Universidad de las Ciencias Informáticas

Facultad 6

Título: “Desarrollo de un software para un Cuadro de Mando Integral”

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

Autores: Michel Díaz Hernández

Gianny Pacheco López

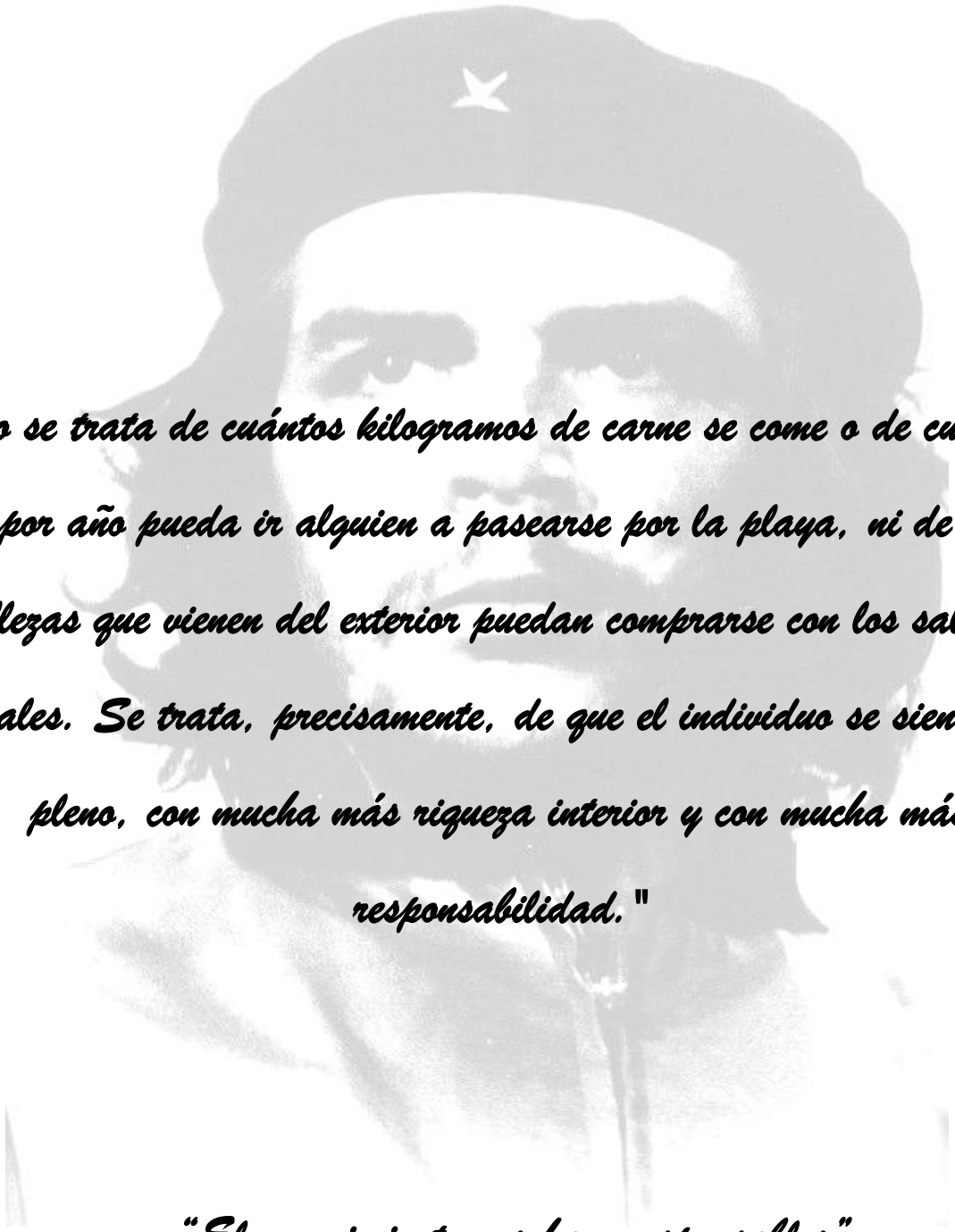
Yadier Mesa Rodríguez

Tutores: MSc. Maykel Y. Leyva Vázquez

MSc. Yosdenis Urrutia Badillo

Ing. Yamila Corona Puig

Junio 2009



"No se trata de cuántos kilogramos de carne se come o de cuántas veces por año pueda ir alguien a pasearse por la playa, ni de cuántas bellezas que vienen del exterior puedan comprarse con los salarios actuales. Se trata, precisamente, de que el individuo se sienta más pleno, con mucha más riqueza interior y con mucha más responsabilidad."

"El conocimiento nos hace responsables"

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 firmamos la presente a los _____días del mes de _____del año 2009.

Michel Diaz Hernández

_____
Firma del Autor

Gianny Pacheco López

_____
Firma del Autor

Yadier Mesa Rodríguez

_____
Firma del Autor

Maikel Y. Leyva Vázquez

_____
Firma del Tutor

Yosdenis Urrutia Badillo

_____
Firma del Tutor

Yamila Corona Puig

_____
Firma del Tutor

Contactos

Tutores:

Ing. Yamila Corona Puig

Especialidad de graduación: Ingeniería Industrial

Categoría docente: Instructor Recién Graduada

Categoría Científica: Ingeniera.

Años de experiencia en el tema: 2

Años de graduado: 2

Universidad de las Ciencia Informáticas, La Habana, Cuba.

ycorona@uci.cu.

MSc. Maikel Yelandi Leyva Vázquez

Especialidad de graduación: Ingeniero Informático

Categoría docente: Instructor

Categoría Científica: Máster

Años de experiencia en el tema: 3

Años de graduado: 5

Universidad de las Ciencia Informáticas, La Habana, Cuba.

mleyvaz@uci.cu

MSc. Yosdenis Urrutia Badillo.

Universidad de las Ciencia Informáticas, La Habana, Cuba.

Especialidad de graduación: Ciencias Empresariales

Categoría docente: Instructor

Categoría Científica: Máster

Años de experiencia en el tema: 2

Años de graduado: 5

yosdenis@uci.cu.

Agradecimientos

Agradecimiento Especial y Eterno a mis padres por haberme formado como la persona que hoy soy y haber permitido que lograse este sueño.

Agradezco a mi abuelo y a mi hermana por las enseñanzas trasmitidas, por siempre haber confiado en mí y por permitirme hoy ser mejor persona, a mi sobrino por haber dado un toque mayor de responsabilidad a la realización de esta Tesis aún cuando no cumple un año de vida, pues soy **Tío.**

Agradecimiento Especial a mi novia Marilenys por todo su apoyo y confianza durante más de dos años, por haber estado siempre para mí y por haberme motivado cuando lo necesité.

Agradezco a mi familia por haber depositado toda su confianza en cada momento difícil de mi vida y haber apostado porque saldría adelante, decirles que espero no defraudarlos nunca.

Agradezco al marido de mi hermana por hacerla feliz y por cuidar de ella, por sus consejos para con mi Tesis y por hoy haberme dado junto a mi hermana mi primer sobrino.

Agradezco a todos los que me ayudaron a confeccionar este Trabajo de Tesis, sin ellos no hubiera sido posible realizar el mismo, dígase Rachid, Maikel Pérez, Felipe, Dayron, Yarell, al primo de Michel, Daryanis, a los tutores y la oponente en fin todos los que de una u otra forma me ayudaron.

Agradezco a mis compañeros de tesis por todo este tiempo que pasamos trabajando juntos, y que también gracias a ellos fue posible esta Tesis.

Agradezco de sobremanera a todos los que de una u otra forma han contribuido en mi formación como persona y han ayudado también a hacer posible este sueño.

Agradezco a todos los que a lo largo de mi formación como estudiante contribuyeron para que hoy pudiera dejar de serlo, con apartado especial para los que más allá de contribuir, lo permitieron.

Agradezco a todos los profesores que he tenido en mi vida pues de todos he aprendido en su debido momento, con apartado especial para los del Pre-Universitario y para los que aquí en la Universidad tanto me han ayudado, me refiero a Leansy, Walter, Castaño, Armenteros, Yaima; en fin a todos ellos gracias.

Agradezco a todos los amigos que he encontrado aquí en la Universidad, y a todos con los que he pasado momentos importantes, en especial a mis amigos de Camagüey y de la facultad.

Agradezco a todos los que siempre han confiado en mí y a los que lo han hecho en algún momento. A los que no confían, nunca lo han hecho y tal vez nunca lo hagan, las gracias, por hacerme más fuerte cada día y por mostrarme el camino hacia la meta.

Yadier

Agradezco a mis padres y hermano por ser mis guías y mi aliento para lograr mis metas, a mi familia por apoyarme y hacer que cada día sea mejor, a mi primo Arniel Serrano Hernández por ser más que un primo, es un hermano del cual estoy orgulloso de sus logros y agradecido por estar ahí en los momentos claves cuando he necesitado de su ayuda. A mis primas Yanay y Yanary Hernández Sosa por ser los dos tesoros más preciados que tengo, por estar siempre dándome ánimo y carisma para poder seguir adelante, por tener siempre su cariño incondicional que me hacen sentir orgullosos de poder contar con ellas en cualquier momento.

Agradecer de una forma especial a Yarell Leyva que ha sido como el hermano mayor que no he tenido en la Universidad, por ayudarme a en la realización de esta tesis, por darme sus consejos y estar ahí en los buenos y malos momentos que he vivido en el transcurso de estos cinco años que han sido una experiencia única, a mis hermanos del preuniversitario Herminio Pérez (Taty), Alejandro González (Ale), Osmel Chapman (El Negro), a sus respectivas madres que han sido madres para mí también, a mis amigos y hermanos de universidad los cuáles nunca olvidaré Norlen Rosales (El Chupao), Carlos Javier (El Perro), a Landy Gonzales (El Punto), por haber sido mis primeros compañeros y amigos que tuve en esta escuela, a Arsenio Llamo (EL Negro), Nosbel Rivera (El Pinareño), Adrian Martinez , Lino Ballagas, Jorge Carlos Puig (El guajiro), Rigoberto Salgado por haber sido otros de los que podría llamar hermanos de Universidad, así como a todas las personas con las que he podido compartir en mis grupos y apartamentos por los cuáles he pasado, a todos ellos los llevo conmigo y de los cuáles estoy orgulloso de haber compartido más que clases con ellos, a Tania Oropesa por haber sido uno de mis motivos para terminar esta tesis, alentándome en los momentos de presión, por estar ahí en los momentos que necesitaba compañía. A todas las personas que sin su ayuda no hubiese podido lograr que este trabajo se realizara, por haberme dado los conocimientos de forma incondicional, por brindarme su mano en momentos de errores fatales de la aplicación, esas personas las cuáles son Rachid Alí, Yenisel Avila, Felipe Arias, Dairon Freddy y aquellos que de una forma u otra me han apoyado a llegar hasta aquí.

Michel

Primero que todo a Dios, pues nunca me dejo sólo, y en la más difícil prueba siempre supo darme la fuerza para seguir adelante…

A mis padres pues su amor incondicional, sabio consejo e inquebrantable confianza hicieron posible que hoy esté aquí….

A mi familia, que siempre estuvo pendiente de mí…

A todos mis amigos, sobre todo a esos que saben que son especiales para mí.
A mis compañeros de tesis, que a pesar de las preocupaciones, siempre supieron echar pa'lante.
A todos aquellos cuya ayuda desinteresada hizo posible que nuestro trabajo de diploma saliera.
A todos, GRACIAS!

