llegan a determinar en cada iteración.
- ✓ **Trabajadores**: Vienen hacer las personas o entes involucrados en cada proceso.
- ✓ **Artefactos**: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

**Características del Proceso Unificado.**

**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. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Representan requerimientos funcionales. Los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso.

**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. La arquitectura se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por sí sola.

**Iterativo e Incremental:** RUP propone dividir el trabajo en partes más pequeñas o mini proyectos, donde cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos. (24)

### 1.13 Lenguaje de modelado UML

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo. UML es un lenguaje que permite la modelación de sistemas con tecnologías orientada a objetos (aprobado como estándar en noviembre de 1997). Es importante resaltar

que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, es el lenguaje en el que está descrito el modelo.

**Bloques de construcción del lenguaje**:

**Elementos**

Estructurales, comportamiento, agrupación, anotación.

**Relaciones**

Dependencia, asociación, generalización, realización.

**Diagramas**

Clases, objetos, casos de uso, secuencia, colaboración, estados, actividades, componentes, despliegue.

Los **objetivos** de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar**: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar**: UML permite especificar cuáles son las características de un sistema antes de su construcción. (27)

## 1.14 Herramientas CASE.

### 1.14.1 Visual Paradigm

Es una herramienta CASE que ofrece un entorno de creación de diagramas para UML, diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de ingeniería directa (versión profesional) e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; disponibilidad de múltiples versiones, para cada necesidad; disponibilidad de integrarse en los

principales IDEs; disponibilidad en múltiples plataformas, y muy útil para la generación de código fuente en PHP, también con el Paradigm se generan script de las tablas de salidas para las clases persistentes. (22)

**Ventajas**

- ✓ Navegación intuitiva entre el modelo visual y el código.
- ✓ Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
- ✓ Sincronización entre el código fuente y el modelo en tiempo real o bajo demanda.
- ✓ Entorno visual de modelado superior.
- ✓ Soporte para toda la notación UML
- ✓
- ✓ Sofisticados y automáticos diagramas de capas.
- ✓ Análisis de textos.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas.

## 1.15 Conclusiones Parciales

En este capítulo se realizó un análisis de los elementos más importantes que giran entorno al trabajo a realizar, realizando un estudio de sus principales potencialidades así como los aspectos negativos que las opacan.

A partir del estudio realizado se han tomado varias decisiones en cuanto a las herramientas a utilizar para el desarrollo un software capaz de resolver el problema presentado.

# 2 CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

## 2.1 Introducción

Antes de comenzar a desarrollar un sistema es necesario comprender el entorno de trabajo, bajo el estudio de los procesos que en él se desarrollan. En este capítulo se hace un análisis de las distintas técnicas utilizadas a la hora de llevar a cabo la ingeniería de requisitos, se analiza de forma objetiva el entorno del sistema, específicamente la administración y configuración de la herramienta. Se realiza el modelo de dominio y todo lo que este conlleva, se analizan los requisitos funcionales para dar paso a la solución del problema y la especificación de los mismos, los requisitos no funcionales que son de suma importancia también pues son requerimientos para el software sea más atractivo, y de cierta seguridad al cliente. Además se hace una pequeña descripción de los patrones utilizados para los casos de uso, se describen todos los Casos de Usos del Sistema, analizando detalladamente cada escenario posible, y se habla de forma resumida de los aportes y beneficios que traerá el Análisis y Diseño de la Herramienta alternativa a cake bake usada para la configuración y creación de aplicaciones en CakePHP.

## 2.2 Técnicas utilizadas en las actividades de la Ingeniería de Requisitos.

Existen varias técnicas propuestas para la ingeniería de requerimientos de las cuales en el presente trabajo de diploma solo se abarcan las que se utilizaron para capturar los requisitos del sistema. Es importante resaltar que estas técnicas pueden ser aplicables a las distintas fases del proceso de la IR (Ingeniería de Requisitos) y es necesario tener en cuenta las características propias del proyecto en desarrollo para aprovechar al máximo la utilidad de las mismas. (28)

### 2.2.1 Entrevistas

Las entrevistas se emplean para reunir información proveniente de personas o de grupos. Durante la entrevista, el analista conversa con el encuestado para obtener información deseada. Las entrevistas se emplean para recolectar información en forma verbal, a través de preguntas elaboradas por el analista, estas preguntas se les realiza a quienes se encuentren afectados por la aplicación, a usuarios con gran nivel de conocimiento del sistema o a personas que pueden proporcionar datos. El éxito de esta técnica, depende de la habilidad del entrevistador y de su preparación para la misma.

### 2.2.2 Sistemas existentes

La técnica de Sistemas existentes consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el sistema a ser construido. Por un lado, se pueden analizar las interfaces de usuario, observando el tipo de información que se maneja y cómo es manejada. Esto puede ser útil para descubrir información importante a tener en cuenta, información que tal vez el cliente/usuario haya fallado en comunicar. Es recomendable que luego de haber analizado el sistema, este sea mostrado al cliente/usuario, ya que por su experiencia puede sugerir importantes ideas nuevas

### 2.2.3 Prototipos

Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Entonces, para validar los requerimientos hallados, se construyen prototipos. Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva.

## 2.3 Patrón utilizado para los requisitos.

**CRUD (Creating, Reading, Updating, Deleting)**

### 2.3.1 Patron CRUD (Creating, Reading, Updating, Deleting)

**Completo**

Este patrón consta de un caso de uso, llamado Información CRUD o gestionar información, que modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico,

tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

**Parcial**

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es mas significativa, larga o más compleja que las otras.

En los requerimientos identificados se pone de manifiesto el patrón CRUD, que aunque está dirigido a casos de uso están presentes tanto el completo como el parcial en los requisitos de software definidos. (17)

## 2.4 Especificación de requisitos de software.

Un requerimiento es una representación documentada de una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, que puede ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Permite definir el ámbito del sistema, define una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario. Establece y mantiene un acuerdo entre clientes y otros involucrados sobre lo que el sistema debería hacer. (19)

### 2.4.1 Requerimientos Funcionales.

Los requerimientos funcionales no son más que las condiciones o capacidades que el sistema debe cumplir. Los requerimientos funcionales que rigen el Análisis y Diseño del presente trabajo, son los siguientes:

**R1**-Autenticar Usuario.
**R2**-Gestionar usuario.
**R 2.1** Crear usuario.
**R 2.2** Eliminar usuario.
**R 2.3** Mostrar datos usuario.

**R 2.4** Editar datos usuario.

**R3**-Gestionar Configuraciones.
**R 3.1** Crear configuración.
**R 3.2** Eliminar configuración.
**R 3.3** Mostrar configuración.
**R 3.4** Editar configuración.

**R4**-Gestionar Proyectos.
**R 4.1** Crear proyecto.
**R 4.2** Eliminar proyectos.
**R 4.3** Mostrar proyectos.
**R 4.4** Editar proyectos.

**R5**-Gestionar Modelos.
**R 5.1** Crear modelos.
**R 5.1.1** Validar campos.
**R 5.1.2** Gestionar relaciones
**R 5.1.2.1** Insertar relaciones
**R 5.1.2.2** Editar relaciones
**R 5.1.2.2** Eliminar relaciones
**R 5.2** Eliminar proyectos.
**R 5.3** Mostrar proyectos.

**R6**-Gestionar Controladores.
**R 6.1** Crear Controladores
**R 6.1.1** Añadir funciones básicas
**R 6.2** Eliminar Controladores
**R 6.3** Mostrar Controladores.

**R 6.4** Editar Controladores.

**R7**-Gestionar Vistas.
**R 7.1** Crear Vistas.
**R 7.2** Eliminar Vistas.
**R 7.3** Mostrar Vistas.
**R 7.4** Editar Vistas.

### 2.4.2 Requerimientos No Funcionales.

Los requerimientos no funcionales especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento o facilidad de mantenimiento.

**Usabilidad:**

En la aplicación se garantizara un acceso fácil y rápido a los programadores. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y de un ambiente web en sentido general.

**Apariencia o interfaz externa**

La aplicación contara con un color que denote seriedad debido a su contenido, además deberá ser atractivo a la vista del usuario, seleccionando como color predominante el blanco.

**Rendimiento**

- El sistema debe ser lo más eficiente posible para poder lograr un tiempo de respuesta adecuado.
- La velocidad de procesamiento de la información debe ser rápida, acorde con las necesidades del usuario.
- Aplicación de las diferentes técnicas de elaboración en el sistema para facilitar el rápido acceso a los datos.

**Soporte**

El sistema deberá permitir posteriores modificaciones y actualizaciones a fin de alcanzar mayor funcionalidad o dado que cambien algunos elementos del negocio.

**Portabilidad**

El sistema operará sobre las plataformas basadas en Linux y Windows.

**Seguridad**

- La información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación.
- La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes por las personas autorizadas.
- A los usuarios autorizados se les garantizará el acceso al sistema y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los mismos para realizar las transferencias necesarias en un momento dado.
- Se deben crear usuarios que tendrán asignados permisos de acción sobre cada información manejada por el sistema para lo cual se requiere la autenticación del usuario.

**Requerimientos de Hardware**

**Para el cliente:**

- Se requiere que las computadoras estén conectadas a la red.

**Para el servidor:**

Las condiciones mínimas de hardware son: que las computadoras tengan procesador Intel Celeron a 2.80 GHz o superior, disco duro de 5 GB y 128 MB de RAM mínimo.

**Requerimientos de Software**

La máquina servidor deberá disponer del servidor web apache versión 2.0. Además se debe contar con el sistema gestor de base de datos MySQL. El lenguaje de programación será php en su versión 4 o más avanzada. Las máquinas clientes podrán contar cualquier versión del sistema operativo Windows o con cualquier distribución de Linux. Las máquinas clientes para acceder al sistema deben hacerlo a través del navegador moxila o Internet Explorer. (23)

**Requerimientos en el diseño y la implementación**

Para organizar el análisis y el diseño del sistema se utilizará:

- La metodología RUP.
- UML como lenguaje de modelado.
- Visual Paradigm como herramienta CASE.
- PHP será el lenguaje de programación a ser usado para la implementación.
- Los nombres de los métodos y de las clases deben ser lo más sencillo posible para un mejor entendimiento entre el equipo de desarrolladores.

## 2.5 Modelo de dominio.

A partir de una evaluación del estado del negocio, cuyo objetivo para la administración informática es la gestión de información y configuración del sistema, se determinó realizar lo que se conoce como modelo del dominio. En este se representan los principales conceptos y las relaciones entre ellos, permitiendo emplear un vocabulario común que posibilita tanto a los desarrolladores como a los clientes entender el contexto y lograr una solución que cumpla con las expectativas de estos últimos.

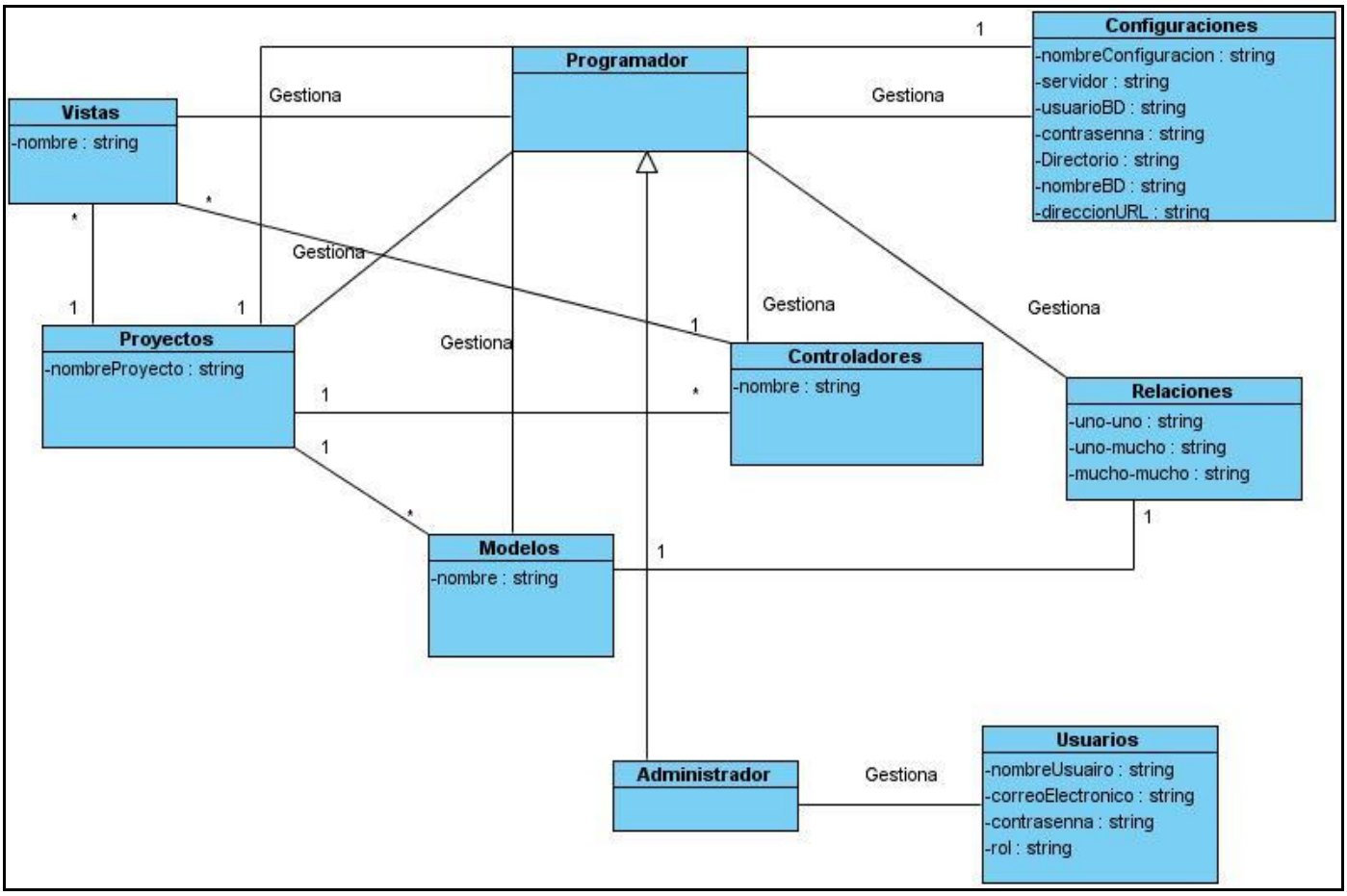



**Figura2.1: Modelo de Dominio**

**Administrador:** Es el encargado de administrar y configurar todo lo referente a la Gestión de proyectos y de usuarios.