Gianny

Agradecemos a nuestros Tutores por el apoyo brindado a lo largo de la realización de este Trabajo de Diploma, pues nos fueron útiles en cada momento que los necesitamos.
Agradecer a la Revolución por permitirnos formar parte de este brillante proyecto de la Batalla de Ideas que es la **UCI** y por contribuir de sobremanera en nuestra formación, tanto profesional como personal.

Dedicatoria

A la memoria de los que hoy no están a mi lado físicamente, pero que han sido y serán parte inigualable en mi vida.

A quien ha sido mi ejemplo intelectual a seguir durante toda mi vida, **mi Abuelo**.

A quienes han sido mi voluntad y mis fuerzas, mi razón de ser y los que siempre han mantenido vigentes mis sueños, **mis Padres**.

A quien siempre ha estado de mi lado en cada momento de mi vida y ha sabido ganarse mi admiración, **mi Hermana**.

A quienes siempre, ante cualquier dificultad han confiado en mí aún cuando he estado frente a la peor de las situaciones, **mis Tíos de Baraguá.**

A quien ha marcado un antes y un después en mi vida, como persona y como pareja, y de igual forma ha confiado en mí aún cuando no lo he merecido, a quien en solo dos años ha logrado en mí lo que a otra persona le hubiese llevado una vida, mi **Novia Eterna, Marilenys.**

A quien ha compartido y vivido conmigo momentos que tanto han contribuido en mi formación, mi **Amigo Marvin.**

A quienes han compartido conmigo momentos importantes desde la infancia así como mis triunfos y derrotas, y aún hoy somos un equipo, mis **Amigos del Pre-Universitario.**

A quienes me han apoyado en los momentos que lo he necesitado y han contribuido a mi formación como persona, mis **Tíos Emilio y Pedro**.

A quienes también han sido parte de mi vida y formación, y en quienes he encontrado un ejemplo a seguir, mis **Primos de Baraguá y Ciego.**

A quienes han compartido parte de los momentos de mi infancia y con quienes he aprendido, mis **Primos de Majagua.**

A mis vecinos en **Majagua** y al resto de las personas que sé que confían en mí como persona y como todo lo que se puede confiar en un ser humano.

A ustedes, quienes han ganado que humildemente los quiera, y que de una u otra forma han ayudado tanto en mis victorias presentes, en las que aún no he conseguido pero que sin duda lucharé por alcanzar, decirles que no los defraudaré, que intentaré ser siempre la persona que un día ganó su respeto y causó en ustedes admiración.

A todos ustedes dedico esta, la que también, sin lugar a dudas, es su Tesis.

Yadier

Dedico esta tesis a mis padres y a mi hermano por ser las personas que más quiero en este mundo en especial a mi mamá Nelsy Esther Hernández Rubio por ser la persona que me trajo a este mundo, por ser el pilar que me ha hecho mantenerme firme ante situaciones difíciles, por aguantar mis malcriadeces y por ser la razón de seguir cada día forjando mi camino para poderle dedicar cada logro a ella, pues gracias a ella soy la persona que soy. A mi hermano Misael Damián Díaz Hernández por ser mi ejemplo y mi reto a seguir, por darme todo el apoyo para lograr mis éxitos, porque es la persona que idolatro y sin él tampoco fuera la persona que soy. A mi padre Misael Aurelio Díaz Torres junto a mi mamá por darme la educación que tengo, por darme el apoyo que me ha hecho falta, por ser mi otro ejemplo a seguir, por su carácter, su responsabilidad y por haberme dado los valores que tengo.

Michel

A “Mamán”, quien me enseñara a dar mis primeros pasos…
A “Paíta”, mi segunda madre...

Gianny

Resumen

El Cuadro de Mando Integral (en lo adelante CMI) o Balanced Scorecard (en lo adelante BSC) es una metodología diseñada para implantar la estrategia de una Empresa u Organización, fue divulgado en 1992 por los autores Robert Kaplan y David Norton. Desde su aparición se ha convertido en una herramienta cada vez más utilizada por reconocidas corporaciones internacionales, las cuáles han obtenido excelentes resultados. Si bien la literatura y la práctica muestran un mayor grado de avance de esta herramienta en el concierto privado, existe evidencia acerca de su uso en instituciones públicas y quiere hacerse expansivo a las Universidades.

La motivación para realizar este trabajo de Tesis nace con el objetivo de que el Centro de Consultoría de la UCI pueda llevar a cabo mejor sus actividades, desde el punto de vista de su administración. Se procura además ampliar los conocimientos en los usuarios y beneficiados en general acerca del uso de aplicaciones informáticas, en el caso particular de un CMI. Así mismo se realiza su descripción, se dice para lo que se utiliza, cuál es su propósito, y los beneficios que se pueden obtener con esta importante herramienta.

Para lograr el resultado deseado se dio cumplimiento a los requerimientos planteados, obteniéndose luego de la terminación, un producto capaz de cumplir con las necesidades y vicisitudes impuestas. Se evidenció la calidad de la aplicación mediante la aceptación de la misma por parte del cliente.

PALABRAS CLAVES: Cuadro de Mando Integral, Organizaciones, Estrategia, Perspectivas, Objetivos, Indicadores.

ÍNDICE

Introducción

La competitividad en la que hoy nos vemos inmersos los seres humanos, la radicalidad en las disímiles esferas de la vida, el inmenso bregar por caminos aún no transitados por el hombre y el honroso anhelo por ser decorosos, humildes por antonomasia, fuertes, mejores y más preparados cada día nos incita a utilizar nuestro intelecto en labores netamente productivas, ya sea con carácter didáctico, financiero, curricular o por sencillamente ser como aquellos que idolatramos desde que pudimos hacerlo.

Las Empresas u Organizaciones no deberían estar de forma alguna exentas a ninguna de estas premisas y es por ello que aquellas que como principio no tienen un control adecuado de su funcionamiento así como un seguimiento sostenido de su rendimiento terminan quebrando. La necesidad de contar con información en tiempo y forma es un tema crucial en las compañías modernas, donde entienden que en el mundo de alta competitividad, contar con *indicadores clave* permite tomar decisiones acertadas, diferenciándose estratégicamente de sus competidores[1] (Argentino, 2007).

Medir el desempeño de una Empresa en todos sus renglones así como su rendimiento siempre fue una tarea bastante compleja, pues existe una gran cantidad de aspectos que son necesarios tener en cuenta a la hora de hacer valoraciones o de arribar a conclusiones desde el punto de vista gerencial. Para cubrir este espacio que no estaba siendo medido en las empresas los doctores Robert S. Kaplan y David P. Norton dan a conocer en 1992: el Balanced Scorecard, conocido también como Cuadro de Mando Integral cuyo objetivo principal es "*convertir la visión y la estrategia de una organización en objetivos, indicadores, metas e iniciativas y que el desarrollo de dicha estrategia no se centre solamente en el marco financiero, sino mas bien enfocada en un esquema integrado de seguimiento y mejora*" [2] ("Balanced Scorecard: Measures that Drive Performance", 1992)

[1] El Diario del Centro del País. Periódico argentino cito en: http://anteriores.eldiariocba.com.ar/

[2] *"Balanced Scorecard: Measures that Drive Performance", Harvard Business Review.* **Kaplan, Robert y Norton, David. 1992.** Boston: s.n., 1992.

Cuba, guiada siempre por un proceso revolucionario excepcional, y contando por si fuera poco, con un líder único como Fidel Castro Ruz, no ha estado nunca ajena a los cambios que para bien se han producido en el mundo. Es por ello que hace algunas décadas comenzó toda una revolución digital en el país en pos de lograr una informatización, que a la postre, será el sustento del esfuerzo realizado. Surgen entonces diversos proyectos informáticos, dentro de los cuáles sobresalió uno de manera esplendorosa, forjado al calor de la Batalla de Ideas, da lugar el surgimiento de la Universidad de las Ciencias Informáticas (UCI), obra de quien será siempre el líder cubano.

La UCI, universidad de excelencia, que está concebida a ciencia cierta para formar Ingenieros en Ciencias Informáticas, hoy en día cuenta con más de 10 000 estudiantes y tres facultades regionales que extienden de sobremanera su ímpetu por todo el país.
En esta Universidad (UCI) la mayoría de los estudiantes están vinculados a proyectos productivos con alcance Nacional e Internacional, estos últimos mayoritariamente, aportando divisas a la economía del país. Pero existen también dentro de la UCI centros encargados de desarrollar y gestionar software para beneplácito del país propiamente dicho, como lo es el Centro de Consultoría de la UCI, en el cual existen proyectos productivos que dan soporte a distintas esferas.
Cuba, nunca ausente a los avances tecnológicos producidos en el mercado mundial, e integrando conocimientos propios con experiencias en la arena internacional, ha ido aplicando el uso de los CMI paulatinamente en algunas instituciones, en las que una vez creado, se han observado resultados satisfactorios.

El Centro de Consultoría UCI no cuenta con un sistema automatizado para gestionar los indicadores, por lo que realizan dicha gestión de los mismos de manera manual, propiciando que esta no sea la vía ideal debido al consumo de recursos humanos que desprende dicho proceso, a la lentitud con que es llevado a cabo, (debido a que se ejerce de manera manual) y a que se prescinde de un sistema capaz de almacenar el estado de estos indicadores; valdría la pena preguntarse entonces sí: ¿Se realiza de forma óptima el proceso de gestión de los indicadores en la consultora? ¿Está necesitada la consultora de un sistema que le permita realizar la gestión de los indicadores mediante una aplicación? ¿Qué beneficios traería un sistema automatizado?

Aún cuando estuviesen llevando a cabo un proceso de gestión de los indicadores adecuado en la consultora, es evidente que no es el óptimo, pues en la rama informática, el trabajo, cuando se realiza

de forma manual, consume demasiados recursos (entiéndase recursos humanos y materiales) para satisfacer la producción, así como para cumplimentar las metas u objetivos trazados. Es ineludible la automatización de los procesos en la rama informática, pues con ello:

- Se minimiza el gasto de recursos humanos y materiales.
- Los trabajadores pueden tener un mayor rendimiento pues les es más factible interactuar con aplicaciones informáticas que llevar en papeles la gestión de los indicadores.
- Se maximiza la elegancia, la fluidez así como la velocidad con que se llevan a cabo las tareas o trabajos.
- Permite arribar a conclusiones con mayor seguridad y certeza, pues la información visual gráfica, supone un mayor entendimiento y acercamiento a la situación.
- Se puede almacenar la información de una mejor manera mediante las Bases de Datos.

Por lo anteriormente expuesto se plantea como **problema científico** de la investigación**:** ¿Cómo mejorar el proceso de toma de decisiones estratégicas en el Centro de Consultoría?

Siendo el **objeto de estudio:** Sistemas de soporte a la toma de decisiones estratégicas.

Lo que produce como **campo de acción**: El Cuadro de Mando Integral.

Para dar solución al problema científico se propone como **objetivo general:** Desarrollar una aplicación que implemente un Cuadro de Mando Integral en el Centro de Consultoría de la Universidad de las Ciencias Informáticas.

Se tendrán como **objetivos específicos** los siguientes:

- Realizar el modelado del negocio y el análisis de los requisitos.
- Diseñar el software para el Cuadro de Mando Integral.
- Implementar el Cuadro de Mando Integral.

Para dar solución a estos objetivos específicos se llevarán a cabo las siguientes **tareas**:

- Investigación de las tendencias y tecnologías actuales en la aplicación del CMI.
- Análisis y selección de las tecnologías para desarrollar la aplicación.
- Modelación del negocio y las actividades para el flujo de trabajo de requerimientos.
- Realización de las actividades del flujo de trabajo de análisis y diseño.
- Realización de las actividades del flujo de trabajo de implementación.
- Despliegue de la solución informática propuesta.