**Programador:** Es el encargado de administrar y configurar todo lo referente a la Gestión de proyectos.

**Configuracion:** Tiene la información necesaria para que los proyectos se conecten a la base de datos.

**Proyecto:** En él se crea toda la estructura de las aplicaciones a construir.

**Modelo:** Archivo que hace referencia a una tabla en la base de datos.

**Vista:** Tiene la información que se le presenta al usuario en cada proyecto

**Controlador:** En él se manejan todas las acciones de los proyectos.

**Usuario:** Tabla en la base de datos en la que se guarda la información necesaria para cada usuario.
**Relación:** Tiene la información que se necesita para relacionar los modelos.

## 2.6 Modelamiento del Sistema.

### 2.6.1 Descripción de la solución propuesta.

La herramienta de configuración para el framework CakePHP que se pretende implementar será una alternativa visual a la herramienta propia del framework llamada cake bake. La herramienta a desarrollar tendrá el nombre VisualBake, asumiendo en su nombre, características de su predecesor cake bake y tomando las cualidades web de la herramienta a desarrollar. Visual Bake fue diseñado con la intención de crear una herramienta con características visuales y amenas al programador, utilizara una base de datos en el gestor MySQL, gestor propio para el framework a mejorar, aunque no el único.

VisualBake será una aplicación Web al que solo pueden acceder y tienen permiso de ejecución, los usuarios previamente autenticados que tengan el rol de administradores o programadores de la misma. Tiene como propósito garantizar la configuración de los proyectos generados en CakePHP, su propósito natural es darle un ámbito visual a la herramienta cake bake.

### 2.7 Actores del sistema.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. A continuación se muestra el actor que interviene en el sistema.

| Actor | Descripción |
|---|---|
| Administrador | Interactúa con todas las funcionalidades de la aplicación, crea, elimina y modifica vistas, controladores y usuarios. Selecciona el gestor de bases de datos a usar, también selecciona los modelos a utilizar, gestiona la configuración que tendrán los proyectos creados. |
| Programador | Interactúa con las funciones básicas del programador, crea elimina y modifica modelos, vistas y controladores, gestiona la configuración que tendrán los proyectos creados por este. |

**Tabla #1:** Actores del sistema.

### 2.8 Diagramas de Casos de Uso del Sistema.

Es un modelo de las funciones deseadas para el sistema y su entorno, y sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis, diseño y prueba.

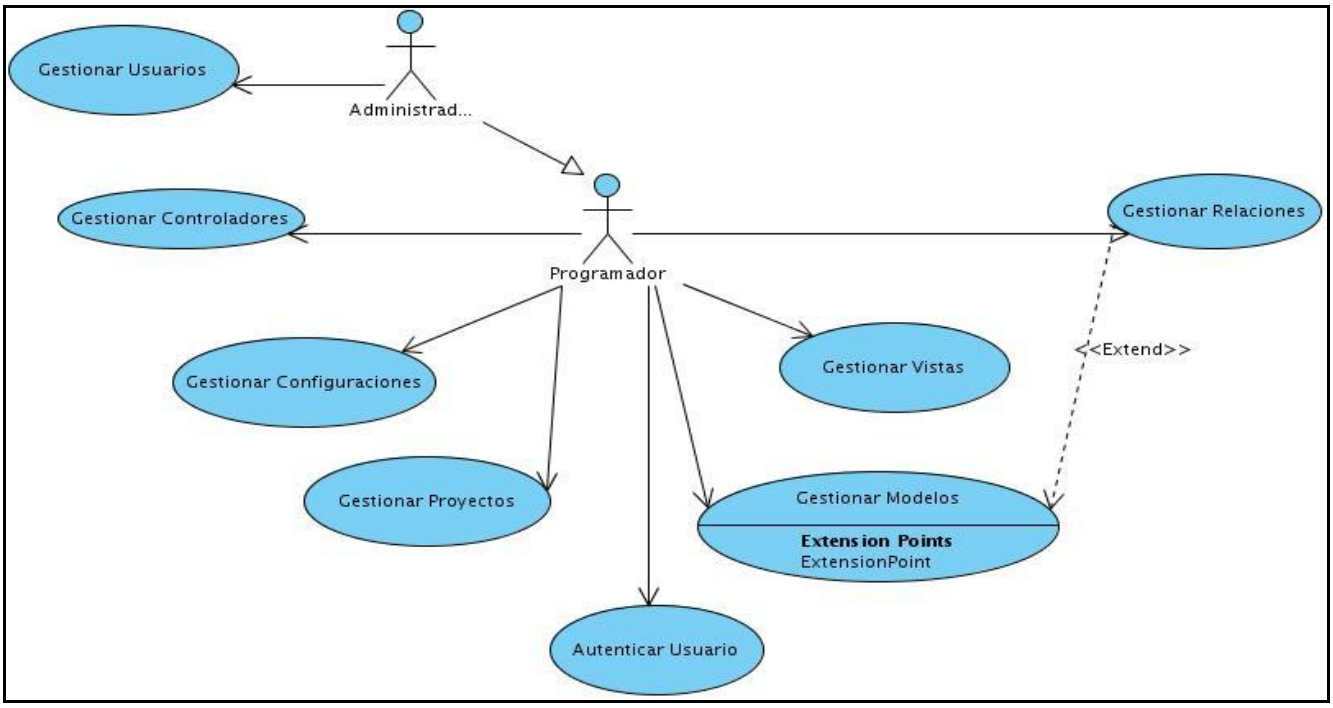



**Figura 2.2-Diagrama de Casos de Uso del Sistema**

## 2.9 Desglose de los Casos de Uso Gestionar.

### Gestionar Configuración

1. Mostrar configuraciones
2. Crear configuraciones.

3. Eliminar configuraciones.
4. Modificar configuraciones.

**Gestionar Proyectos**

1. Mostrar Proyectos.
2. Crear Proyectos.
3. Modificar Proyectos.
4. Eliminar Proyectos.

**Gestionar modelos**

1. Mostrar modelos.
2. Crear modelos.
3. Validar campos.
4. Eliminar modelos.

**Gestionar relaciones**

1. Mostrar relaciones
2. Insertar relaciones.
3. Modificar relaciones.
4. Eliminar relaciones.

**Gestionar controladores**

1. Mostrar controladores.
2. Crear controladores.
3. Añadir funciones básicas.
4. Modificar controladores.
5. Eliminar controladores.

**Gestionar vistas**

1. Mostrar vistas.
2. Crear vistas.
3. Eliminar vistas.
4. Modificar vistas.

**Gestionar Usuario**

1. Mostrar usuarios.
2. Crear usuarios.
3. Eliminar usuarios.
4. Modificar usuarios.

## 2.10 Expansión de los Casos de Uso del Sistema.

Los casos de uso expandidos describen detalladamente la secuencia de pasos que realizan los actores en su interacción con el sistema para completar cada proceso.

(Ver anexo #1)

## 2.11 Aportes y Beneficios.

El sistema propuesto permitirá gestionar la creación de proyectos en CakePHP de una manera muy flexible, amigable y fácil de usar, ya que brindará la posibilidad de configurar de manera manual, y de una manera centralizada los distintos objetos configurables de un proyecto web. Será una herramienta muy singular ya que existen pocas con las características que esta presenta. Su principal aporte es eliminar la consola propia de CakePHP con la cual se gestionan todas las estructuras de los proyectos web del dicho framework.

## 2.12 Conclusiones Parciales.

En este capítulo se hizo un profundo estudio de las características que tendrá el sistema a implementar, nos apropiamos de elementos concisos al analizar el entorno del problema desde el punto de vista del usuario que manejará la futura herramienta de configuración para CakePHP, que nos acerca más a la propuesta de implementación, y crea los cimientos para el análisis y diseño. Se valoraron los requisitos funcionales y no funcionales para de esta forma ir definiendo y pormenorizando los detalles que debe tener el mismo. Se realizó además una propuesta de interfaz de usuario no funcional, y se llegó a la conclusión de que la misma cumplía los requisitos para ser la interfaz funcional de la aplicación con algunas pequeñas modificaciones. Y se llegaron a conclusiones importantes sobre los aportes y beneficios que este trabajo y la consiguiente implementación de la aplicación.

# CAPITULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

## 3.1 Introducción

En el capítulo que inicia se procederá a realizar los restantes pasos para que todo quede listo para la implementación del sistema. Inicialmente se realizará el modelo de análisis, así como las clases que el incluye y los diagramas de colaboración, lo que servirá de entrada para el diseño. También se describen los patrones de diseño usados, ya que los mismos permiten solucionar problemas de reutilización de código, organización y mayor claridad para los programadores cuando se dispongan a implementar. Luego de esto se pasará a realizar las clases del diseño utilizando estereotipos Web, así como la descripción de las clases más significativas, y por último en el presente trabajo se tratarán de aplicar métricas para la evaluación del diseño, lo cual de resultar efectivas garantizarán una correcta implementación.

## 3.2 Modelo de Análisis.

El Análisis es el flujo de trabajo donde se refinan y estructuran los requisitos obtenidos con anterioridad, con el objetivo de facilitar la compresión, preparación y modificación de los mismos. Un modelo de análisis es aquel que estructura los requisitos de un modo que facilita su comprensión y puede considerarse además como una primera aproximación al Modelo del Diseño.

Una realización de casos de uso proporciona, por tanto, una traza directa hacia un caso de uso concreto del modelo de casos de uso, además de ayudar a comprender los requisitos del software y no cómo se implementará la solución; logrando una profundización más precisa de los requerimientos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura. (20)

### 3.2.1 Diagramas de Clase del Análisis.

El objetivo principal del modelo de análisis es obtener una comprensión profunda y una descripción detallada de los requisitos de software que ayuden a estructurar la vista interna del sistema. Para ello se

definen las realizaciones de los casos de uso, donde se describe cómo se llevan a cabo cada uno de ellos, a través de las clases del análisis y de la interacción entre sus objetos Es descrito en el lenguaje de los desarrolladores pero sin tener en cuenta el lenguaje de programación ni la plataforma, pues no está presente en sus propósitos precisar cómo se implementará la solución. Sirve como una primera aproximación al diseño constituyendo sus artefactos la entrada principal de este último.

**Clase Interfaz:** Se utilizan para modelar la interacción entre el sistema y sus actores, cada interfaz debe asociarse con al menos un actor y viceversa.