La tesis quedará estructurada de la siguiente forma:

***Capítulo 1*: Fundamentación Teórica.** Se realiza la fundamentación teórica de la investigación donde se incluye el estado del arte, a nivel internacional, nacional y de la universidad; las técnicas, las tecnologías y el software que se utilizaron o que sirvieron de apoyo para la solución del problema. Además se abordan temas y conceptos relacionados con la investigación y la influencia de las Tecnologías de la Información y las Comunicaciones para el desarrollo del sistema.

***Capítulo 2*: Características del Sistema.** Comprende el funcionamiento del negocio. Se realiza modelo de negocio, haciendo un análisis de cada uno de los conceptos y entidades presentes en el negocio. Se muestran además los requisitos funcionales (RF) y los requisitos no funcionales (RNF) del sistema. Se describe su funcionamiento a través del diagrama de Casos de Uso del Sistema (DCUS) y las descripciones de los Casos de Uso (CU) para comprender mejor el funcionamiento de la aplicación que se diseñará.

***Capítulo 3*: Análisis y Diseño del Sistema.** En este capítulo se obtienen los artefactos del flujo de trabajo correspondiente: diagrama de clases del diseño, la realización de los CU entre otros, se realiza una descripción a los principales patrones de Diseño utilizados, de igual forma se muestra la arquitectura con la que se trabajó y el resto de los diagramas correspondientes al flujo de trabajo.

***Capítulo 4:* Implementación del Sistema.** En este capítulo se analizan los diagramas de componentes, se abordan los estándares de codificación a seguir y se ejemplifica con códigos de algunas clases el uso de los patrones de arquitectura y diseño. Se realizaron pruebas de aceptación.

Capítulo 1: Fundamentación Teórica de la Investigación

El presente capítulo brinda una descripción general del objeto de estudio y las tecnologías utilizadas por los CMI para su informatización. Se hace una descripción de las técnicas, metodologías y herramientas usadas para dar solución al problema científico.

1.1 ¿Qué es un Cuadro de Mando Integral?

En la actualidad, las empresas se enfrentan a un mercado cada vez más competitivo, que las ha obligado a cambiar sus sistemas de gestión para adaptarlos a las características cambiantes de dicho mercado. Como quiera que se hace inminente saber que:

El Cuadro de Mando Integral: Es un Sistema de Medición Estratégico que utiliza cuatro perspectivas para controlar la implementación de la estrategia en sus factores críticos de éxito y su adecuación al entorno. Estas perspectivas son: financiera, del cliente, de procesos internos y de formación y crecimiento. Todas éstas ligadas, contribuirían a controlar el cumplimiento de los objetivos estratégicos midiendo los factores claves y sus inductores de actuación para lograr un control anticipado y enfocado no solamente en la actuación financiera. Es la filosofía perfecta que permite a la dirección de una institución inducirla al éxito competitivo. Además este incluye un Dashboard (Tablero de Control).

Como definieran Norton y Kaplan en 1992 el CMI:

"Es un poderoso instrumento para medir el desempeño corporativo y se ha demostrado que es la herramienta más efectiva para enlazar la visión, misión y la estrategia. Además permite ofrecer una visión completa de la organización, siendo el elemento esencial del sistema de información que sirve de apoyo al sistema de control de gestión en su misión de mejorar su nivel de competitividad en el largo plazo" [3] ("Balanced Scorecard: Measures that Drive Performance", 1992)

[3] *"Balanced Scorecard: Measures that Drive Performance", Harvard Business Review.* **Kaplan, Robert y Norton, David. 1992.** Boston: s.n., 1992.

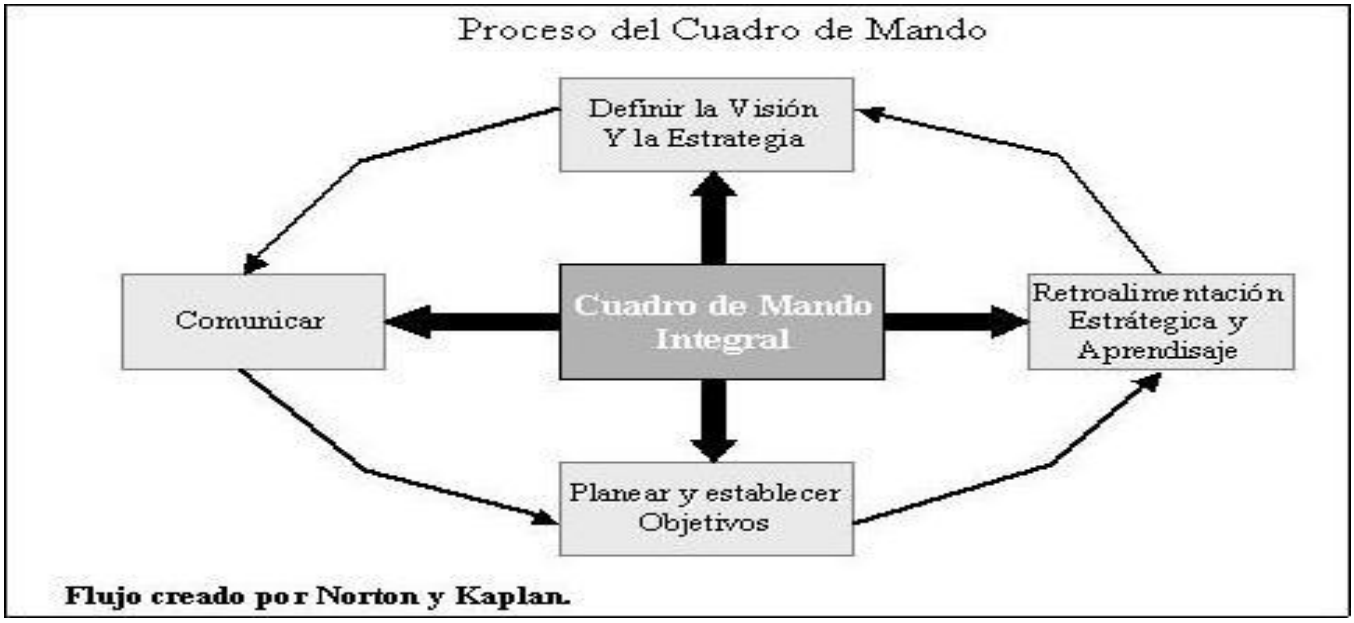


Figura 1 Representación de un CMI según Norton y Kaplan

1.2 Perspectivas de los CMI

El CMI mide la actuación de la organización desde cuatro perspectivas equilibradas: **las finanzas, los clientes, los procesos internos y la formación y crecimiento.**

Figura 2 Representación de las Perspectivas de un CMI

Perspectiva del Cliente o Consumidor: ¿qué esperan de la empresa?

El buen Servicio al cliente es muy importante y es la base para poder permanecer en un mercado competido. Los clientes esperan productos de óptima calidad, con un costo adecuado, que se entreguen a tiempo y que su rendimiento sea el convenido.

Perspectiva Interna o de Procesos Internos: ¿en qué podemos destacarnos?

¿Qué hacer dentro de la empresa para cumplir con las expectativas de los clientes? Los Procesos de la empresa deben estudiarse y evaluarse para conseguir la satisfacción de los consumidores.

Perspectiva de la innovación o Aprendizaje y Crecimiento: ¿Qué se debe continuar mejorando?

La competencia es feroz en este nuevo milenio, por ello la Empresa debe ser apta para innovar y mejorar. Los productos cumplen su ciclo de vida y es necesario disponer de unos nuevos, con capacidades mayores y atractivas.

Perspectiva Financiera: ¿Qué esperan los accionistas?

Quiénes invierten su dinero esperan, en forma legítima un rendimiento adecuado. Si esto no se complace, es probable que inviertan su dinero en una empresa diferente. Incluye además aquellos aspectos relacionados con los recursos humanos necesarios para poder implementar las mejoras en el resto de las perspectivas[4]. (El Balanced Scorecard ayudando a implantar la estrategia, marzo 2004)

1.3 Dónde Aplicar un CMI y cuáles son sus Objetivos

El CMI puede ser aplicado a empresas privadas, organizaciones no gubernamentales, entidades estatales e instituciones sin fines de lucro.

Debido a que el CMI agrupa los objetivos estratégicos en cuatro perspectivas: Financiera, Clientes, Procesos internos y, Aprendizaje y Crecimiento; todas las áreas de la organización son incluidas en la estrategia institucional, a través de objetivos estratégicos de las áreas financiera, comercial, producción, logística, RRHH, sistemas, infraestructura y otras.

Los objetivos del CMI son:

- Analizar, definir y operacionalizar la misión y visión.
- Traducir la estrategia a términos operativos.
- Alinear la organización y su estructura a la estrategia.
- Movilizar el cambio organizacional y hacer de la estrategia un proceso continuo y participativo.
- Introducir herramientas de medición, evaluación y control.

[4] **Fernández Alberto, El Balanced Scorecard ayudando a implantar la estrategia**, Revista de Antiguos Alumnos, marzo 2004, 42.

1.4 Origen del CMI

El BSC o CMI definido por Kaplan y Norton en 1992, supone la evolución del Cuadro de Mando tradicional **Tableau de bord** (traducido literalmente, vendría a significar algo así como tableros de mando o cuadro de instrumentos) puesto de moda en los años 60 en Francia, que aportaba información sobre aspectos concretos de una organización, pero que no nos permitía tener una visión global de la misma tal y como lo hace el CMI.

De hecho, la originalidad de esta herramienta no radica, precisamente, en la combinación de indicadores financieros y no financieros, pues durante la revolución de la Dirección Científica a principios del siglo XX, ingenieros en empresas innovadoras habían desarrollado tableros de control con la combinación de estos indicadores; por lo tanto "*es una idea con cien años de antigüedad*" (Dávila, 1999). La idea de combinarlos para realizar el seguimiento de los procesos estratégicos tiene casi la misma edad que el concepto de estrategia; es decir, unos 40 años. De manera que, lo novedoso del CMI es, exactamente, el modo como se seleccionan, determinan e interrelacionan los mencionados indicadores.

1.4.1 Valoraciones Especializadas acerca del CMI

El CMI utiliza un conjunto de indicadores que, a diferencia de los cuadros de mando tradicionales, están plenamente integrados y coordinados, a través de relaciones causa-efecto, con los objetivos y metas de la organización. Se trata de *"una herramienta de gestión que traduce la estrategia de la empresa en un conjunto coherente de indicadores"* (Dávila, 1999). Pero, al mismo tiempo, destacamos que el CMI, al ofrecer una visión global, obliga a que las organizaciones expliciten su modelo de negocio, definan su estrategia y arbitren nuevos sistemas de información.

En su origen, *"incorporaba en un único documento diversos ratios para el control financiero de la empresa. Esta herramienta fue evolucionando, con el paso de los años, y, actualmente, combina no sólo ratios financieros, sino también indicadores no financieros que permiten controlar los diferentes procesos del negocio, por lo que se le conoce entonces como CMI"* [5] (Dávila, 1999).

[5] **Dávila, Antonio. 1999.** *Nuevas herramientas de control: El Cuadro de Mando Integral.* 1999.

El CMI es una filosofía práctica de gerenciamiento y fue desarrollada en la Universidad de Harvard por los profesores Robert Kaplan y David Norton en 1992, su principal característica es que mide los factores financieros y no financieros del estado de resultados de la empresa.