**Clase Controladora:** Representan coordinación, secuencia, transacciones y control de otros objetos.

**Clase Entidad:** Se utilizan para modelar la información que posee una larga vida y que es a menudo persistente.

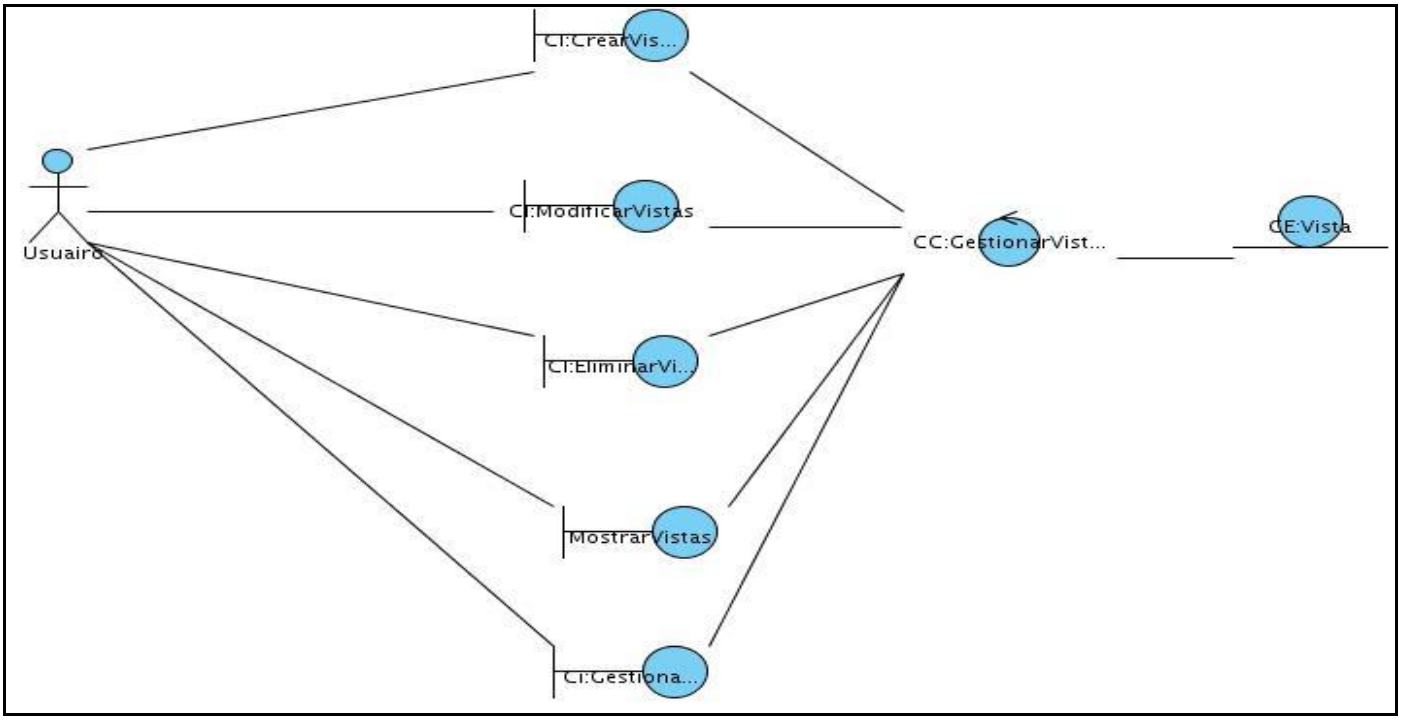



**Figura 3.1-Diagrama de clases del Análisis del CUS Gestionar Vistas**

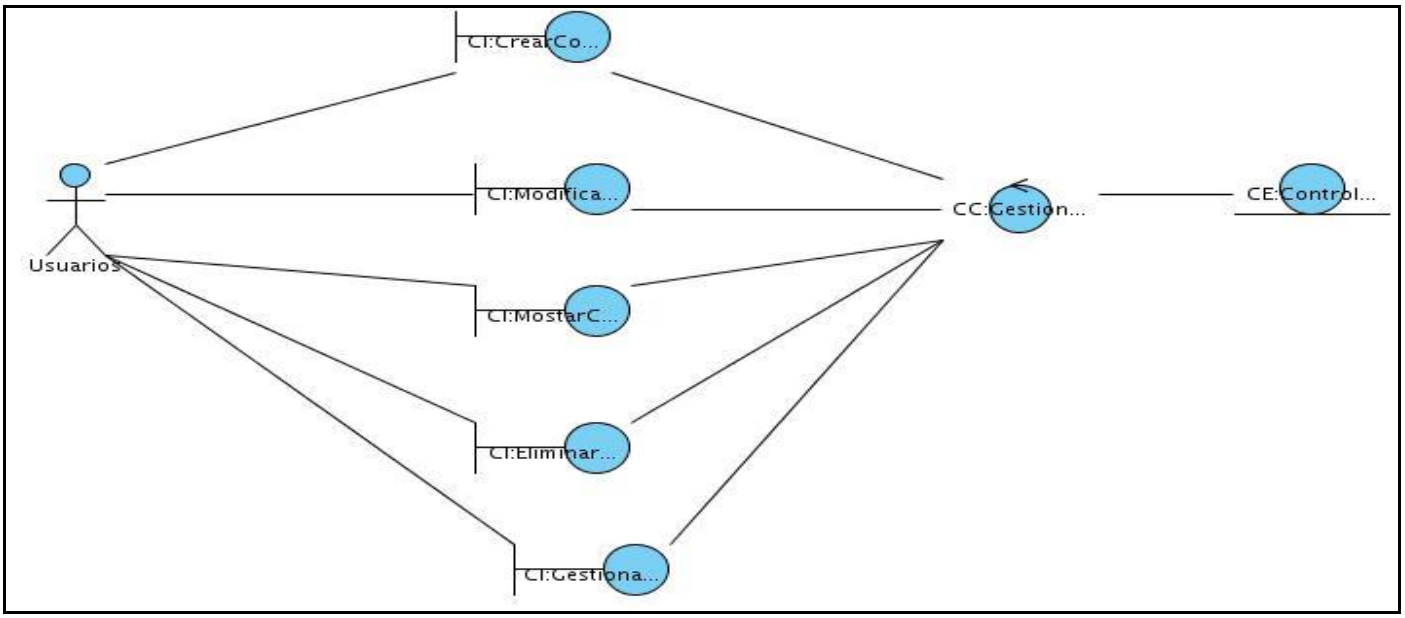



**Figura 3.2-Diagrama de clases del Análisis del CUS Gestionar Controladores**

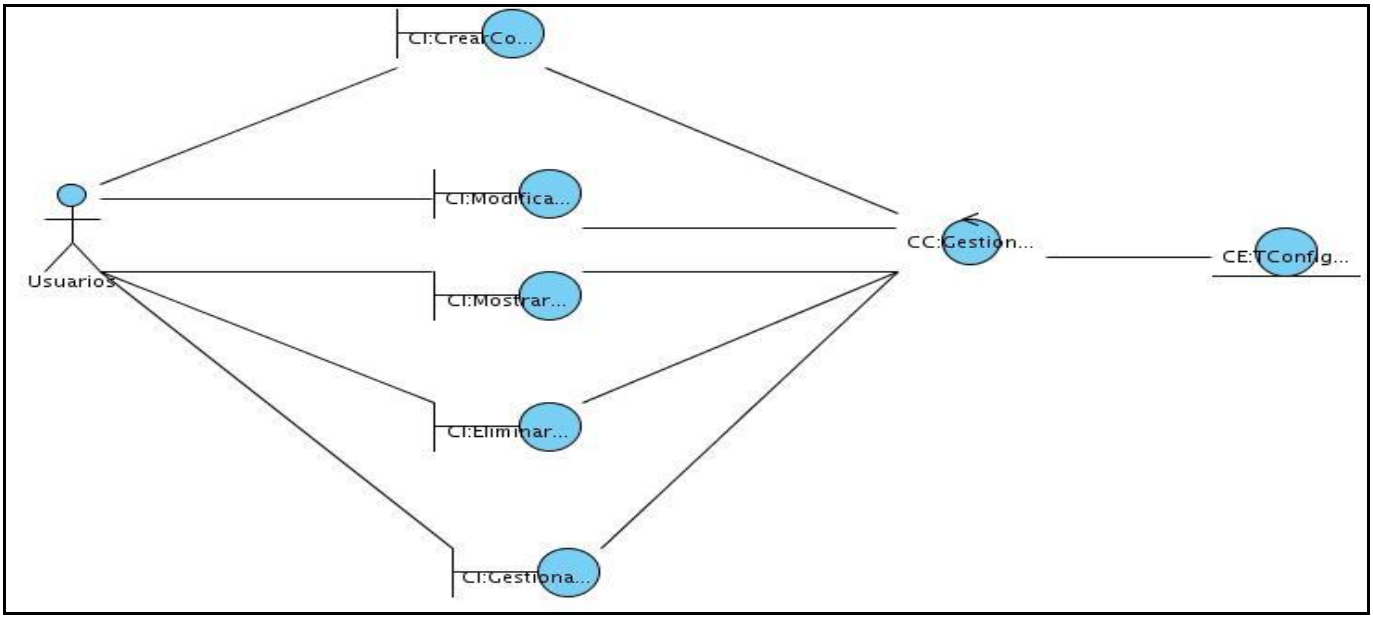



**Figura 3.3-Diagrama de clases del Análisis del CUS Gestionar Configuraciones**

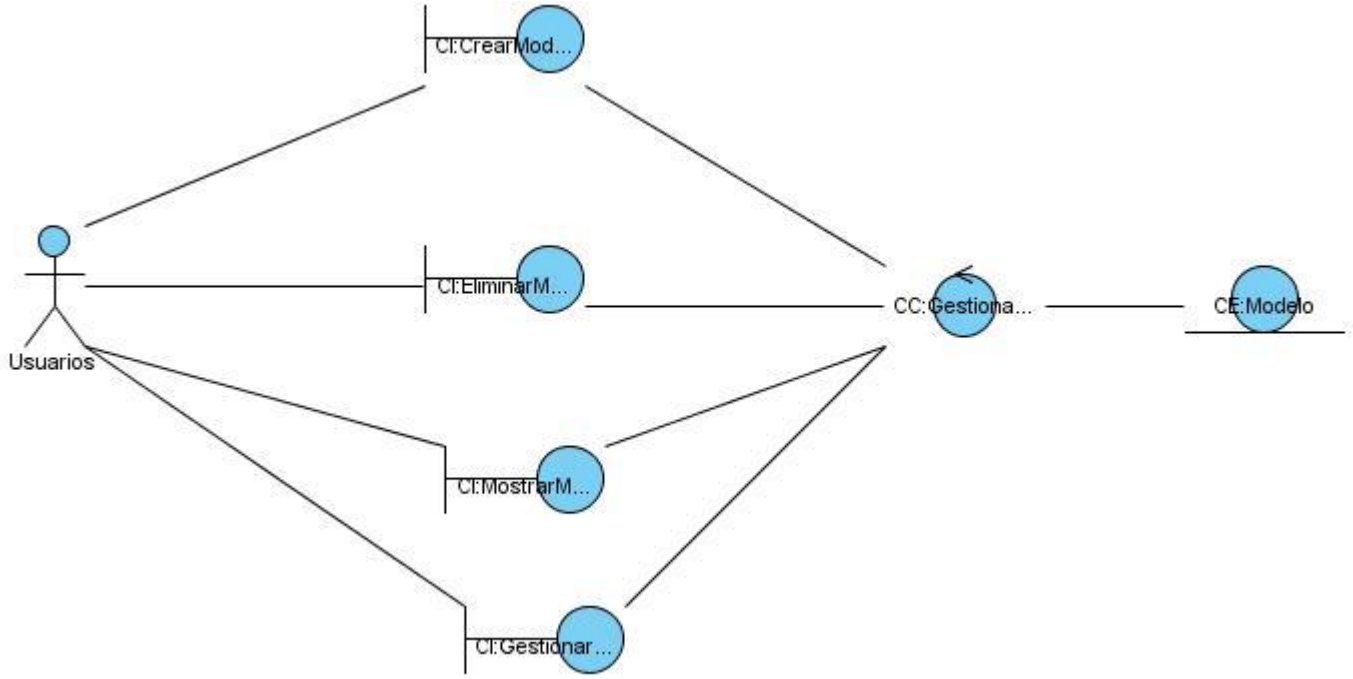



**Figura 3.4-Diagrama de clases del Análisis del CUS Gestionar Modelos**

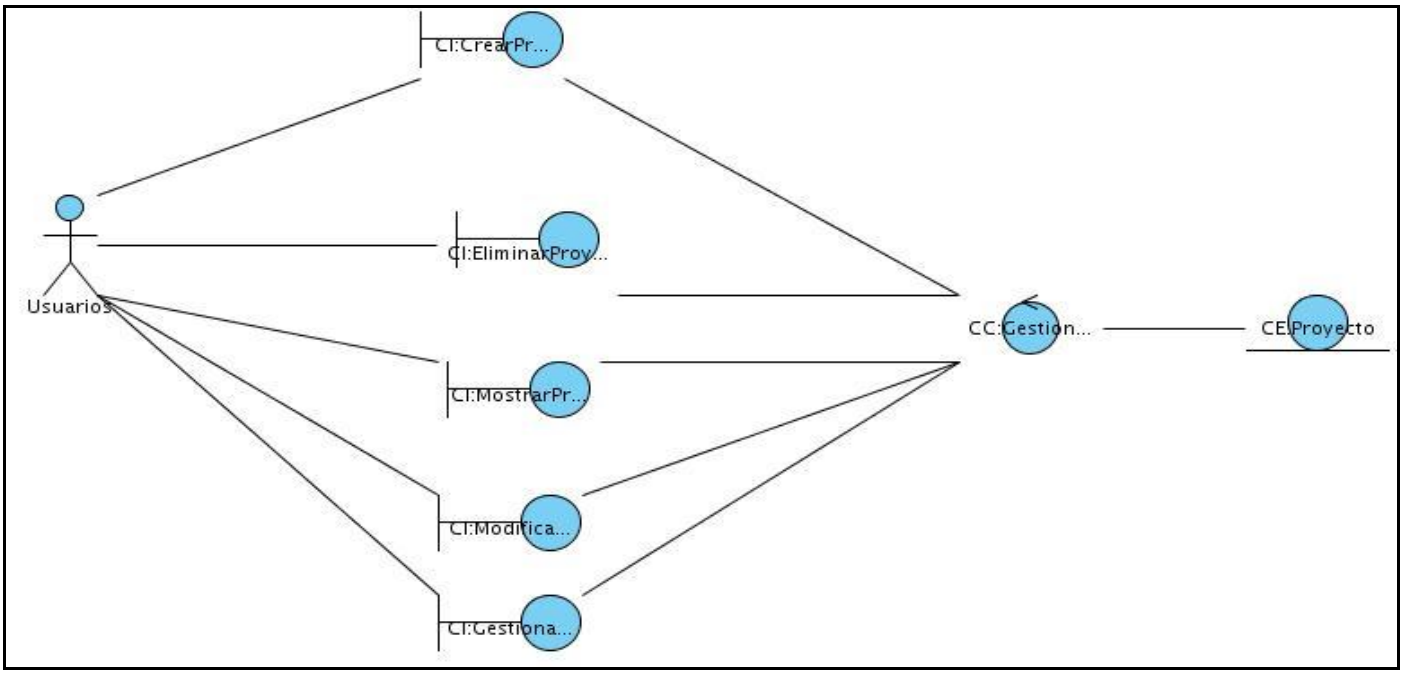



**Figura 3.5-Diagrama de clases del Análisis del CUS Gestionar Proyectos**

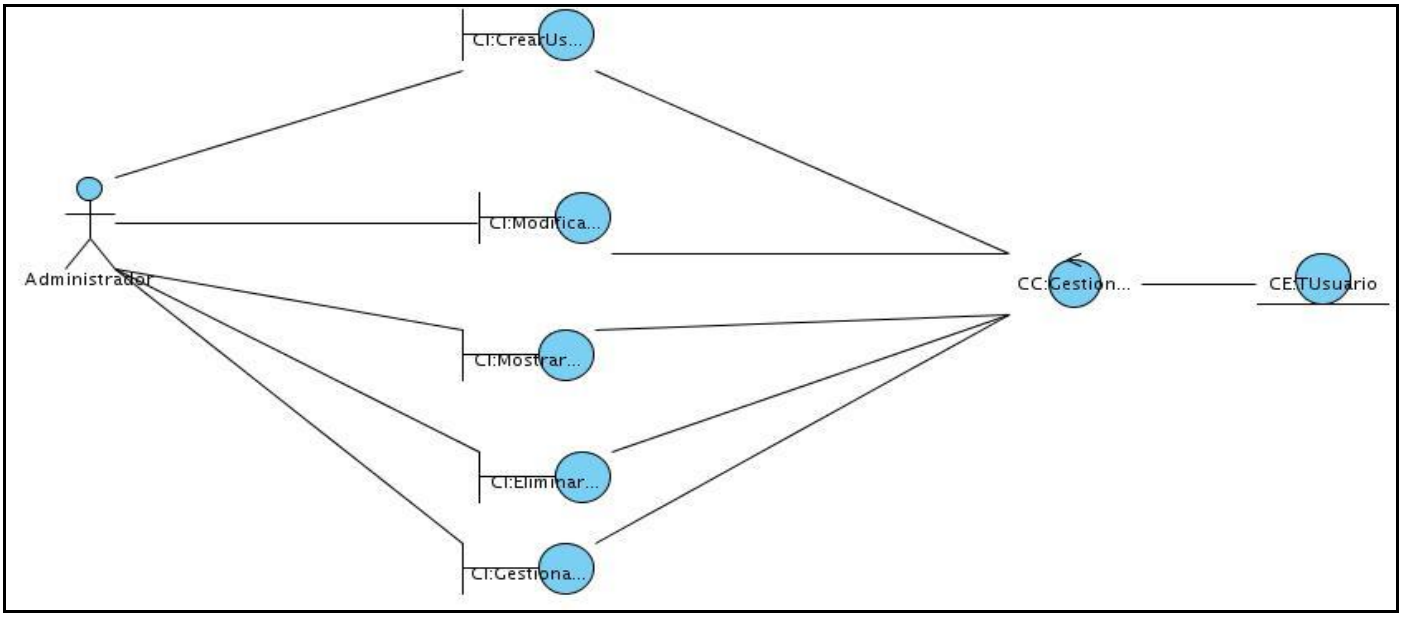



**Figura 3.6-Diagrama de clases del Análisis del CUS Gestionar Usuarios**

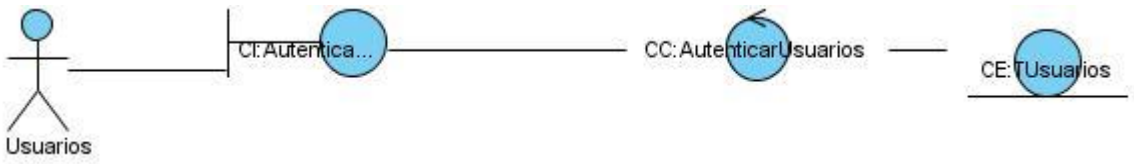



**Figura 3.7-Diagrama de clases del Análisis del CUS Autenticar Usuarios**

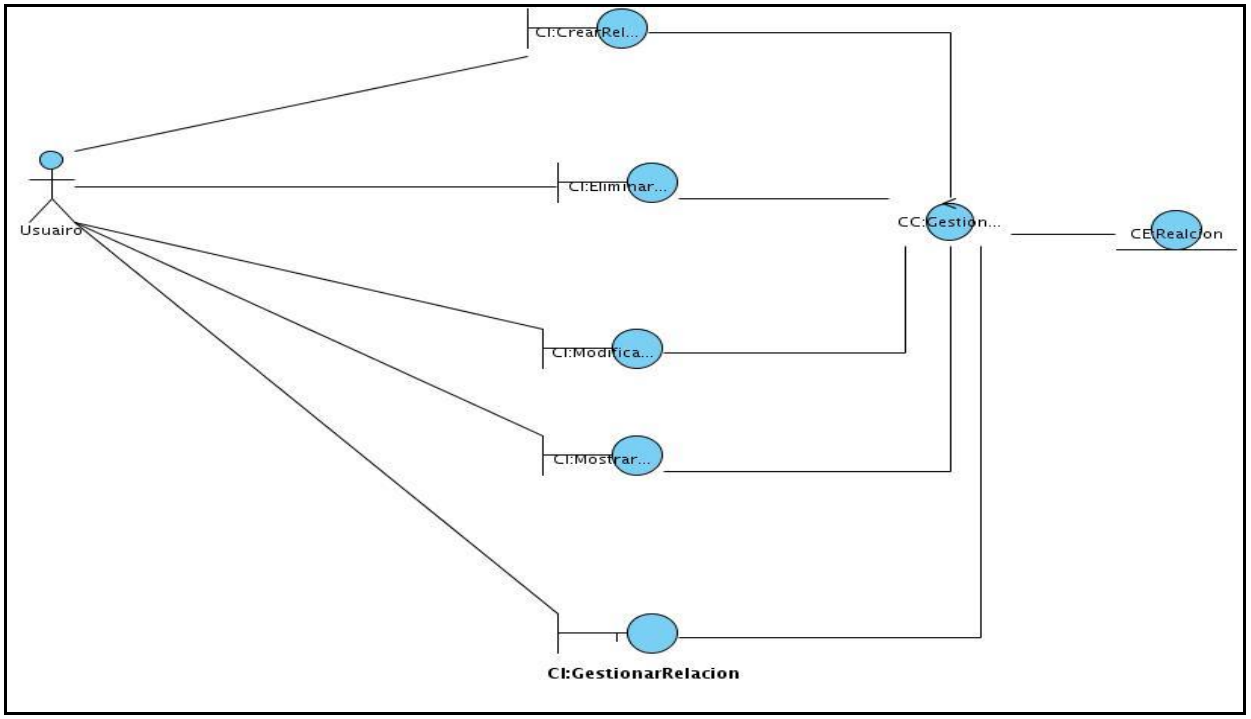



**Figura 3.8-Diagrama de clases del Análisis del CUS Gestionar Relaciones**

### 3.2.2 Diagramas de Colaboración.

Los diagramas de colaboración son usados para modelar la dinámica del sistema a través de los objetos de las diferentes clases del análisis, sus relaciones y los mensajes que se puedan enviar entre ellos. Un diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes. Los diagramas de colaboración de los Caso de Uso del Sistema se pueden encontrar en el Anexo#2

## 3.3 Modelo de diseño.

Dada la comprensión detallada de los requisitos de software que proporciona el Modelo de Análisis y tratando de conservar todo lo que se pueda la estructura del sistema que propone, en el Modelo de Diseño se modela la aplicación de forma tal que satisfaga todos los requisitos incluyendo los no funcionales. Es muy importante pues permite que se reduzcan los riesgos de cometer errores en la implementación, de construir un sistema inestable, sensible a los cambios o difícil de comprobar si es realmente funcional. Si se evita lo anteriormente expuesto con un buen diseño, se aumenta considerablemente la calidad del software, el diseño es el lugar donde se fomentará la calidad del software.

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, que proporciona una comprensión detallada de los requisitos. El diseño del sistema se realizará teniendo en cuenta el patrón de arquitectura Modelo Vista Controlador (MVC). (29)

## 3.4 Patrones de Diseño.

¿Que es un patrón de diseño?

*"Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan". Son:*

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.
- La forma de un diagrama de objeto o de un modelo de objeto. (21)

### 3.4.1 Patrones GOF

Estos patrones se dividen en tres grupos (Creación, Estructurales, Comportamiento).

**Singleton:** Este patrón garantiza que solamente se cree una instancia de la clase y provee un punto de acceso global a dicha instancia y pertenece al grupo de los patrones GOF de Creación.

**Facade:** Este patrón simplifica los accesos a un conjunto de objetos relacionados proporcionando un objeto de comunicación.

**Factory:** Define una interfaz para crear un objeto dejando a las subclases decidir el tipo específico al que pertenecen.

### 3.4.2 Patrones de Acceso a Datos

**Active Record:** Este patrón representa de forma Orientada a Objetos los datos de una Base de Datos Relacional, definiendo interfaces sencillas para acceder y manipular esos datos. Es un enfoque al

problema de acceder a los datos de una base de datos. Una fila en la tabla de la base de datos (o vista) se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. Cuando se crea uno de estos objetos, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos. CakePHP usa este patrón para manejar la base de datos.

### 3.4.3 Patrones de Asignación de Responsabilidades (GRASP).

**Bajo Acoplamiento:** Este patrón brinda como solución asignar responsabilidades de manera que las clases no dependan fuertemente de otras. Ofreciendo como beneficio que son fáciles de entender por separadas, fáciles de reutilizar y no se afectan por cambios de otros componentes. Dicho patrón se tiene en cuenta debido a la importancia de realizar un diseño de clases independientes que soporten los cambios.

**Alta cohesión:** El siguiente patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización.

## 3.5 Diagramas de Clases del Diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contienen la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.

- Información sobre los tipos de atributos.
- Navegabilidad.
- Dependencias.

Como parte del flujo de trabajo Diseño se generaron también los diagramas de secuencia para cada caso de uso, los mismos expresan las relaciones entre las clases y dan al programador una guía para realizar la implementación a partir de los mensajes y llamadas a funciones que se intercambian entre las clases

Los diagramas de secuencia del diseño se pueden encontrar en el anexo #3.

### 3.5.1 Diagrama de Clases con Estereotipos Web.

La aplicación a desarrollar es Web, es decir, que tiene como elemento significativo de su arquitectura un navegador y un protocolo de comunicación HTTP. Para el modelado de este tipo de aplicaciones se aprovecha una de las características que le da flexibilidad a la notación de UML. La misma consiste en un conjunto de mecanismos de extensión. El uso de estereotipos permite enriquecer el significado de los elementos clásicos de su notación. Las clases del diseño involucradas con sus respectivos estereotipos se muestran a continuación:

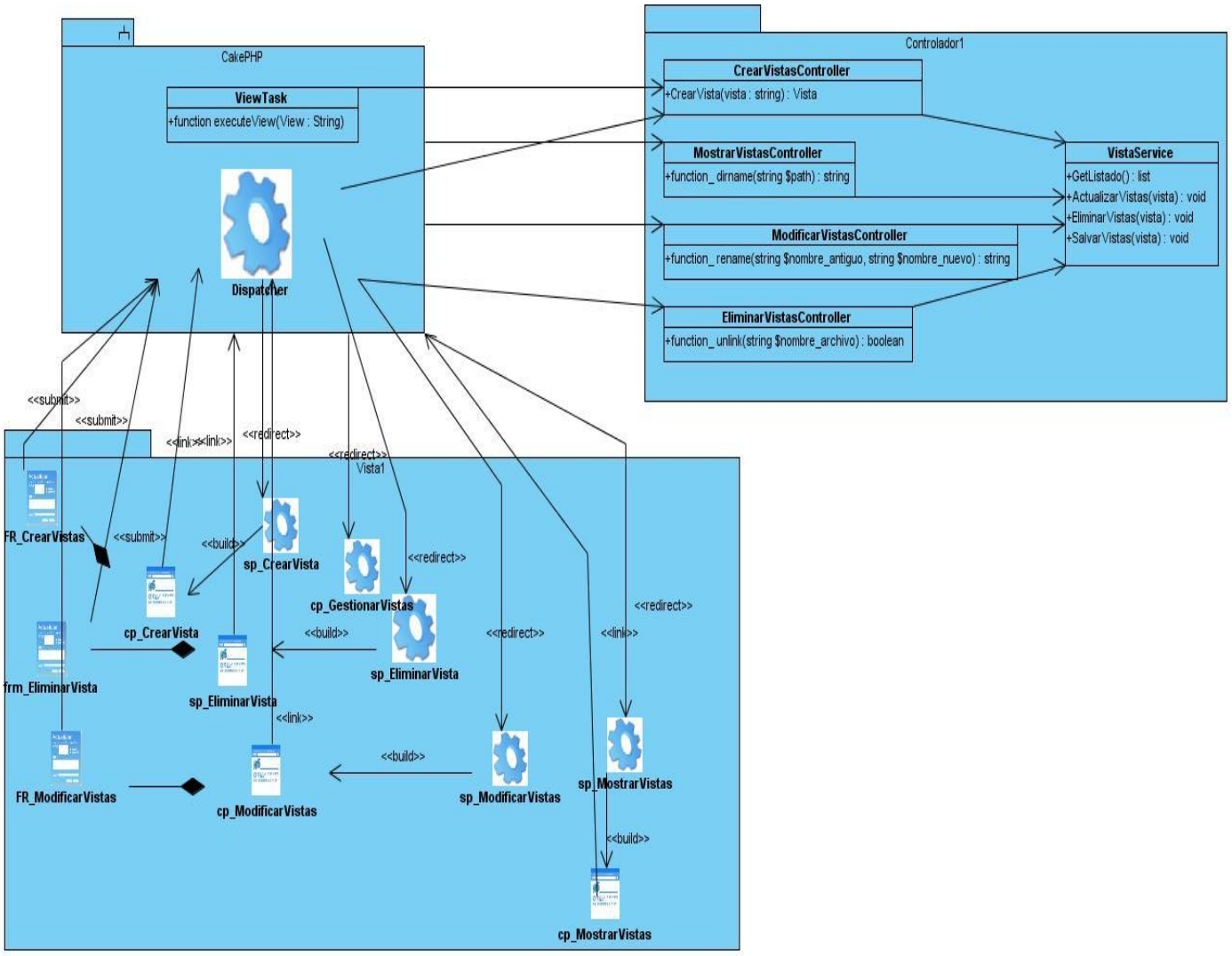



**Figura 3.9- Diagrama de Clases del Diseño Gestionar Vistas**

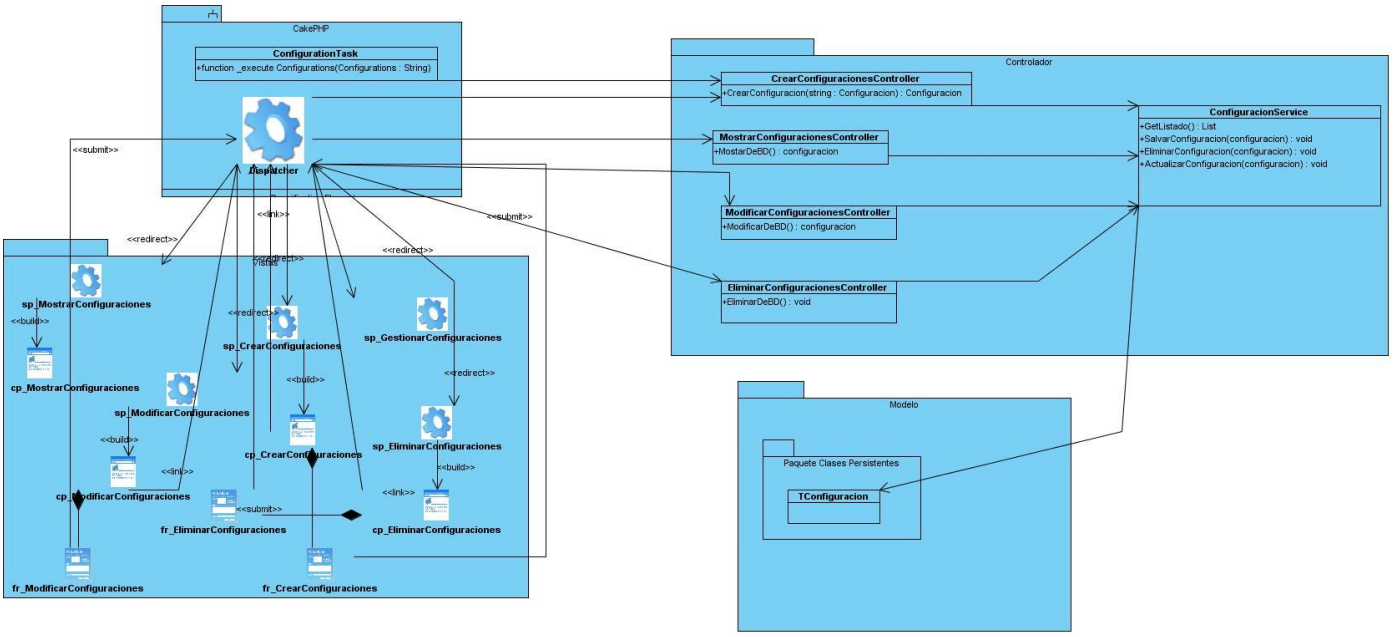



**Figura 3.10- Diagrama de Clases del Diseño Gestionar Configuraciones**

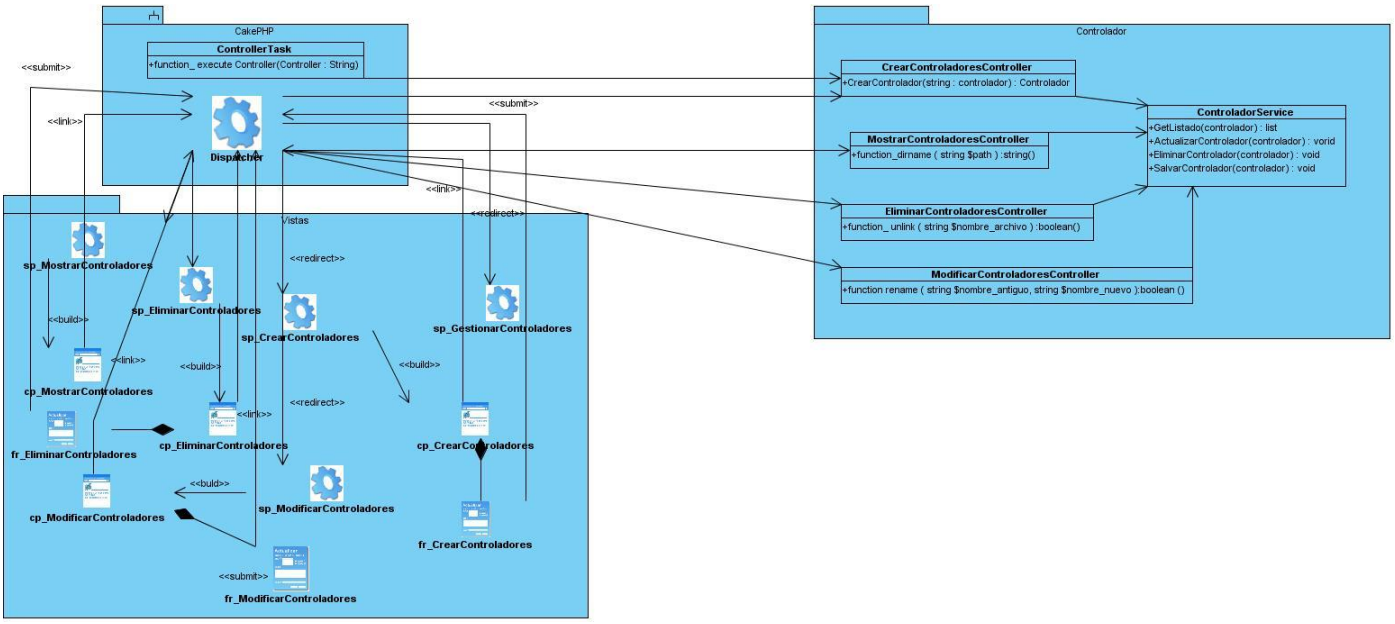



**Figura 3.11- Diagrama de Clases del Diseño Gestionar Controladores**

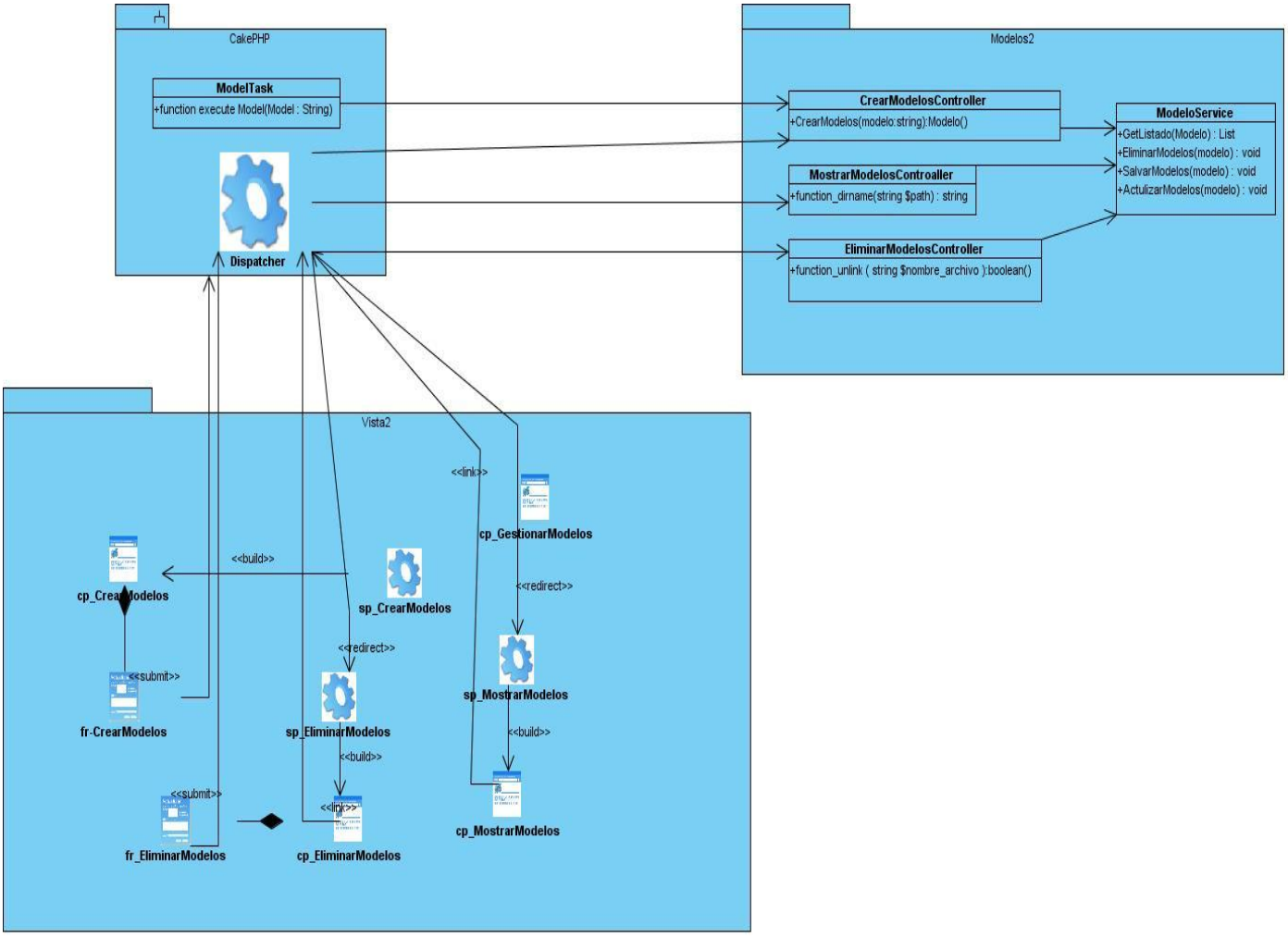



**Figura 3.12- Diagrama de Clases del Diseño Gestionar Modelos**

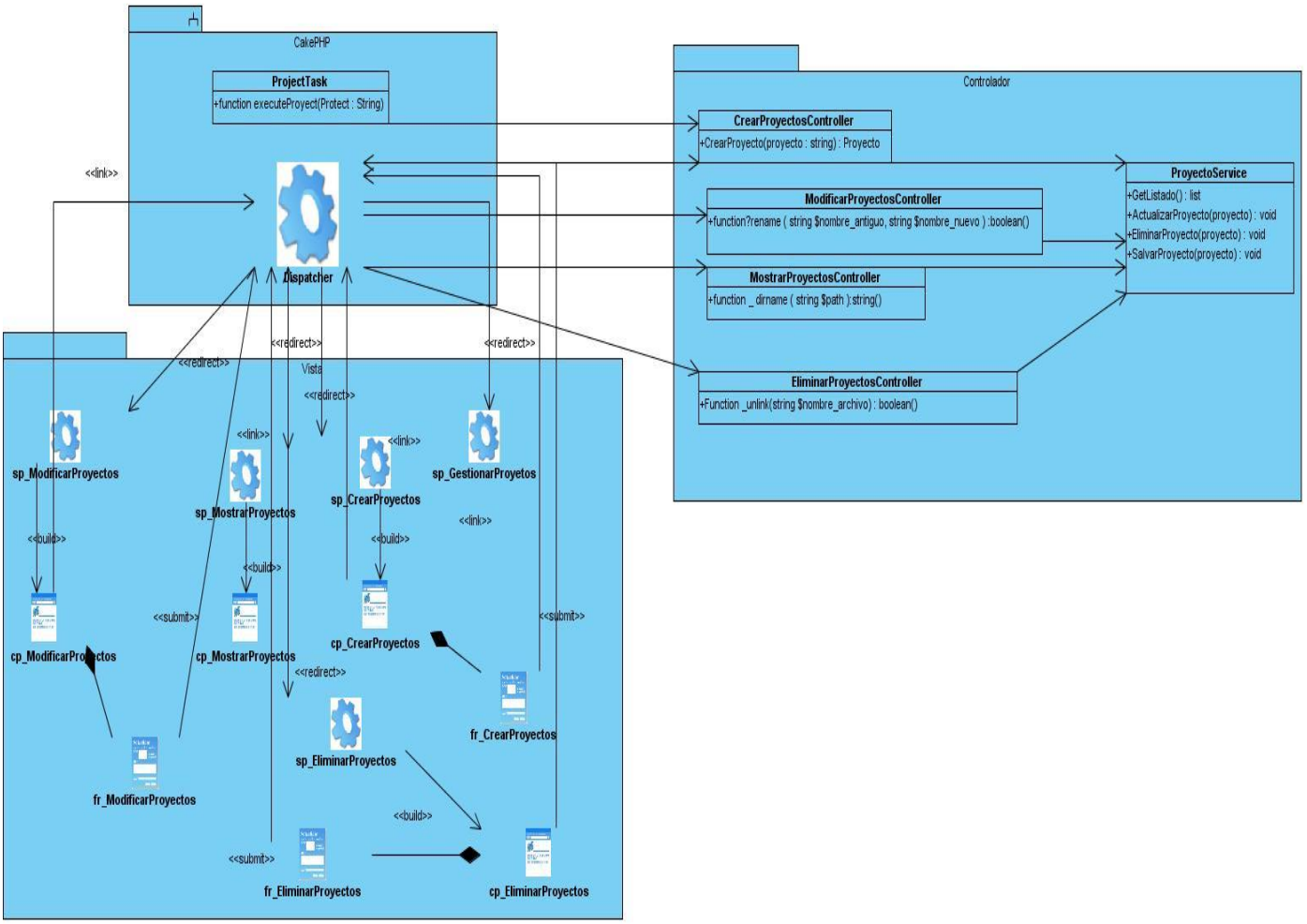



**Figura 3.13- Diagrama de Clases del Diseño Gestionar  Proyectos**

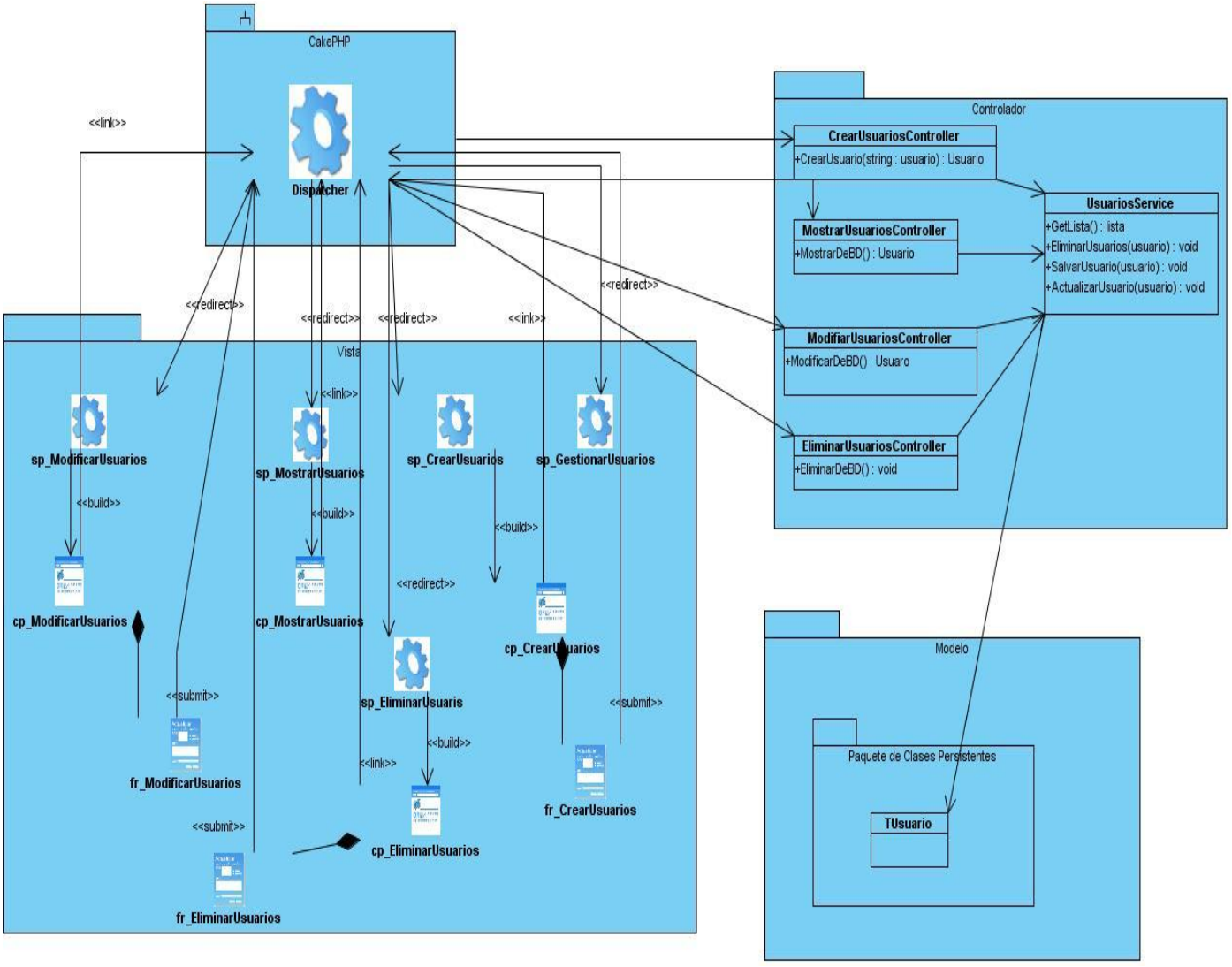



**Figura 3.14- Diagrama de Clases del Diseño Gestionar Usuarios**

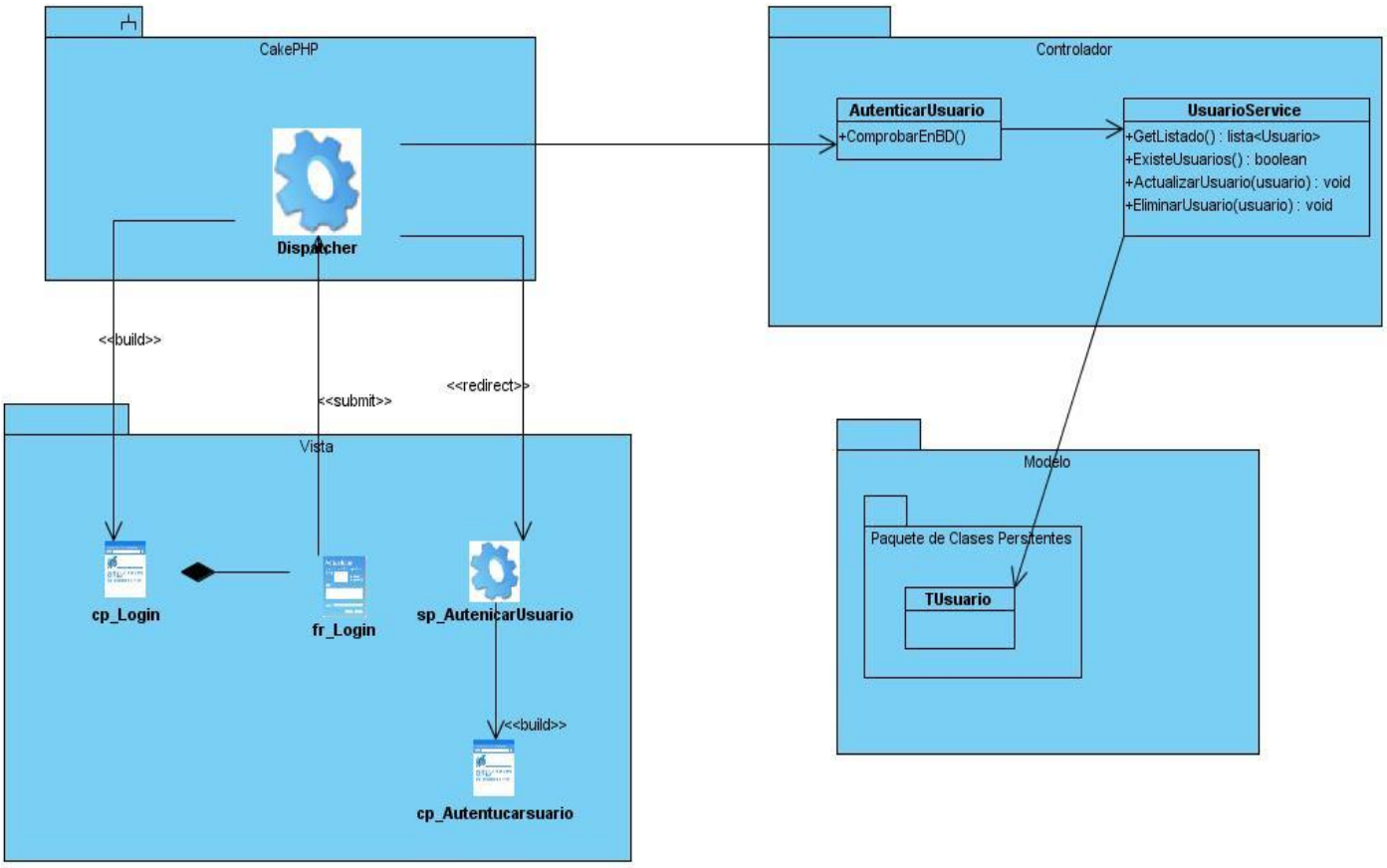



**Figura 3.15- Diagrama de Clases del Diseño Autenticar Usuarios**

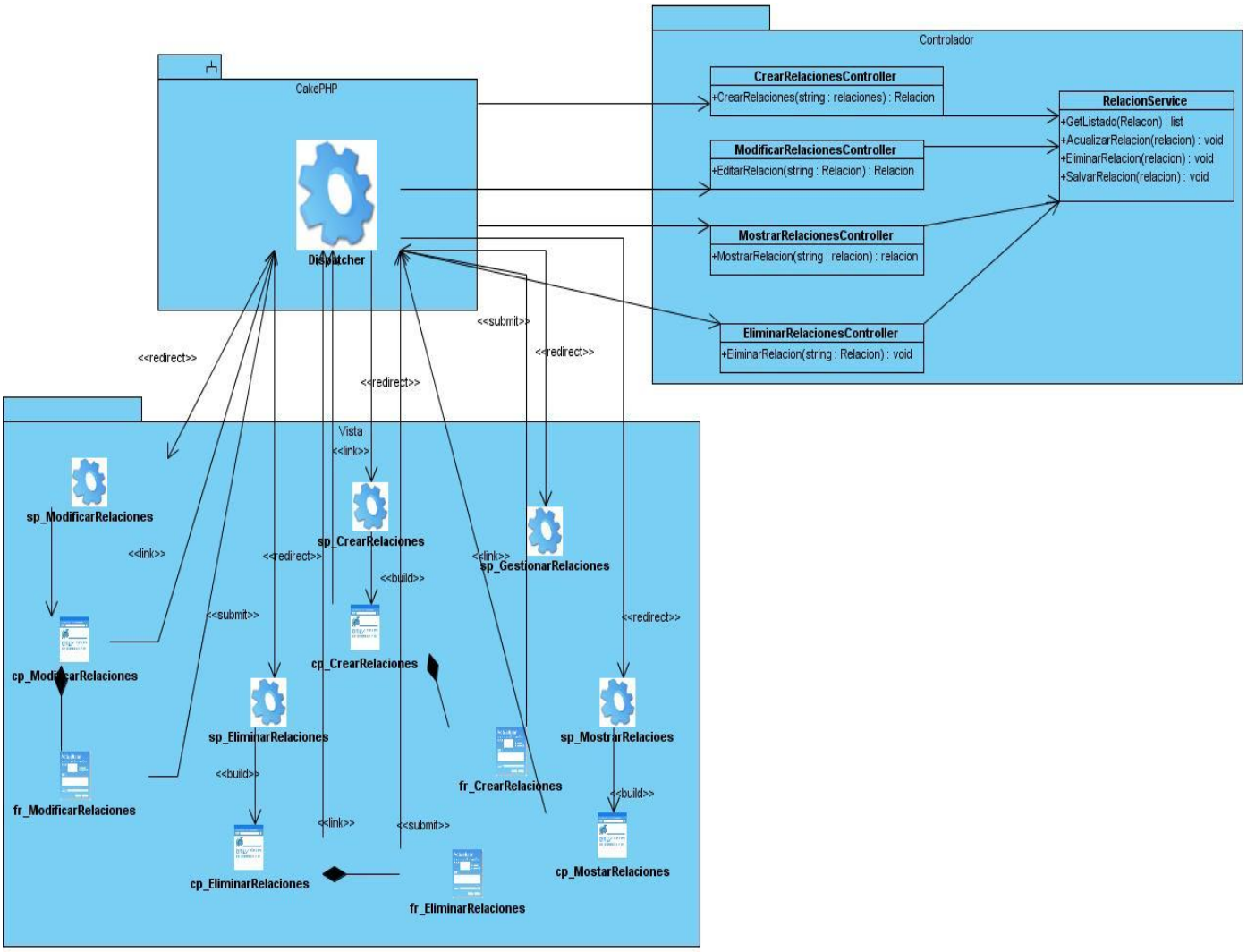



**Figura 3.16- Diagrama de Clases del Diseño Gestionar Relaciones.**

### 3.6 Descripción de las Clases Más Significativas.

Con este epígrafe se ofrece la información detallada de la estructura y funcionalidades que se muestran en las clases identificadas durante el diseño. Se toman como referencia las clases que controlan las operaciones y gestionan el intercambio de información entre el usuario y la base de datos del sistema.

**Clase: GestionarVistas**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con las vistas. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearVista() | Vista | public | void |
| ModificarVista () | Vista | public | void |
| ObtenerTodosLasVistas () | Vista | public | List |
| EliminarVista () | Vista | public | void |

**Tabla #2:** Descripción de la clase GestionarVistas.

**Clase: GestionarConfiguracion**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con las configuraciones. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearConfiguraciones () | Configuración | public | void |
| ModificarConfiguraciones () | Configuración | public | void |
| ObtenerTodosLasConfiguraciones () | Configuración | public | List |
| EliminarConfiguraciones () | Configuración | public | void |

**Tabla #3:** Descripción de la clase GestionarConfiguraciones.

**Clase: GestionarModelos**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con los Modelos. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearModelo() | Modelo | public | void |
| ModificarModelo () | Modelo | public | void |