Para Howard Rohm del Balanced Scorecard Institute de EE.UU, el CMI es *"un sistema de administración de desempeño que puede utilizarse en cualquier organización, grande o pequeña, para alinear la visión y misión con los requerimientos del cliente, las tareas diarias, administrar las estrategias del negocio, monitorear las mejoras en la eficiencia de las operaciones, crear capacidad organizacional, comunicando los progresos a todo el personal"* [6] (Rhom, 1992).

"La aportación que ha convertido al CMI en una de las herramientas más significadas de los últimos años es que se cimienta en un modelo de negocio. El éxito de su implantación radica en que el equipo de dirección dedique tiempo al desarrollo de su propio modelo de negocio". (Dávila, 1999) Ricardo Martínez Rivadeneira plantea que el CMI es una forma integrada, balanceada y estratégica de medir el progreso actual y suministrar la dirección futura de la compañía que le permitirá convertir la visión en acción, por medio de un conjunto coherente de indicadores agrupados en cuatro diferentes perspectivas, a través de las cuales es posible ver el negocio en conjunto.[7] (Martínez, 2001)

La implementación del CMI establece la combinación de indicadores financieros y no financieros, permitiendo controlar los diferentes procesos del negocio en una empresa; además es una filosofía práctica, que logra enlazar: visión, misión y estrategia con el fin de medir el desempeño corporativo; ayuda a los directivos de la empresa a trazar objetivos y métodos para obtener un mayor éxito competitivo; proporciona equilibrio a los distintos indicadores, manteniendo la integridad de objetivos y métodos establecidos.

1.5 ¿Cómo proceder para la obtención de un CMI?

Existen diferentes metodologías para la implantación de un sistema de medición del rendimiento basado en CMI.

[6] *A Balancing Act.* **Rhom, Howard. 1992.** 1992.

[7] **Martínez, Ricardo. 2001.** *Balancead Scorecard - Sistema de comunicación, control y aprendizaje estratégico.* 2001.

Entre las más conocidas, además de la de Kaplan y Norton (1996, 2000), se puede citar a (Ahn, 2001; Letza, 1996; Lohman et al., 2004; Papalexandris et al., 2005).

El proceso para la implantación del CMI consta de cuatro etapas:

Fase 1: Diagnostico Inicial.

- Factores Clave de Éxito.
- Cadena de generación de Valor.
- Proposición de Valor Organizacional
- Misión y Visión Organizacionales

Fase 2: Elaboración del CMI.

- Definición de Objetivos Estratégicos.
- Elaboración del Mapa Estratégico.
- Definición de Indicadores relevantes y medibles para cada objetivo.
- Establecimiento de metas para cada indicador estratégico (frecuencia y responsable de medición)

Fase 3: Construcción de Tableros Operativos.

- Tablero Operativo para la perspectiva de Aprendizaje y Crecimiento.
- Tablero Operativo para la perspectiva de Procesos Internos.
- Incluyen los programas, actividades y tareas requeridos para el logro de los objetivos estratégicos.

Fase 4: Sistematización del CMI.

- Introducción al software del conjunto de información resultante.
- Capacitación y entrenamiento para la administración y uso del software.

Notar en los anexos la figura 29 que muestra la metodología a seguir para generar un CMI

1.5.1 ¿Por qué el Cuadro de Mando Integral? ¿Cuáles son sus aportes y/o beneficios?

Valorando los resultados obtenidos a través de un CMI y siendo este evaluado mediante sus cuatro perspectivas se puede afirmar que desde el punto de vista:

Financiero: Ocurre un retorno de la inversión con valor agregado.

Cliente: Participación y posicionamiento en el mercado así como asegura la satisfacción, retención y fidelización del cliente.

Procesos Internos: Identificación de los procesos críticos (costo, calidad, oportunidad), creación de nuevos productos, mantiene altos los estándares de eficiencia con adecuado uso de tecnologías y sistemas de información, permite tener una visión integral del negocio.

Aprendizaje y crecimiento: Capacidad de la Empresa para aprender y crecer así como asegura un clima laboral y organizacional favorable, a partir de satisfacción del recurso humano como la principal fuerza impulsora de innovación y desarrollo. Produce de igual forma Competencias (talento, conocimientos y habilidades), Tecnología (sistemas, redes y bases de datos) y en el clima para la acción (identidad y compromiso).

Otros: Permite a los directivos saber si la mejora obtenida en un área de gestión se ha logrado a expensas de un empeoramiento en la gestión de otra área. Así mismo el CMI ofrece a la gestión una imagen gráfica y por tanto más clara de las operaciones relevantes del negocio, como también la metodología facilita la comunicación y entendimiento de los objetivos de la compañía en todos los niveles de la organización; ayuda a clarificar cómo las acciones del día a día afectan el corto y largo plazo.

Existen también algunas características que son importantes a la hora de valorar un CMI:

- La naturaleza de las informaciones recogidas en él, dando cierto privilegio a las secciones operativas, (ventas entre otras) para poder informar a las secciones de carácter financiero, siendo éstas últimas el producto resultante de las demás.
- La rapidez de ascenso de la información entre los distintos niveles de responsabilidad.
- La selección de los indicadores necesarios para la toma de decisiones, sobre todo en el menor número posible.

Se adoptaría una postura categórica si se afirmara que el CMI es el mejor de todos los modelos de medición estratégica para ser utilizado en una empresa, pero si podría decirse que es el CMI un modelo de medición estratégica que garantiza un gran número de ventajas a las Entidades, permitiéndoles alinear y apoyar los procesos claves, así como:

- Clarificar, actualizar y comunicar la estrategia a toda la organización.
- Alinear los objetivos personales y de los departamentos con la estrategia.
- Identificar y alinear las iniciativas estratégicas.
- Vincular los objetivos estratégicos con las metas a largo plazo y los presupuestos anuales.

1.5.2 ¿Por qué es importante un Sistema Integral de Control de Gestión?

- Es importante, ya que en una empresa se requiere un control integral de todas las funciones que se realizan en la misma y no solamente las financieras.
- Todo proceso de control tiene relación con asegurar que lo que se intenta hacer se cumpla. Incluye la definición de una meta, la medición del desempeño, la comparación de la meta con el resultado actual y las acciones correctivas.

1.5.3 Requisitos de funcionamiento de un Sistema de Control.

- Entendibles.
- Seguir la estructura de la Organización.
- Rápidos.
- Flexibles.
- Económicos.

1.5.4 Características de la solución CMI

Cuando se ofrece una solución a un problema de gestión de la información particular (aclarar que no existe un CMI estándar que solucione todo tipo de problemas en las empresas, pues cada uno se adecua a las necesidades de la Institución), se debe entregar un producto que básicamente cumpla con ser:

- **Adaptable**
  - ✓ Solución adaptable a la organización en cuestión.
- **Intuitivo**
  - ✓ Enfocada a usuarios no técnicos.
- **Versátil**
  - ✓ Implantación en cualquier ámbito de la empresa.
- **Integral**
  - ✓ Permite gestionar varias unidades de negocio.
- **Multiusuario**
  - ✓ Acceso por un número de usuarios ilimitados y concurrentes dentro de la organización.
- **Accesible**
  - ✓ Para usuarios no técnicos: facilidad de instalación y puesta en marcha inmediata.

1.5.5 Premisas fundamentales y factores de riesgo en los CMI

La implementación de un CMI versa acerca de aspectos fundamentales a tener en cuenta, para construir un sistema automatizado que de soporte a todas las definiciones realizadas en el diseño del mismo. La teoría del CMI no hace hincapié en ninguna tecnología en particular para su ejecución en las Organizaciones[8].

La mejor solución depende de cada organización en particular, la mayoría de las empresas confeccionan su CMI utilizando las herramientas de automatización de oficinas disponibles, como Microsoft Office y otras, usando planillas de cálculo y BD pequeñas, como por ejemplo MS Access.

Sin embargo las premisas fundamentales a las que apunta la utilización de un CMI son las siguientes:

- Debe ayudar en el proceso de definición de estrategias, objetivos, medidas, metas y acciones.
- Debe facilitar la comunicación de la dirección estratégica y ayudar a transmitir lo que debe hacer cada integrante de la organización, para que sus acciones individuales aporten al cumplimiento de estos objetivos.
- Debe permitir comparar la evolución de las metas y su cumplimiento (real vs planificado).
- Debe ser simple de entender y fácil de manejar para el usuario final, y fácil de mantener para los administradores.

Como se puede apreciar, en la medida que las herramientas no estén integradas a un único sistema informático, estas premisas se vuelven imposibles de cumplir y empiezan a transformarse en barreras para la implementación del CMI.

Algunos de los posibles factores de riesgo para el éxito de un programa de CMI son:

- **Falta de Compromiso de la Dirección**, es decir, la dirección no se involucra en el inicio del proceso, esto provoca que se delegue la responsabilidad en gerentes o mandos medios, lo que trae como consecuencia que el líder del proyecto pierda autoridad ante la empresa. La dirección de la organización es la primera que debe apoyar el programa con tareas específicas, como participar en las reuniones del lanzamiento y divulgarlo a toda la empresa.
- **Falta de Continuidad,** o sea, el CMI es una herramienta a largo plazo, no significa que sea una herramienta estática, pero se recomienda que se le hagan ajustes periódicamente para

[8] El Cuadro de Mando Integral como Herramienta de Gestión, **Norton y Kaplan 2000**.

mantener los lineamientos básicos y hacer comparaciones significativas de un momento a otro.

- **Sistema de comunicación deficiente**, sino existe un mecanismo por donde fluya la información, tanto para alimentar el CMI y distribuir sus resultados a todas las áreas, las personas no verán los beneficios que brinda el sistema, y verán al CMI como un intento de control deficiente y no como una herramienta para el desarrollo de todos los integrantes de la organización.
- **Definiciones Débiles**, si en el momento de definir los indicadores no existe un lenguaje unificado y un objetivo específico, esto provocará dobles interpretaciones por parte del personal y trae como consecuencia que cada persona haga su propia interpretación y genere controversias, y el programa empiece a perder confiabilidad.
- **Problemas en la Escalabilidad,** en este caso, si el CMI está compuesto por la unión de varios CMI para cada unidad de negocios, entonces esto provocará que se vaya complicando más la integridad de la información, y la actualización de la misma, propiciando que si no se tiene un sistema que permita la conexión de muchos usuarios, se presentarán inconvenientes en la distribución.

1.6 ¿Qué se obtiene con la implantación de un CMI?

Una de las máximas para una empresa que implementa un CMI es dotarse de una herramienta inteligente que permita gestionar todo el volumen de datos e informaciones que genera esta nueva filosofía de alineación estratégica. Perfecciona el funcionamiento del proceso de toma de decisiones estratégicas de una organización así como su rentabilidad, satisfacción y lealtad del cliente. Por lo general las empresas se enfocan en reducir los costos y gastos, mientras que al implementar el CMI se centran más en la rentabilidad y por ende incluye el concepto de reducción, la gran diferencia es que la energía se orienta a crecer y todos en la Organización actúan con base a esta motivación.

Una eficaz implantación del CMI supondría:

- **Sistema de Gestión orientado a resultados:**

  El CMI ayuda a alinear los objetivos de los empleados con los de la Organización. Favoreciendo la implantación de un sistema de retribución variable coherente con la estrategia.
- **Delimitación de Activos Intangibles como inductores de valor:**

Los sistemas de gestión tradicionales no son capaces de medir los mecanismos de generación de valor, como la excelencia en los procesos de producción, el saber hacer, la fidelización de clientes o la capacidad de la organización para implantar eficazmente su estrategia. El Cuadro de mando integral proporciona indicadores y relaciones de causalidad que delimitan y cuantifican esos intangibles, cada día más presentes en el valor de mercado de las empresas.