| ObtenerTodosLosModelos () | Modelo | public | List |
| EliminarModelo () | Modelo | public | void |

**Tabla #4:** Descripción de la clase GestionarModelos.

**Clase: GestionarControladores**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con los Controladores. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearControladores () | Controlador | public | void |
| ModificarControladores () | Controlador | public | void |
| ObtenerTodosLosControladores() | Controlador | public | List |
| EliminarControladores () | Controlador | public | void |

**Tabla #5:** Descripción de la clase GestionarControladores.

**Clase: GestionarProyectos**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con los Proyectos. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearProyecto () | Proyecto | public | void |
| ModificarProyecto () | Proyecto | public | void |
| ObtenerTodosLosProyecto () | Proyecto | public | List |
| EliminarProyecto () | Proyecto | public | void |

**Tabla #6:** Descripción de la clase GestionarProyectos.

**Clase: GestionarUsuarios**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con los Usuarios. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| CrearUsuario () | Usuario | public | void |
| ModificarUsuario () | Usuario | public | void |
| ObtenerTodosLosUsuarios () | Usuario | public | List |
| EliminarUsuario () | Usuario | public | void |

**Tabla #7:** Descripción de la clase GestionarUsuarios.

**Clase: AutenticarUsuarios**

| | |
|---|---|
| **Estereotipo:** | << Controladora >> |
| **Descripción:** | Clase que gestiona los procesos de negocio relacionados con la autenticación de los usuarios. |

**Operaciones**

| Nombre | Parámetros | Visibilidad | Tipo retorno |
|---|---|---|---|
| ComprovarExistenciaDeUsuario () | Usuario | public | void |

**Tabla #8:** Descripción de la clase AutenticarUsuarios.

### 3.7 Diseño de la Base de Datos.

En el desarrollo de una aplicación informática, el diseño de la BD es de gran importancia, ya que en ella se almacenan todos los datos que son necesarios en la modelación del problema que se desea resolver, además ésta es la fuente de obtención de toda la información que se quiera recuperar del sistema. Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido, y recuperar la información.