- **Enlaza modelos de planificación y de gestión:**
  La Empresa prioriza acciones e inversiones en función de sus objetivos. El modelo de gestión operativo debe guardar correlación con la estrategia para que las iniciativas sean consistentes.

1.6.1 Qué beneficios traerá consigo la aplicación en cuestión al Centro de Consultoría

El Centro de Consultoría de la UCI una vez cuente con el CMI, estará dotado de una herramienta inteligente que le permitirá gestionar los indicadores de forma automatizada, manejar de forma eficiente la información y tener estricto control sobre los indicadores y su estado, todo ello de forma gráfica gracias al Dashboard implementado, así como las informaciones que se generarán mediante esta nueva filosofía de alineación estratégica y que permitirán luego a la dirección del centro llevar a cabo un mejor proceso de toma de decisiones estratégicas. Podrá seguir el estado de sus indicadores de forma visual, lo que supondrá una mayor comprensión de su funcionamiento, permitiendo así conocer el estado de la producción y la eficiencia.

Gracias a esta herramienta será posible visualizar la información referente al estado de los indicadores en una única pantalla, permitirá almacenar la información, guardar plantillas de informes con la periodicidad deseada, distribuir, compartir e interactuar con otros usuarios la misma, al tiempo que se permite la autentificación de usuarios.

1.7 El CMI en el Mundo

Las empresas que en el mundo han implantado un CMI tienen hoy una economía mucho más favorable, cuentan con personal más capacitado en el trabajo y mayor cantidad de clientes, ya que midiendo su rendimiento de manera sostenida son mucho más eficientes, saben en cuál o cuáles indicadores hacer énfasis una vez observado el rendimiento de estos y suelen ser más competitivas en un mundo bastante convulso.

Como ejemplo de Software de CMI en el mundo se puede citar a:

Delphos: (Costa Rica)

Delphos es un administrador de indicadores de gestión, conocido normalmente como Cuadro de Mando Integral o Balanced Scorecard. Está especialmente diseñado para gerentes a efecto de facilitar el monitoreo y seguimiento del cumplimiento de los objetivos estratégicos definidos por la organización[9]. (Deinsa)

Delphos almacena toda la información necesaria para administrar un plan de desarrollo, ya sea Nacional, Estatal, Municipal o de cualquier otra índole. También puede administrar todo lo relacionado con Planes Anuales Operativos (PAO o POA).

Delphos permite cargar toda la estructura del modelo ya sea de manera completa o por niveles. Para más información contacte el sitio web (S.A)

Nombre Software: Delphos
Versión: 350.10.0
Empresa: Deinsa
País: Costa Rica
Página Web: www.deinsa.com
Pantalla: Consola Principal

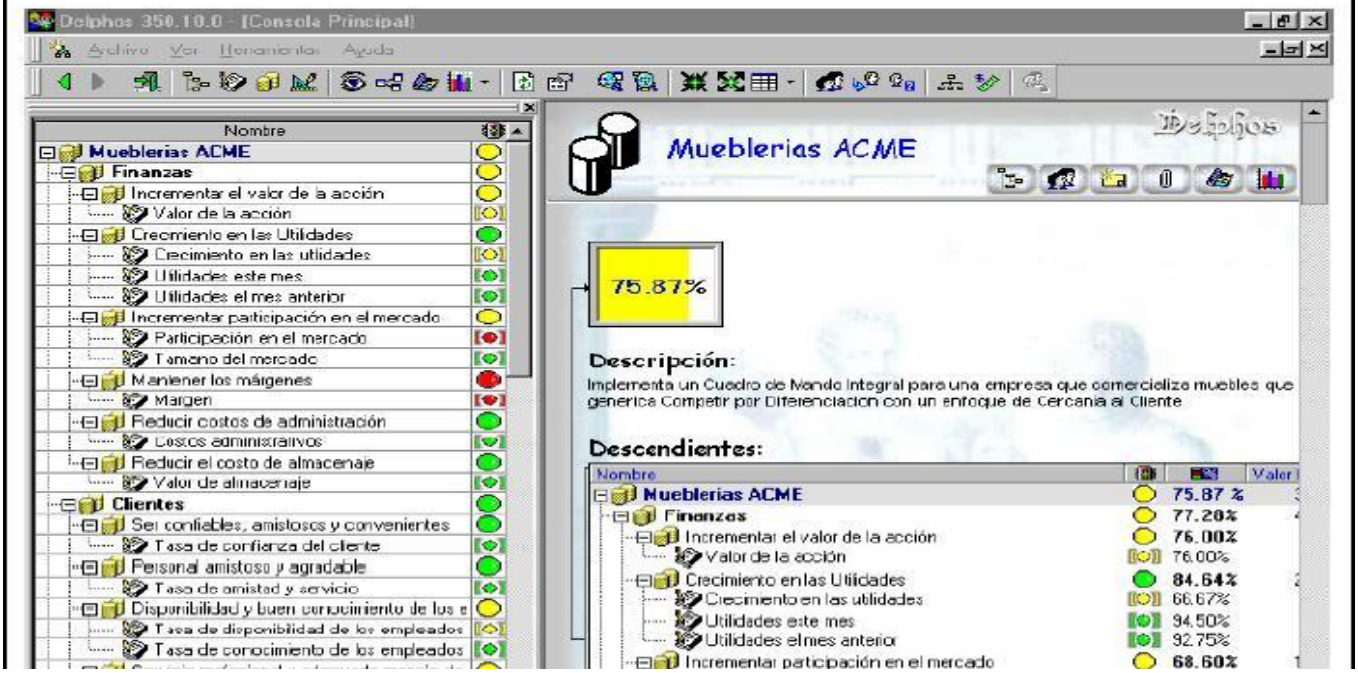


Figura 3 Consola del Software Delphos

[9] Cito en (http://www.deinsa.com/delphos.html).

Otro software que utiliza CMI es **Sixtina:** (Argentina)

Con este programa se podrá crear un modelo de negocios y crear los indicadores, métricas y reportes necesarios para aumentar drásticamente los resultados económicos de su empresa[10]. (Sixtina)

Dentro de sus beneficios mencionar que:

1. Soluciona las deficiencias en la información que impiden que la empresa haga un uso profundo de los datos registrados. Para ello Sixtina Dashboard permite:
   - Incorporar información no registrada por los sistemas tradicionales o ya establecidos.
   - Traer información ubicada en distintas localidades que se encuentra desconectada entre ella y los sistemas centrales.
   - Integrar información registrada en diferentes formatos de bases de datos, como Oracle, SQL, Database, Informix, Access, Txt, Excel.
2. Provee información en tiempo real, en forma instantánea con la posibilidad de emitir Reportes inmediatos. Visualización potente en base a un amplio menú de gráficos y grillas de datos.
3. Uso de filtros para acotar información que permita un mejor análisis de las causas del comportamiento de las variables de negocios.
4. Ayuda a toda la organización en aspectos específicos como, por ejemplo:
   - Acelerar la respuesta a problemas emergentes antes de que se conviertan en mayores perjuicios para la empresa.
   - Asignar los recursos apropiados para responder a pedidos de clientes en tiempo y forma.
5. Proveer un mejor control de costos, con renovados criterios de monitoreo interactivo con la cadena interna y externa de consumo de costos de la empresa.

1.8 El CMI en Cuba

La situación actual del país y los cambios producidos a nivel internacional ha traído consigo que las empresas cubanas tomen ciertas posturas para adaptarse a las nuevas transformaciones y tendencias ocurridas desde el punto de vista de la tecnología. Para medir la calidad del desempeño y la eficiencia para mejorar los resultados en las finanzas, muchas de las empresas toman como normas de calidad las ISO 9000 obteniendo méritos satisfactorios en su producción.

[10] Para más información contactar el sitio http://www.sixtina.com.ar/

Cuba no se encuentra de forma alguna ajena al avance tecnológico y científico, lo cual se manifiesta en la Resolución Económica del V Congreso del Partido de Cuba:

"El empleo de técnicas modernas de dirección empresarial, adecuadas a nuestras características y basadas en las mejores y más avanzadas prácticas contemporáneas"[11] (Resolución Económica del V Congreso del Partido Comunista de Cuba, 1998)

De esta forma se ha demostrado que en Cuba existe debilidad a la hora de definir alcance, objetivos, estrategia, visión y misión de las empresas para lograr un mayor rendimiento; reconociendo así mismo la necesidad de un cambio que de solución a esta dificultad. El CMI es una opción viable para lograr un rendimiento deseado en las empresas, contribuyendo a fortalecer en las mismas el proceso de toma de decisiones estratégicas.

Una de las pioneras en experimentar el trabajo con un CMI fue GIGET y la Dirección de Centro Oeste de INTERMAR.SA de Cienfuegos, donde se efectuó el primer encuentro de empresarios cubanos para debatir temas de BSC, superando las expectativas, pues en el semestre próximo se promovieron encuentros de ocho empresas de Ciudad de la Habana y Cienfuegos. A finales del 2003 la cantidad de empresas trabajando en la implementación del CMI no pasaban de cinco, pero muchas se incorporarían a partir del 2004.

Algunas de las empresas cubanas que han implementado CMI son:

- INTERMAR CIENFUEGOS.
- El Centro de Estudios Contables, Financieros y de Seguros (CECOFIS).
- Servicios Especializados de Protección .SA (SEPSA) en Cienfuegos.
- Cubapetróleo (CUPET).
- Universidad de Cienfuegos.
- Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba (CITMA).
- Empresa de Telecomunicaciones de Cuba .SA (ETECSA).
- Servicios Técnicos de Defectoscopía y Soldadura (CENEX).
- Empresa Cubana Nacional de Software (DESOFT).

[11] *Resolución Económica del V Congreso del Partido Comunista de Cuba.* 1998.

1.8.1 Antecedentes en el desarrollo de los CMI en la UCI

En la UCI se han realizado varios estudios acerca de los CMI, los que han quedado plasmados en los diversos trabajos de diploma. Se puede afirmar que en los trabajos antecedentes no se ha obtenido como producto final un software. A continuación se hace referencia a dichos trabajos:

Módulo Gestión de estrategias e indicadores del Cuadro de Mando Integral. [Millet Y. & Piloto Y.]

Módulo Gestión de reportes y la estructura de una organización del Cuadro de Mando Integral. [Burgos Y. & Azán Y.]

Módulo Gestión de los Objetivos y los Responsables de un Cuadro de Mando Integral" [Estrada S. & Vásquez A.]

Módulo de Gestión de Plan de Acción y Sistema de Alerta para las perspectivas del Cuadro de Mando Integral. [Almaguer Y. & Cala Y.]

Propuesta de procedimiento para el desarrollo de un Cuadro de Mando Integral para la gestionar proyectos informáticos. [Mendoza Y. & Carballosa O.]

Desarrollo del módulo de Análisis Estratégico para la implementación de un Cuadro de Mando Integral para gestionar proyectos informáticos en la Universidad de las Ciencias Informáticas. [Capote L. & Ruiz R.]

CMI-Soft: "Herramienta para la gestión estratégica de los procesos productivos en la Facultad 6. Análisis y el Diseño". [González A. Silvente Y]

Existen diversos software de CMI en el mundo, y en su mayoría son herramientas eficientes para cada empresa en cuestión como tal, lo cual permite que estas realicen mejor su proceso de negocio y sean más competitivas; se citó como ejemplo a dos software de CMI que actualmente son usados en el mundo.

No es menos cierto que indistintamente de las funcionalidades que dichos software permiten a las empresas que los utilizan, estos en su mayoría son privativos, con altos costos en sus licencias de uso, y que generalmente poseen una serie de RNF que no permiten a países menos desarrollados hacer uso de estas herramientas, además de otras restricciones, como ser aplicaciones de escritorio y no web, que es el tipo de aplicaciones más usado en la actualidad y que fue el tipo de aplicación solicitado por el cliente. Además no se cuenta con el código fuente de estos software.

Por lo antes expuesto y debido a que la UCI no cuenta con un CMI propio implementado, es que se decide proceder con la realización de este trabajo, con el principal objetivo de satisfacer las

necesidades del Centro de Consultoría UCI y de proporcionar a la vez una herramienta totalmente realizada en esta Universidad y que respondiera a los intereses de la misma.

1.9 ¿Qué es un Dashboard?

Se podría decir que el nombre **Dashboard** se refiere básicamente al tablero de un automóvil, el cual ofrece al conductor información permanente sobre el estado del vehículo. Sin embargo el mundo de los negocios toma la palabra con un sentido similar pero en lugar de aplicarlo a los automóviles lo refiere a la empresa, por lo que se puede afirmar que es una interfaz visual.

Así Dashboard es una página desarrollada en base a tecnología web mediante la cual se despliega en tiempo real información de la empresa, extraída de varias fuentes o bases de datos. Su característica de tiempo real otorga a los usuarios un conocimiento completo sobre la marcha de la empresa y permite hacer análisis instantáneos e inteligencia de negocios.

1.10 ¿Por qué utilizar Dashboard en esta aplicación?

Lo que marca la diferencia una vez se utiliza una herramienta informática para los negocios es el uso inteligente de la información. Aún teniendo montañas de información sobre sus clientes, procesos, proyectos y otros tipos de datos el Centro de Consultoría no se hace por ello más eficiente.

El aspecto central radica en cómo hacer un uso inteligente sobre datos e información, sobre operaciones, estado de los indicadores entre otros. Es decir cómo transformar datos dispersos en información estratégica.

Cuando las personas piensan en un reporte “tradicional”, con seguridad imaginan filas y columnas de papeles interminables donde los datos se ordenan por categorías, pero que rara vez vienen complementados con cuadros y gráficos. Ese método ya quedó obsoleto y está siendo reemplazado por una herramienta nueva mucho más intuitiva y productiva: el Dashboard.

Se trata de una solución que ofrece gran atractivo visual, elaboración y análisis "en tiempo real". Y permite a los ejecutivos explorar la información de manera mucho más profunda, lo que concluye en mejores decisiones desde el punto de vista estratégico.

Además de las ventajas visuales, el Dashboard permite la exploración de los datos. Es decir, si un usuario desea más información acerca de un indicador especial, él puede profundizar la investigación obteniendo datos primarios a través de la misma herramienta. También existe la posibilidad de crear escenarios especiales del tipo “que pasa si” ("what if"). Este análisis consiste en lograr una

interacción entre datos fijos y variables que conduzcan a representar situaciones posibles y evaluar resultados en caso de que las hipótesis se cumplan.

¿Cómo? Gracias al intenso despliegue visual de los gráficos de barras y líneas utilizados. Los datos provenientes de distintas fuentes se pueden "integrar" y consolidar de tal manera que muestren el concepto que se necesita conocer, su pasado y sus tendencias.

1.10.1 Principios a tener en cuenta a la hora de diseñar el Dashboard

Para diseñar el Dashboard son varios los parámetros que se deben seguir, a continuación se citan algunos:

- **Respecto a la disposición de la página:**

Menos es más: se tiene una capacidad limitada para analizar información. Es necesario hacer foco sobre lo importante. De manera que no es conveniente tener demasiadas vistas por página.

- **Regla de la mano:**

Continuando con lo anterior, se limita el número de informes por página a tres.

- **No usar scroll de forma excesiva:**

Esto es válido tanto para un Dashboard como para una página web. El usuario avanzado no hará demasiado scroll.

- **Disposición en pantalla:**

Existen patrones de atención ineludibles para crear un Dashboard tales como:

Arriba a la Izquierda: según estudios realizados se afirma que este será el primer punto de atención**.** Por lo que se convierte en el lugar natural dónde disponer la información más importante.

Centro: es este el segundo punto de atención, por lo que será el segundo lugar natural dónde disponer de la información restante, la más importante.

Arriba Derecha: parte neutral.

Abajo a la Derecha: por lo general aquí se coloca la información menos relevante pues es el lugar de la pantalla que menos llama la atención a los usuarios.

- **Usar menús fijos:**

Reducir la cantidad de puntos de navegación por página es esencial para evitar la confusión y la sobre información. Concentrarse en la página principal dónde los usuarios centran la mayor parte de su atención.

- **Usar componentes gráficas y destacarlas en caso de que se utilicen:**

Destacar los links con un color significativo (por ejemplo, el azul).

- **Respecto al contenido:**

No todos los gráficos tienen sentido para todo tipo de datos (tener bien claro esto).

Aquí es válido destacar que cada grupo de datos tiene su propia identificación con ciertas gráficas, y que por otra parte al mismo grupo de datos le resta sentido una gráfica menos acorde con dicha información. Por ejemplo supongamos que debemos graficar la velocidad de un automóvil en movimiento, eso claro está podría hacerse con un gráfico de barras o bien con un gráfico de líneas, pero tanto en uno como en otro caso sería menos visible para los usuarios que un gráfico de reloj, puesto que con este último el usuario no requiere fijar tanto su atención en los altibajos de velocidad producidos, lo que se traduce en lo contrario si se utilizara alguno de los gráficos anteriormente señalados (barra y líneas).

Usar fuentes claras para la lectura, así como la misma familia de colores para todos los gráficos. Una vez en el tema de los colores es importante no basarse sólo en ellos, dado que hay usuarios que no ven bien todos los colores. Los iconos son menos claros que el texto, el cual debe de estar alineado.

A la hora de acentuar el texto debe hacerse con fondos del color opuesto.

1.11 Fundamentación de las Herramientas y las tecnologías a utilizar.

1.11.1 Metodologías de desarrollo de software.

El principal objetivo de las metodologías de desarrollo es elevar la calidad del software a través de una mayor transparencia y control sobre el proceso. La competencia del mercado, y el contar con la información en tiempo y forma, requiere ser seguros, precisos y confiables creando la necesidad de trabajar con aplicaciones de software cada vez mejores, con mayor cantidad de prestaciones y que permitan a su vez un eficiente desarrollo y mantenimiento. Existen estándares y procesos de desarrollo de software que determinan buenas prácticas para el posterior desarrollo de las aplicaciones. El siguiente estudio fue realizado con las metodologías más utilizadas.

Proceso Unificado de Rational (RUP, Rational Unified Process): RUP es una infraestructura flexible de desarrollo de software que ha sido probada y a su vez proporciona prácticas recomendadas así como una arquitectura configurable[12] (Ivar Jacobson, 2000). Unido al Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis,

[12] **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software. .* s.l.: Pesaron Educación, 2000.

implementación y documentación de sistemas orientados a objetos (OO). RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado. No es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Es una forma disciplinada de asignar tareas y responsabilidades (quién (trabajador) hace qué (artefacto), cuándo (flujo de actividades) y cómo (actividades)).

Extreme Programming (XP):

La Programación Extrema (XP) es una metodología de desarrollo de software ligera (o ágil) basada en una serie de valores y de prácticas que posibilitan comunicación, realimentación y reutilización del código con el cual se trabaja, además tiene como objetivo aumentar la productividad a la hora de desarrollar programas. XP procura la sencillez en el software.

"La Programación Extrema asume que la planificación nunca será perfecta, y que variará en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de feedback que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar"[13] (Kent, 2000)

En este trabajo se utilizó la metodología RUP porque cubre todo el ciclo de vida de desarrollo del software y constituye la metodología estándar más utilizada mundialmente para el análisis, implementación y documentación de sistemas OO, lo que cumple al máximo con las expectativas de este trabajo. Además presenta características que le permite crear modelos iterativos e incrementales posibilitando la adaptación a los diferentes cambios que se podrían producir en el sistema, aplicable de igual manera en este trabajo. RUP elimina los errores cometidos en las iteraciones previas, lo cual permite desarrollar un proceso con calidad, para eso se agrupan las actividades en grupos lógicos definiéndose flujos de trabajo principales divididos en cuatro fases (inicio, elaboración, construcción y transición). Los seis primeros flujos son conocidos como flujos de ingeniería y los tres últimos como de apoyo, donde todos y cada uno de ellos cobran vital importancia en el proceso de desarrollo de software.

[13] **Kent Beck**, **2000** "*Programación Extrema*".

1.11.2 Herramientas Case para el desarrollo de software.

Existen varias herramientas case cada una con sus peculiaridades, pero esta vez se expondrán las que a juicio de muchos son las principales exponentes. **Notar** figura 30 en los anexos)

Rational Rose brinda soporte al Lenguaje de Modelado Unificado (UML). Es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Ofrece un lenguaje de modelado común que agiliza la creación del software. Es un software propietario.

Visual Paradigm por su parte es una herramienta CASE que utiliza UML como lenguaje de modelado. Visual Paradigm-UML soporta los últimos estándares de anotaciones de JAVA y UML, provee soporte para la generación de código y la ingeniería inversa para Java. Además, se integra con Eclipse, Borland, JBuilder, NetBeans IDE/Sun ONE, IntelliJ IDEA, Oracle JDeveloper y BEA WebLogic Workshop para soportar las fases de implementación en el desarrollo de software. Tiene dentro de sus características principales que es multiplataforma, portable y posee gran facilidad de uso.

Debido a que Rational Rose es una herramienta CASE de software propietario y según lo planteado anteriormente, se decidió utilizar el Visual Paradigm, ya que es una herramienta de fácil integración con JAVA que será el lenguaje a utilizar para desarrollar el sistema y como se dijo anteriormente soporta los últimos estándares de anotaciones de JAVA y UML, así como provee soporte para la generación de código y la ingeniería inversa para dicho lenguaje; permite de igual forma realizar todos los diagramas y artefactos que se generan a lo largo del desarrollo del software; además es multiplataforma y a pesar de ser software propietario la UCI posee la licencia que hace legal su uso.

1.11.3 Herramientas CASE para el modelado de base datos

ERwin es una herramienta para el diseño de BD que brinda productividad al mismo, así como generación y mantenimiento de aplicaciones, desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la BD diseñada. Además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la BD. Genera automáticamente las tablas y miles de líneas de *procedimientos almacenados (stored procedure) y triggers* para los principales tipos de BD.

ERwin hace fácil el diseño de una BD. Los diseñadores de BD sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes.

La migración automática garantiza la integridad referencial de la base de datos. ERwin establece una conexión entre una base de datos diseñada y una base de datos, permitiendo transferencia entre ambas y la aplicación de ingeniería reversa. Usando esta conexión, ERwin genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto y restricciones de campos y dominios.

ERwin soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen Oracle, Microsoft SQL Server, Sybase. El mismo modelo puede ser usado para generar múltiples bases de datos, o convertir una aplicación de una plataforma de base de datos a otra.

EasyCASE Profesional es el centro de productos para procesos, modelamiento de datos y eventos, e Ingeniería de Base de Datos, es un producto para la generación de esquemas de base de datos e ingeniería reversa, trabaja además para proveer una solución comprensible para el diseño, consistencia y documentación del sistema en conjunto.