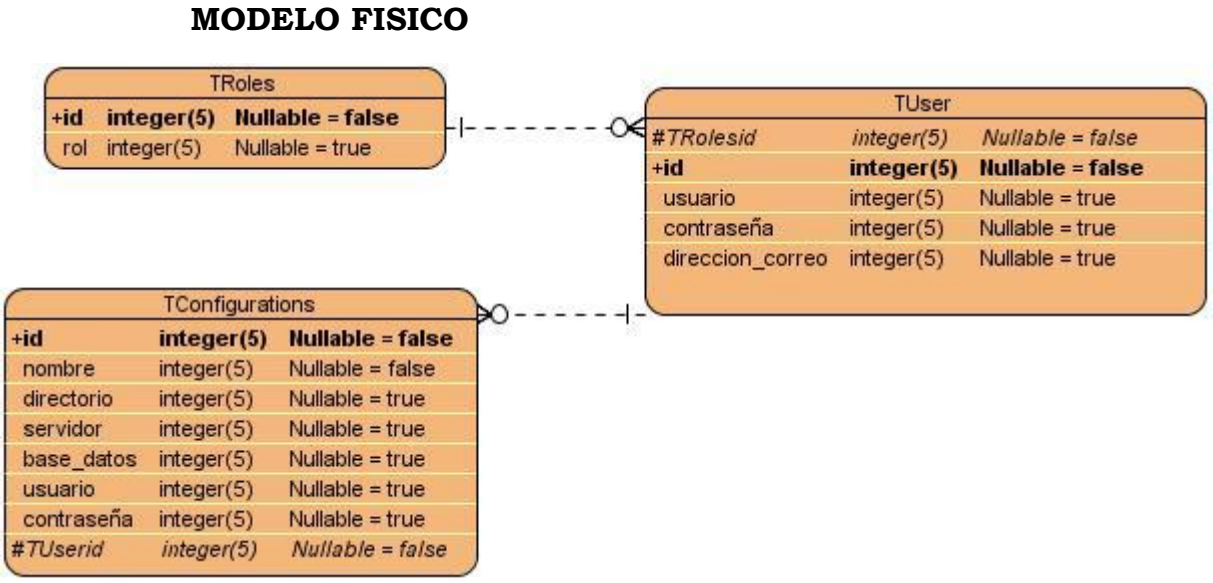



**Figura 3.18- Modelo de datos Físico.**

### 3.8 Diagrama de Despliegue.

El modelo de despliegue describe cómo una aplicación se despliega a través de una infraestructura. La intención del modelo de despliegue no es para describir la infraestructura, es el camino en el cual los componentes específicos deben corresponder a una aplicación que despliega a través de él. El modelo de

despliegue muestra la configuración (relaciones físicas) de los nodos que participaron en la ejecución y de los componentes hardware y software que residen en ellos.

En el siguiente epígrafe se muestra la distribución que deberá tener el sistema para su correcto funcionamiento, tanto en la arquitectura de software como de hardware, a través de un diagrama de despliegue.

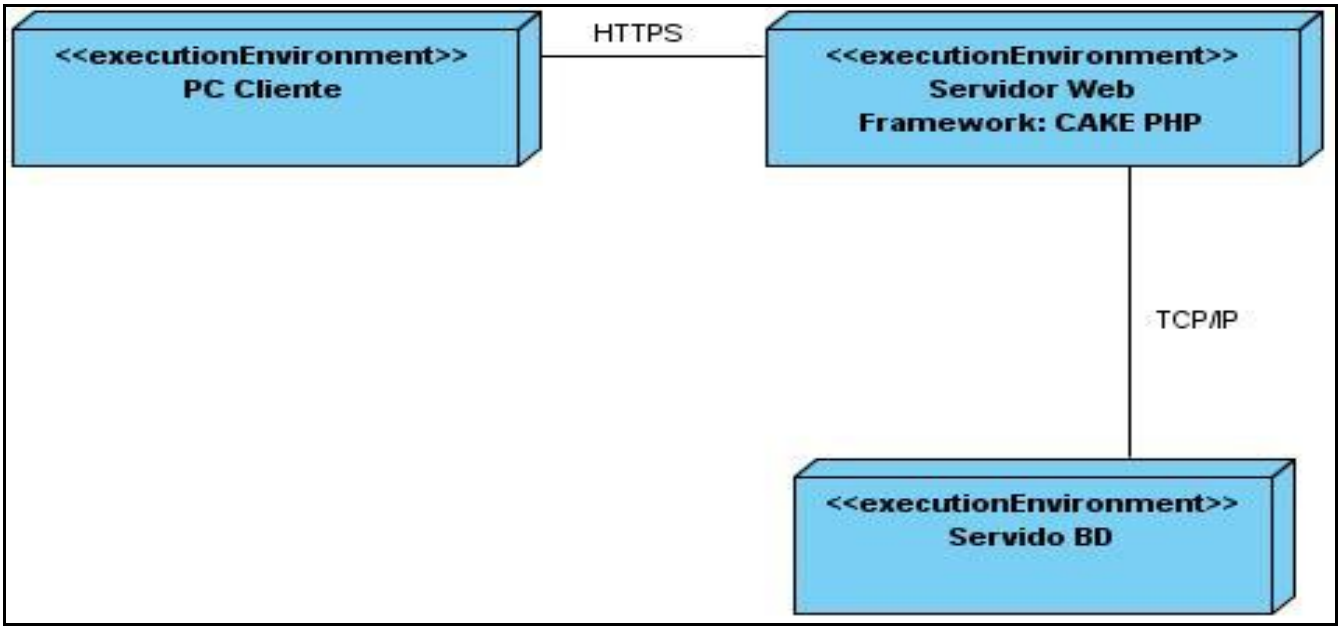



**Figura 3.19-Diagrama de Despliegue**

### 3.9 Métricas para la evaluación del Diseño.

Cuando los modelos de diseño aumentan en tamaño y complejidad resulta beneficioso aplicar métricas para verificar la calidad y objetividad del mismo. La visión objetiva de un modelo de diseño Orientado a Objetos (OO) se obtiene a través de la adecuada aplicación de métricas, cuyo resultado aporta el componente cualitativo en la evaluación que se hace. Dentro del diseño OO, se definen nueve características medibles: tamaño, complejidad, acoplamiento, suficiencia, integridad, cohesión, originalidad, similitud y volatilidad.

Las métricas buscan comprobar, de manera cuantitativa, en qué grado el software posee las distintas características que definen la calidad de un producto software, de ahí que su utilización sea una fase importante dentro de la evaluación del diseño. Entre las referenciadas por Pressman (30), se han seleccionado las que se aplican con mayor facilidad dentro del sistema en cuestión, considerando que este estudio brinda un modelo sencillo de implementar y que a su vez cubre los principales atributos de calidad de software. Lo referido al aseguramiento de la calidad es la principal razón por la que se conciben estas métricas. (29)

La valoración de las actividades realizadas dentro del modelo de diseño se ha realizado con la aplicación de las métricas siguientes:

- **Tamaño operacional de clase (TOC).**

Esta métrica expresa el número de métodos asignados a una clase. El impacto que tiene su aplicación en los atributos de calidad mencionados se evidencia mayormente en la responsabilidad, complejidad y reutilización. La relación está expresada porque el aumento de la cantidad de métodos de la clase es directamente proporcional a la responsabilidad y complejidad de la clase, lo que supone un aumento de los mencionados indicadores. De igual manera, se traduce en una reducción de la reutilización que pueda tener la misma.

- **Relaciones entre clases (RC)**

Métrica que está dada por el número de relaciones de uso de una clase con otras. El aumento de las RC se traduce en un aumento del acoplamiento, complejidad de mantenimiento, cantidad de pruebas que

requiere la clase; al tiempo que, de la misma forma que la métrica anterior, implica una reducción en la reutilización de la clase.

Luego de efectuar los cálculos para un:

| | |
|---|---|
| Total de clases | 16 |
| Promedio de procedimientos | 3,625 |

**Tabla 9 Cantidad de procedimientos por clases.**

| No | Nombre | Procedimientos |
|---|---|---|
| 1 | Lnegocio(CU_Gestionar_Vista) | 4 |
| 2 | LDatos(CU_Gestionar_Vista) | 4 |
| 3 | Lnegocio(CU_Gestionar_Controlador) | 4 |
| 4 | LDatos(CU_Gestionar_Controlador) | 4 |
| 5 | Lnegocio(CU_Gestionar_Modelos) | 4 |
| 6 | LDatos(CU_Gestionar_Modelos) | 4 |
| 7 | Lnegocio(CU_Gestionar_Configuraciones) | 4 |
| 8 | LDatos(CU_Gestionar_Configuraciones) | 4 |
| 9 | Lnegocio(CU_Gestionar_Relaciones) | 4 |
| 10 | LDatos(CU_Gestionar_Relaciones) | 4 |
| 11 | Lnegocio(CU_Gestionar_Proyecto) | 4 |
| 12 | LDatos(CU_Gestionar_Proyecto) | 4 |
| 13 | Lnegocio(CU_Gestionar_Usuarios) | 4 |
| 14 | LDatos(CU_Gestionar_Usuarios) | 4 |
| 15 | Lnegocio(CU_Autenticar_Usuarios) | 1 |
| 16 | LDatos(CU_Autenticar_Usuarios) | 1 |

Se obtuvieron las siguientes gráficas para la Responsabilidad, Complejidad y Reutilización, utilizando las siguientes tablas para obtener las categorías:

| | Categoria | Criterio |
|---|---|---|
| Responsabilidad | Baja | < =Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | > 2* Prom. |
| | | |
| Complejidad implementación | Baja | < =Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | > 2* Prom. |
| | | |
| Reutilización | Baja | > 2*Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | <= Prom. |

**Tabla 10: Responsabilidad y Complejidad de las clases.**

| Responsabilidad | Cantidad de clases | Promedio |
|---|---|---|
| Baja | 16 | 30,76923077 |
| Media | 0 | 0 |
| Alta | 0 | 0 |

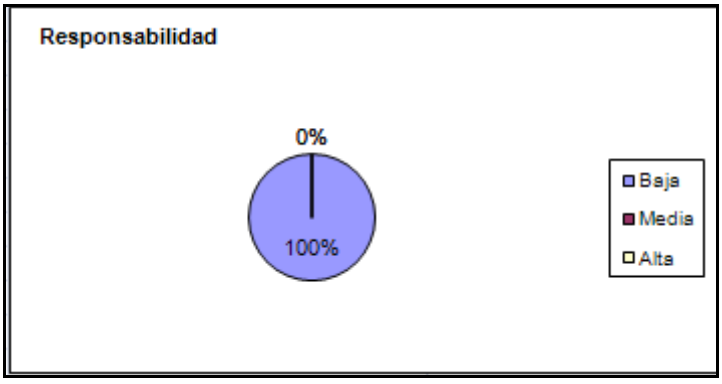



**Figura 3.20 - Responsabilidad de las clases**

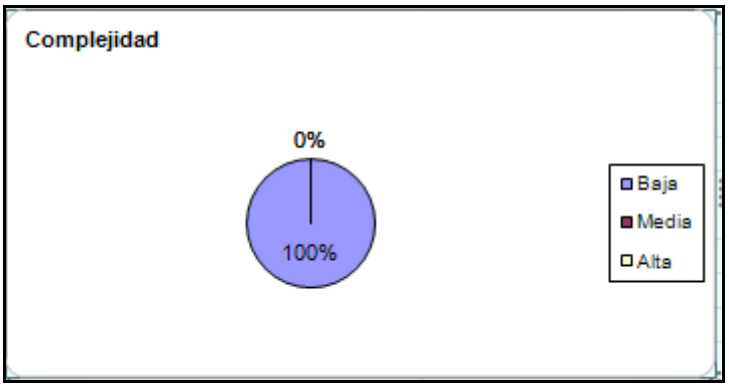



**Figura 3.21- Complejidad de las clases**

**Tabla 6 Reutilización de las clases.**

| Reutilización | Cantidad de clases | Promedio |
|---|---|---|
| Alta | 16 | 30,76923077 |
| Media | 0 | 0 |
| Baja | 0 | 0 |

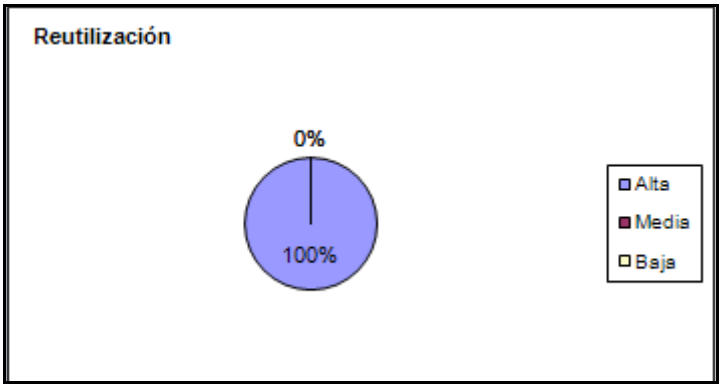



**Figura 3.22- Reutilización**

**Cuando la responsabilidad (Figura 3.20) de las clases entre media y baja es mayor que el 80% se puede decir que cumple con el patrón de alta cohesión, además de que presentan una reutilización alta.**

Para aplicar la métrica RC se tuvo en cuenta la cantidad de relaciones que tenían las clases del componente Incidencias, a partir del promedio de las relaciones y mediante un criterio se obtuvo la categoría (baja, media, alta y ninguna en caso del Acoplamiento) y (baja, media, alta) para la Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas.

Luego de efectuar los cálculos para un:

| | |
|---|---|
| Total de clases | 16 |
| Promedio de asociaciones de uso | 1,0625 |

**Tabla 7 de relaciones de uso por clases.**

| No | Nombre | Relaciones |
|---|---|---|
| 1 | Lnegocio(CU_Gestionar_Vista) | 1 |
| 2 | LDatos(CU_Gestionar_Vista) | 1 |
| 3 | Lnegocio(CU_Gestionar_Controlador) | 1 |
| 4 | LDatos(CU_Gestionar_Controlador) | 1 |
| 5 | Lnegocio(CU_Gestionar_Modelos) | 1 |
| 6 | LDatos(CU_Gestionar_Modelos) | 1 |
| 7 | Lnegocio(CU_Gestionar_Configuraciones) | 1 |
| 8 | LDatos(CU_Gestionar_Configuraciones) | 1 |
| 9 | Lnegocio(CU_Gestionar_Relaciones) | 1 |
| 10 | LDatos(CU_Gestionar_Relaciones) | 1 |
| 11 | Lnegocio(CU_Gestionar_Proyecto) | 1 |
| 12 | LDatos(CU_Gestionar_Proyecto) | 1 |
| 13 | Lnegocio(CU_Gestionar_Usuarios) | 1 |
| 14 | LDatos(CU_Gestionar_Usuarios) | 1 |
| 15 | Lnegocio(CU_Autenticar_Usuarios) | 1 |
| 16 | LDatos(CU_Autenticar_Usuarios) | 1 |

**Tabla10: Acoplamiento de las clases.**

| Acoplamiento | Cantidad de clases | Promedio |
|---|---|---|
| Ninguno | 0 | 0 |
| Bajo | 16 | 20,51282051 |
| Medio | 0 | 0 |
| Alto | 0 | 0 |

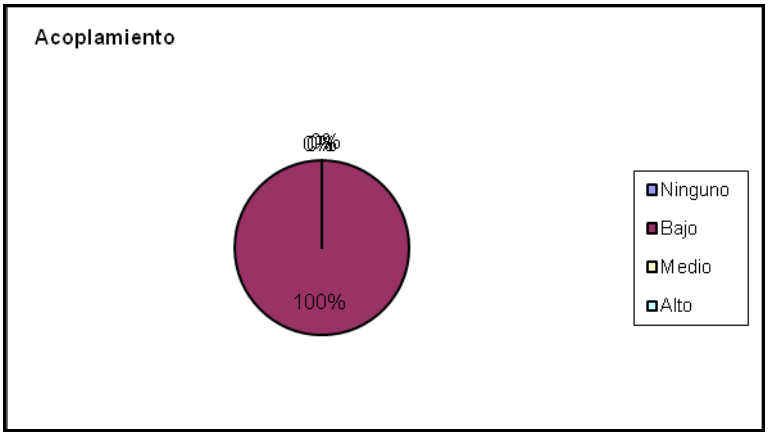



**Figura 3.23- Representación del acoplamiento entre las clases.**

**Tabla Reutilización de las clases.**

| Reutilización | Cantidad de clases | Promedio |
|---|---|---|
| Baja | 0 | 0 |
| Media | 0 | 0 |
| Alta | 16 | 20,51282051 |

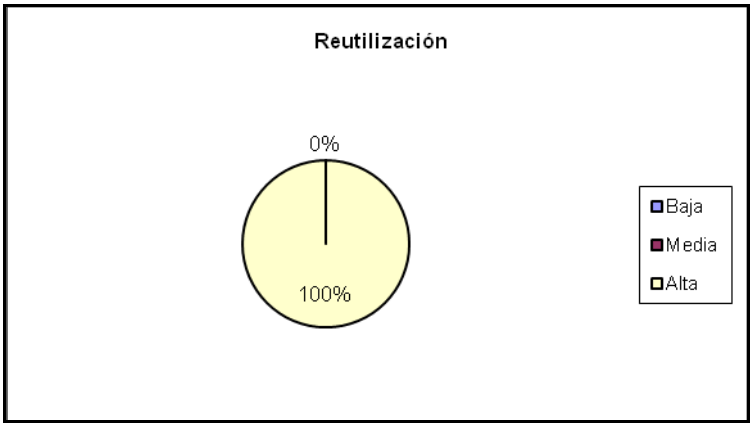



**Figura 3.23- Representación de la reutilización**

## 3.10 Conclusiones Parciales

En el presente capitulo se realizo el análisis y diseño de la aplicación que se construirá posteriormente. Para el mismo se obtuvieron varios diagramas como: el diagrama de despliegue, los de clases del análisis y de clases del diseño, así como los diagramas de colaboración para cada una de las funcionalidades descritas. Se describe la arquitectura y los patrones que se tuvieron en cuenta para el diseño de los casos de uso. Se muestran además los modelos físicos y lógico, modelos indispensables en el diseño de la base de datos. Como parte de la validación del diseño se le aplicaron sendas métricas con el propósito de comprobar el grado de validez que brindará el diseño del sistema para su posterior implementación.

## CONCLUSIONES

Con el desarrollo del siguiente trabajo de diploma se arriba a las siguientes conclusiones:

- Después de un detallado estudio detallado del framework CakePHP se procedió a su posterior caracterización en aras de mejorar su proyección como software libre de desarrollo web.
- Se procedió al desglose y caracterización de todas las funcionalidades de la consola cake bake como herramienta de configuración y desarrollo de CakePHP.
- Se aplicaron una serie de técnicas para la captura de requisitos con el objetivo de identificar requisitos funcionales y no funcionales y se realizó la descripción de una herramienta de administración y configuración para CakePHP.
- Se realizó el diseño del sistema utilizando patrones, logrando una estructura general robusta y de fácil mantenimiento.
- Se procedió a la validación del diseño, aplicando dos tipos de métricas y se concluyó que el diseño del sistema no presenta fisuras.

## RECOMENDACIONES:

Después de la investigación y desarrollo del presente trabajo y para concluir el mismo de forma satisfactoria, recomendamos:

- Que se realice la posterior implementación de la herramienta y extender su uso en la universidad y en toda la comunidad del software libre mundial.
- Publicar el presente trabajo en eventos relacionados con el tema.
- Incluir en posteriores versiones del producto las nuevas funcionalidades que se le agregan a la consola de CakePHP en sus futuras versiones.

## BIBLIOGRAFÍA REFERENCIADA


1. **Potencier, Fabien.** *Symfony.* 2008.

2. **Alvarez, Miguel Angel.** [En línea] 23 de diciembre de 2009. [Citado el: 12 de enero de 2010.] http://www.desarrolloweb.com/articulos/modelo-vista-controlador-codeigniter

3. **Anderson, John David.** *CAKEPHP 1.2.* 2007.

4. **Pressman, Roger S.** Ingeniería de Software, un enfoque práctico. Quinta edición. s.l. : McGraw-Hill Companies, 2002. pág. 640. ISBN: 8448132149.

5. **Alvarez, Miguel Angel.** [En línea] 09 de mayo de 2001. [Citado el: 25 de febrero de 2010.] http://www.desarrolloweb.com/articulos/392.php

6. **Alvarez, Miguel Angel.** [En línea] 14 de Marzo de 2009. [Citado el: 23 de enero de 2010.] http://www.desarrolloweb.com/manuales/27

7 **Martínez, Jaime Solís.** [En línea] 2008. [Citado el: 5 de febrero, 2010.]

8. **Joseph, Jaime Solís.** [En línea] 2008. [Citado el: 5 de febrero, 2010.] http://netbeans.org

9. **Illera, Alvaro Marín.** [En línea] 5 de enero de 2007. [Citado el: 12 de abril de 2010.] http://www.e-ghost.deusto.es/docs/TutorialMySQL.html.

10. **Gutiérrez., Javier J**. ¿Qué es un framework web? 2008.

11. **Sánchez, Jordi.** [En línea] 29 de Septiembre de 2006 . [Citado el: 6 de enero de 2010.] http://jordisan.net/blog/2006/que-es-un-framework.

12. **Alvarez, Miguel Angel.** [En línea] 9 de Mayo de 2001. [Citado el: 10 de febrero de 2010.] http://www.desarrolloweb.com/articulos/392.php.

**13. Anderson, John David.** *CAKEPHP 1.2.* 2007

**14.** 14. **Sanchez, María A. Mendoza**. [Online] Junio 7, 2004. [Citado el: Febrero 15, 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

**15.** Lenguaje de modelado UML

**16. Estevan, Dina.** [En línea] 24 de Noviembre de 2008. [Citado el: 8 de Marzo de 2010.] http://www.todoprogramas.com/programalinux/visualparadigmforuml.

**17. Toro, A. Durán.** [En línea] 23 de marzo de 2008. [Citado el: 3 de marzo de 2010.] http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER00/toro.pdf

**18. Tedeschi, Nicolás.** [En línea] junio de 2003. [Citado el: 1 de Marzo de 2010.] http://msdn.microsoft.com/es-es/library/bb972240.aspx

**19. Zavala-Ruiz, J.** [En línea] 2008. [Citado el: 4 de febrero de 2010.] http://directorio.adfound.com/visitar.php?ID=1612&nombre=Apuntes%20sobre%20Ingenier%EDa%20de%20Software&url=http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html&descripcion=Apartes%20de%20una%20tesis%20de%20Maestr%EDa%20sobre%20el%20proceso%20.

**20. Cuaresma, María José Escalona.** [En línea] 12 de enero de 2007. [Citado el: 3 de mayo de 2010.] http://www.lsi.us.es/docencia/get.php?id=2118. (Modelo de Análisis)

**21. Tedeschi, Nicolás.** [En línea] junio de 2003. [Citado el: 1 de Marzo de 2010.] http://msdn.microsoft.com/es-es/library/bb972240.aspx

**22. INFORMATICA, INSTITUTO NACIONAL DE ESTADISTICA E.** *Herramientas Case.* 1999.

**23. Garcerant, Iván.** [En línea] 7 de julio de 2007. [Citado el: 15 de febrero de 2010.] http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional.

24. **Sanchez, María A. Mendoza.** [En línea] 7 de Junio de 2004. [Citado el: 23 de Marzo de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

25. **Gonzales, Carlos.** [En línea] 17 de Junio de 2009. [Citado el: 23 de Marzo de 2010.] http://comunidadcodeigniter.wordpress.com/2009/06/17/utilizando-el-scaffolding.

26. **Valle, Amaury** E. del. juventudrebelde. [Online] Febrero 14, 2008. [Citado el: Enero 25, 2010.] http://www.juventudrebelde.cu/cuba/2008-02-14/software-libre-ii-una-estrategia-decisiva-de-desarrollo

27. **Orallo, Enrique Hernández**. El Lenguaje Unificado de Modelado(UML). Valencia,España : s.n., 2007.

28. **Chaves, Michael Arias.** La ingenieria de requerimientos y su importancia en el desarrollo de proyectos de software. San José : s.n., 2005.

29. **Torres, Manuel Lagos.** El Rincón del Programador. sitio Web El Rincón del Programador. [En línea] 28 de Diciembre de 2002. [Citado el: 10 de Marzo de 2010.]

30. **Pressman, Roger S.** Ingeniería de Software, un enfoque práctico. Quinta edición. s.l. : McGraw-Hill Companies, 2002. pág. 640. ISBN: 8448132149.

## GLOSARIO DE TERIMINOS

**Internet :**es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California y una en Utah, Estados Unidos.

**ASP:** Active Server Pages (ASP), también conocido como ASP clásico, es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Services (IIS).

**ASP.NET:** es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

**JAVA**: La **plataforma Java** es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de bibliotecas estándar que ofrecen funcionalidad.

**Orientación a Objeto: La programación orientada a objetos** o **POO** (**OOP** según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos. La orientación a objetos puede describirse como el conjunto de disciplinas que desarrollan y modelan el software que facilitan la construcción de sistemas complejos a partir de componentes.

**Licencia del MIT** es a licencia libre del software el originar en Instituto de Tecnología de Massachusetts (MIT). Es a permisivo licencie, significando que permite la reutilización dentro software propietario a condición de que la licencia se distribuye con ese software, y GLP-compatible, significando que GLP permisos combinación y redistribución con el software que utiliza la licencia del MIT.

**CodeIgniter**: es un entorno de desarrollo abierto que permite crear webs dinámicas con PHP. Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero.

**Symfony**: es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web.

**Open Source**: **Código abierto** es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

**Ingeniería de Requisitos:** La ingeniería de requisitos o los requisitos en sí, constituyen el enlace entre las necesidades reales de los clientes, usuarios y otros participantes vinculados al sistema. La ingeniería de requisitos consiste en un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requerimientos del sistema siguiendo un determinado estándar.