Esta herramienta permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, para poder crear las aplicaciones eficazmente desde el procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real.

Oracle Designer es un conjunto de herramientas para guardar las definiciones que necesita el usuario y automatizar la construcción rápida de aplicaciones cliente/servidor gráficas. Integrado con Oracle Developer, Oracle Designer, que provee una solución para desarrollar sistemas empresariales de segunda generación.

Todos los datos ingresados por cualquier herramienta de Oracle Designer, en cualquier fase de desarrollo, se guardan en un repositorio central, habilitando el trabajo fácil del equipo y la dirección del proyecto.

En el lado del Servidor, Oracle Designer soporta la definición, generación y captura de diseño de varios tipos de bases de datos por conexión de Oracle, pero ninguno es compatible con el SGBD Postgres.

Luego de analizar las herramientas CASE para el modelado de la Base de Datos se llegó a la conclusión que se debía utilizar ERwin, pues además de ser una de las más conocidas en nuestra Universidad esta asegura consistencia, rehúso, e integración de los datos al proporcionar el bosquejo que las Tecnologías de la Información (IT) necesitan para entender, analizar y comunicar la estructura de la base de datos, además de mejorar la productividad entre los desarrolladores cuando los diseños de la BD son divididos, compartidos, y reutilizados..

1.11.4 Lenguaje de Modelado

El UML es en la actualidad el lenguaje de modelado de sistemas de software más conocido y utilizado. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software[14]. (Craig, 1999)

No se encontró competencia a la hora de seleccionar a UML ya que dentro de sus características más relevantes se encuentra que:

Permite modelar sistemas utilizando técnicas OO. Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.

Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).

- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones y demás).Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Es muy expresivo, ya que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.
- Es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental [15] (Craig, 1999)

[14] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

[15] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

1.11.5 Lenguajes de Programación.

Uno de los aspectos importantes a tener en cuenta en un lenguaje de programación es la portabilidad. Poder usar la misma aplicación en distintas arquitecturas o sistemas operativos sin tener que recompilar supone un gran ahorro de desarrollo. He aquí la característica más importante de **Java**, esto se debe a que no es el sistema operativo quien ejecuta directamente un programa, sino la máquina virtual.

Por otra parte el código de **C#** se ejecuta en un entorno de ejecución manejado, lo que supone el paso tecnológico más importante para hacer que pueda ejecutarse en distintos SO. Sin embargo algunas de las bibliotecas de .NET se basan en Windows, particularmente la biblioteca WinForms, que depende de algunos detalles fundamentales de la aplicación.

Phyton es uno de los lenguajes de script más frecuentemente empleados, se destaca por estar orientado a objetos de principio a fin. Su sintaxis emplea tabuladores para marcar bloques de código, destaca por la claridad y legibilidad de sus programas. Contiene una estructura mínima, ya que todo el lenguaje está desarrollado a partir de componentes básicos, los cuales también pueden ser modificados.

1.11.6 ¿Por qué utilizar Java como lenguaje de Programación?

Se utiliza Java como lenguaje de programación ya que la aplicación a realizar será Web y este lenguaje destaca notablemente por su uso con este tipo de aplicaciones, además por ser un lenguaje de objetos, aquí es importante destacar que los lenguajes OO son lenguajes modificados para trabajar con objetos, pero Java es creado para trabajar con objetos desde cero, de hecho todo en Java tiene que ver con objetos; por ser independiente de la plataforma, lo que permitirá correr la aplicación en varios Sistemas Operativos (SO); en cuanto a la herencia, se pueden crear nuevas clases que hereden de otras preexistentes, lo que simplifica la programación, implicando que se pueda reutilizar mucho código; por otra parte Java brinda múltiples prestaciones y servicios así como presenta un gran número de características que lo sitúan por delante de muchos otros lenguajes, dentro de estas podemos señalar:

Arquitectura neutral, portátil y robusta: Es neutral, al adoptar un sistema de código binario que es independiente de arquitecturas hardware, SO y sistemas de ventanas. Es portátil, al definir de forma

precisa los tipos y tamaños de los datos. Y es robusta, al poseer un chequeo del código tanto en tiempo de compilación como de ejecución.

Simple: Posee las estructuras mínimas de un lenguaje de programación tradicional, sin añadir ninguna estructura más.

Independiente de la plataforma: Java se compila a un formato de código de byte que puede ser leído e interpretado por muchas plataformas, incluyendo Windows 95, Windows NT, Solaris 2.3, GNU\Linux, Mac OS entre otros.

Seguro: El código de Java puede ser ejecutado en un entorno que prohíbe la introducción de virus, borrar y modificar ficheros o la ejecución de operaciones que provoquen la caída del ordenador y la pérdida de datos.

Multitarea (Capacidad multihilo): En Java, todo transcurre de forma paralela, con varias tareas de forma simultánea. Un único programa Java puede procesar diferentes tareas de forma independiente y continua. Permite la creación de aplicaciones distribuidas.

Además Java ha sido mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores: Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.

- Crear programas para que funcionen en un navegador web y en servicios web así como combinar aplicaciones o servicios que usan el lenguaje para crear servicios o aplicaciones totalmente personalizadas.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML entre otros.

1.11.7 Servidor Web

Internet Information Server: Es una serie de servicios para los ordenadores que funcionan con Windows. Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que lo tienen instalado se pueden publicar páginas web tanto local como de manera remota. Este servidor constituye software propietario, por lo que no es muy viable, además no es multiplataforma.

Cherokee: Es un servidor web libre, multiplataforma, abierto bajo la licencia GPL. Apunta a ser un servidor web bastante rápido que también soporta las funcionalidades más comunes de servidor.

Cherokee puede ejecutar Interfaz de Entrada Común (CGI), Hypertext Preprocessor (PHP) tanto como PHP-CGI o FastCGI. También soporta registro y autenticación de usuarios, puede además realizar redirecciones y soporta la configuración de Servidores Virtuales.

Apache Tomcat: el servidor de aplicaciones Tomcat de Apache y las tecnologías afines proporcionan a los programadores de Java un completo conjunto de herramientas para crear de forma rápida sofisticadas aplicaciones web[16], es un servidor web (http) y constituye uno de los más completos contenedores de Servlet gratuito, que programadores de Servlet de Java o Páginas de JavaServer (JSP) utilizan con frecuencia para probar su código. Tomcat persigue además la total compatibilidad con las versiones de Servlet y las especificaciones de API (Application Programming Interface o Interfaz de Programación de Aplicación) JSP que admite. Sin embargo, se trata de algo más que un servidor de pruebas, dado que muchas empresas lo emplean en la actualidad en entornos de producción debido a su contrastada estabilidad. Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual de Java. Por otra parte es fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java.

Luego de analizar varios servidores web, se decidió utilizar el Apache Tomcat debido a toda una serie de características expuestas anteriormente, pero lo principal es su integración al lenguaje de programación con Java y que brinda mucha comodidad a la hora de trabajar. Es además una de las mejores tecnologías en lo que a servidores web respecta.

1.11.8 Framework

El Framework es un una especie de programa, componente, código en general que posibilita un asequible desarrollo de aplicaciones de software relativamente estandarizadas (o al menos una parte o partes de éstas) lo que proporciona el diseño y en ocasiones algunos componentes auxiliares. De igual forma son soluciones que implementan patrones y contribuyen al desarrollo de funciones dentro de las distintas aplicaciones a nivel superior. Como función adicional abstraen implementaciones complejas y producen un desarrollo ágil, proveyendo soluciones a problemas cotidianos; se puede ver también como un conjunto de librerías que usan las aplicaciones.

[16] **Programación Java Server con J2EE Edición 1.3**

A partir de que la aplicación a desarrollar será basada en la Web, y que el **Spring** es un framework modular que cuenta con una arquitectura dividida en siete capas o módulos (Spring Web MVC, Spring ORM, Spring Context, Spring AOP, Spring Core, Spring Web y Spring DAO (Data Access Object)) se pudo utilizar la capa de Spring Web MVC, ya que esta destaca por ser bastante flexible y altamente configurable, hace una clara división entre controladores, modelos de JavaBeans y vistas y está basada en interfaces para enmarcar mejor las funcionalidades; los controladores son configurados a través de las Inversiones de Control (IoC), que es una de las funcionalidades más importantes del Spring y se encarga de separar del código de la aplicación que se está desarrollando, los aspectos de configuración y especificaciones de dependencia del framework. Es el Spring un framework de aplicación, desarrollado para aplicaciones escritas en el lenguaje Java, es además catalogado como liviano o ligero pues no es una aplicación que requiera de muchos recursos para su ejecución, por otra parte puede ser distribuido completo en un archivo .jar de alrededor de un Megabyte lo cual representa muy poco tamaño para la cantidad de servicios que ofrece. Es importante destacar que es una aplicación Open Source, lo que implica que no tiene ningún costo ni se necesita de una licencia para utilizarlo y tiene disponible su código fuente en el paquete de instalación.
Spring agrupa además un conjunto de Frameworks propios que hacen posible un mejor desarrollo de la aplicación como lo son:

- *Acegi:* para todo lo relacionado con la seguridad.
- *Spring MVC:* para lo referente al desarrollo Web.
- *Spring:* Frameworks de inyección de dependencias.
- *Spring-JDBC:* framework de persistencia.

Debido a que el Spring agrupa en un solo Framework la combinación de herramientas tales como: *Enterprise JavaBeans (EJB), Servlets y Java Server Pages (JSP),* para el desarrollo de aplicaciones *Java 2 Enterprise Editions* (J2EE), y gracias a su total vinculación al lenguaje de programación utilizado para desarrollar el Trabajo de Diploma y a que en este los objetos que se gestionan no necesitan implementar o extender funcionalidades especiales, por cuanto es un mecanismo totalmente transparente, así como que además, los servicios que brinda Spring son fundamentales en cualquier aplicación y contribuyen de manera directa con el programador (evitándole código), lo que supone un aumento de la producción en los proyectos que lo utilizan es que se decide su utilización para este trabajo.

Figura 4 Arquitectura de Spring en Capas

1.11.9 Hibernate

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.

La principal diferencia entre una aplicación que utiliza código tradicional JDBC (Java Database Connectivity) y otra que lo hace utilizando código que se escribe con Hibernate es que desaparece el SQL pues es precisamente esto de lo que se ocupa Hibernate, básicamente enlaza el mundo objetual con el relacional, o lo que es igual se encarga de transformar nuestra consulta al dialecto SQL que toque. Además Hibernate sabe deducir el tipo de datos que estamos utilizando a partir de la introspección así como analiza en tiempo de ejecución las clases Java y obtiene la información que necesita.

Hibernate soporta actualmente los dialectos siguientes y muchos más que no son objetivo de este trabajo:

- *PostgreSQL* con su clase net.sf.hibernate.dialect.PostgreSQLDialect
- *Oracle* (cualquier versión) con su clase net.sf.hibernate.dialect.OracleDialect
- *MySQL* con su clase net.sf.hibernate.dialect.MySQLDialect

Hibernate para enlazar el modelo objetual con el relacional solo necesita saber:

- Quién está detrás del modelo relacional
  - ✓ Qué Gestor de Base de Datos está detrás.

- ✓ A qué BD se realizará la conexión.
- ✓ Cómo se hará la conexión.

- Cómo se emparejan propiedades y campos de tablas:
  - ✓ Clave primaria (que propiedad de la tabla se corresponde con la clave primaria).
  - ✓ Cómo se gestiona las relaciones entre tablas (uno-uno, muchos-muchos y otras).

Para responder a ello se utilizan dos archivos distintos:

- El **archivo de propiedades de Hibernate** (`Hibernate.properties`) que es el encargado de determinar los aspectos relacionados con el gestor de bases de datos y las conexiones con él.
- Los **archivos que definen el emparejamiento** (*mapping*) de propiedades con tablas y columnas (`*.hbm.xml`).

En este trabajo lo que se definió con el *Hibernate.properties* fue el gestor de base de datos utilizado (PostgreSQL), a que BD como tal se realizó la conexión y cómo se hizo:

hibernate.connection.url=jdbc:postgresql://10.34.2.107:5432/cmi_final
hibernate.connection.driver_class=org.postgresql.Driver
hibernate.connection.username=postgres
hibernate.connection.password=postgres
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

Mientras que con el *mapping* (NombreDeClase.hbm.xml) se describió cómo se relacionaron las clases y tablas y las propiedades y columnas, a continuación un ejemplo sencillo:

<class name="modelo.Indicador" table="indicadores" schema="public">
<id name="idindicador" type="int">
<column name="idindicador" />
<generator class="increment" />

1.12 Entorno de Desarrollo Integrado (IDE)

Un IDE es una aplicación que reúne varios programas necesarios para el desarrollador en este caso de Java: editor, compilador, depurador y otros más.

Eclipse es un IDE creado íntegramente en Java (lenguaje de programación utilizado en esta Tesis), aunque en la actualidad este entorno lo reconocen otros lenguajes como PHP sigue siendo muy útil

para aplicaciones en Java; se utilizó este IDE para facilitar el trabajo debido a que el desarrollo ya iba manejando un buen número de clases y con este entorno se logró mucha más versatilidad para depurar el programa, puesto que tiene un debbugers mucho más avanzado que supone mayor traceabilidad al código. Eclipse permite agregar pluggins lo que le da un extra a la aplicación a desarrollar, además consume menos recursos que otro de los IDE utilizados para las aplicaciones Web como es el **NetBeans**; desde Eclipse, IDE para Java EE Developers se pudo programar diseñando y trabajando con cada módulo de forma independiente. Para desarrollar cada uno de los aspectos del diseño se dispuso de un completísimo paquete de funciones, utilidades y herramientas que facilitaron todo el trabajo.

Existen otros IDE que tienen también integración con Java pero se desecharon por varias razones, por ejemplo como se dijo anteriormente el **NetBeans** supone mucho gasto de recurso de Random Access Memory (RAM) para la PC; por otra parte otros como **Borland JBuilder**, **Oracle JDeveloper** y demás están también presentes, pero en su mayoría o bien no son Open Source ni gratuitos, o no se integran con Java o tienen alguna que otra debilidad que lo hace menos fuerte que Eclipse.

1.13 Sistema Gestor de Base de Datos (SGBD)

Una Base de Datos (en lo adelante BD) es un conjunto de información que se puede acceder por medios informáticos y sobre el que se puede realizar diferentes acciones (insertar nuevos datos, selección de datos, actualización y el borrado de registros, combinación con otras bases de datos, generación de informes impresos).

Un Sistema de Gestión de Bases de Datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos[17] (A. Silberschatz, 2006)

Los SGBD ofrecen un control centralizado de la información, entre sus principales objetivos se encuentran:

- Evitar la redundancia de los datos.

[17] Silberschatz, A., H.F. Korth, and S. Susarshan, *FUNDAMENTOS DE BASES DE DATOS*, ed. McGraw-Hill. 2006. 994.

- Mejorar los mecanismos de seguridad y privacidad de los datos.
- Mantener la integridad de los datos realizando las validaciones necesarias.
- Mejorar la eficacia de acceso a los datos.

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Este gestor de Bases de Datos es muy versátil, es multiplataforma y posee un amplio ANSI SQL 99 y otras extensiones, además soporta funcionalidades como procedimientos almacenados, triggers, vistas actualizables, SSL entre otros.

Oracle: Es una herramienta cliente/servidor para la gestión de bases de datos. Es un producto altamente vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio provocan que generalmente se utilice en grandes empresas y multinacionales. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos como: Access, MySQL, SQL Server y otros. Tiene el inconveniente que es un software privado.

PostgreSQL: Está considerado como la base de datos de código abierto más avanzada del mundo y es un SGBD objeto-relacional (ORDBMS), o sea es similar a un front-end dentro de una BD relacional que permite que los datos sean grabados como objetos, sin embargo la BD puede ser accedida también como una BD relacional; pero lo más importante del modelo objeto-relacional es la implementación de los User-Defineg-Types (UDTs) y de los User-Defineg-Funtions (UDFs) que son nuevos tipos de datos y nuevas funciones personalizadas por el usuario. PostgreSQL proporciona persistencia a los datos, gestiona BD de gran tamaño, soporta usuarios concurrentes y es capaz de recuperarse de fallos de Software y Hardware con gran facilidad, así como también proporciona una forma simple de consultar los datos; características estas que serán todas aplicables para el desarrollo de este trabajo, permitiendo así obtener un producto de mayor calidad. Es por otra parte un SGBD de fácil integración con Java, lenguaje utilizado para desarrollar la Tesis, además de ello este permite hacer una copia de seguridad de datos con la BD en funcionamiento. PostgreSQL no es sólo un sistema de BD por lotes, sino que también funciona como recurso en línea con aplicaciones web.

Debido a que MySQL no posee *integridad referencial* y suele ser lento con grandes BD, a que Oracle no es libre, y el precio de su licencia es excesivamente elevado y contando todas las características a su favor que tiene PostgreSQL es que se decidió utilizarlo para esta Tesis.

1.14 Con qué se puede graficar. Librería Utilizada

Una tarea sumamente difícil para cualquier programador Web es llevar a la práctica mediante gráficas los objetivos y lineamientos cumplidos cuando programa, para ello existen diversas librerías que han sido de una u otra forma puestas al alcance de estos para que puedan realizar su trabajo.

Java en su forma estándar no viene con nada que nos permita hacer gráficos cómodamente, pero existen muchas **librerías externas** que pueden ser usadas con este fin, por ejemplo:

Java 3D de Sun que es gratuita puede ser utilizada para dibujo 3D.

JFreeChart, gratuita, cuenta con gran variedad de tipos de gráfico (tartas, histogramas, de puntos entre otros).

JMSL (Numerical Library for Java) ofrece una colección de clases matemáticas, estadísticas, financieras, gráficas y de minería de datos disponible para Java y además combina gráficas integradas con funciones matemáticas y estadísticas. Acotar que es una muy buena librería pero tiene que ver más directamente con gráficas integradas a las funcionalidades matemáticas y estadísticas de los algoritmos.

Para la realización de este trabajo, específicamente para graficar haciendo uso del Dashboard, se utilizó la librería **JFreeChart.** Esta librería tiene como función principal mostrar los gráficos referentes a las aplicaciones, es totalmente software libre, distribuida bajo licencias GPL y cuenta con un gran número de tipos de gráficos, lo que posibilita escoger mejor el tipo de gráfico que se utilizará, proporcionando una mayor comprensión de acuerdo a los escenarios propuestos; está considerada como una de las mejores librerías de dibujo de gráficas que existe. Tiene un diseño bastante flexible y de fácil entendimiento, lo que supone que con un conocimiento no muy avanzado de la herramienta se pueden llegar a obtener resultados deseados, para su utilización solamente es necesario bajar las librerías **JFreeChart** y **JCommon** y luego proceder a su instalación**.**

1.14.1 Sobre la utilización de la Librería JFreeChart

Cada uno de los posibles gráficos que se pueden obtener con *JFreeChart*, necesita unos datos que son los que se pintarán en dicho gráfico. Estos datos por lo general no se pasan directamente al gráfico, sino que se pasan a una clase encargada de almacenarlos, que se suele conocer como modelo de datos (o dataset en el lenguaje de *JFreeChart*). En *JFreeChart* hay disponibles muchos posibles modelos de datos o *datasets*, según el tipo de gráfico que queramos pintar. Por ejemplo

vamos a centrarnos en *CategoryDataset*, que nos permitirá pintar gráficos de barras (*BarChart*) y de líneas (*LineChart*).

CategoryDataset es sólo una interface, por lo que se debe usar alguna clase concreta que la implemente. *JFreeChart* brinda la posibilidad de hacer una clase propia, o elegir usar alguna de las muchas que ofrece *JFreeChart.*

Para mayor comprensión un ejemplo de cómo quedan las gráficas de *BarChart y LineChart* en una comparación de visitas a dos sitios web (Ver figuras en los anexos).

1.15 Roles y Artefactos de RUP

Un **rol** define el comportamiento y responsabilidades de un individuo[18] (Ivar Jacobson, 2000). Generalmente son realizados por una o varias personas, trabajando como un equipo, donde un grupo de proyecto satisface diferentes roles. Los **artefactos** constituyen un producto de trabajo usado en el proceso, son elementos de información producidos y usados por los trabajadores para realizar nuevas actividades. Para logar el objetivo general y con él cada uno de los objetivos específicos se desarrollarán los roles y artefactos que propone RUP. Los roles por cada uno de los flujos de trabajos asociados al alcance de la investigación son:

Modelado del Negocio

- *Analista de Procesos del Negocio:* Es el responsable de definir la arquitectura del negocio; los casos de uso del negocio y los actores del negocio.
- *Diseñador del Negocio:* Es el encargado de detallar la organización y especificar el flujo de trabajo de los CU del Negocio.
- *Revisor Técnico:* Revisa formalmente el modelo de casos de uso del negocio y de objetos de negocio obtenido.
- *Stakeholder:* Personas u organizaciones que están activamente implicadas en el negocio.

Requerimientos

- *Especificador de Requerimientos:* Es el encargado de especificar y detallar las funcionalidades del sistema, definiendo varios aspectos de los requerimientos y de agrupar los casos de uso por paquetes en caso de que proceda.

[18] **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* s.l.: Pesaron Educación, 2000.

- *Diseñador de interfaz de usuario:* Es el que elabora el diseño de la interfaz de usuario abarcando en la misma los requerimientos de forma tal que sean cumplidos. No se encarga de implementar dicha interfaz, solo de diseñarla.

Análisis y Diseño

- *Arquitecto:* Es el responsable de la arquitectura del software, incluye las principales decisiones técnicas que limitan el diseño global de la ejecución del proyecto.
- *Diseñador:* Es el encargado de diseñar una parte del sistema, dentro de las limitaciones de los requerimientos, así como la arquitectura y los procesos de desarrollo del proyecto, identifica y define las responsabilidades, operaciones y relaciones entre los elementos del diseño. Asegura que el diseño sea consistente con la arquitectura del software.

Implementación

- *Implementador:* Es el encargado de implementar el sistema.
- *Arquitecto de software:* Tiene como responsabilidad realizar la parte restante de la arquitectura de software para dar fin al cumplimiento de esta en el proyecto.

Los artefactos generados en cada una de las fases son:

En el modelado del negocio:

- Descripción de los casos de uso del negocio.
- Diagrama de caso de uso del modelo de negocio.

En los requerimientos:

- Especificación de requerimientos del Software.
- Actores del Sistema.
- Casos de uso del Sistema.
- Modelo de casos de uso del Sistema.
- Especificación de casos de uso del Sistema.
- Prototipos de interfaz no funcionales.

En el Análisis y Diseño:

- Realización de casos de uso del Sistema.
- Clases del diseño.
- Modelo de Datos.
- Modelo de Diseño.
- Paquetes del Diseño.

A continuación se ilustran algunos de los artefactos que propone RUP para los flujos de trabajo antes mencionados, circulados aquellos que se generaran en cada uno de dichos flujos:

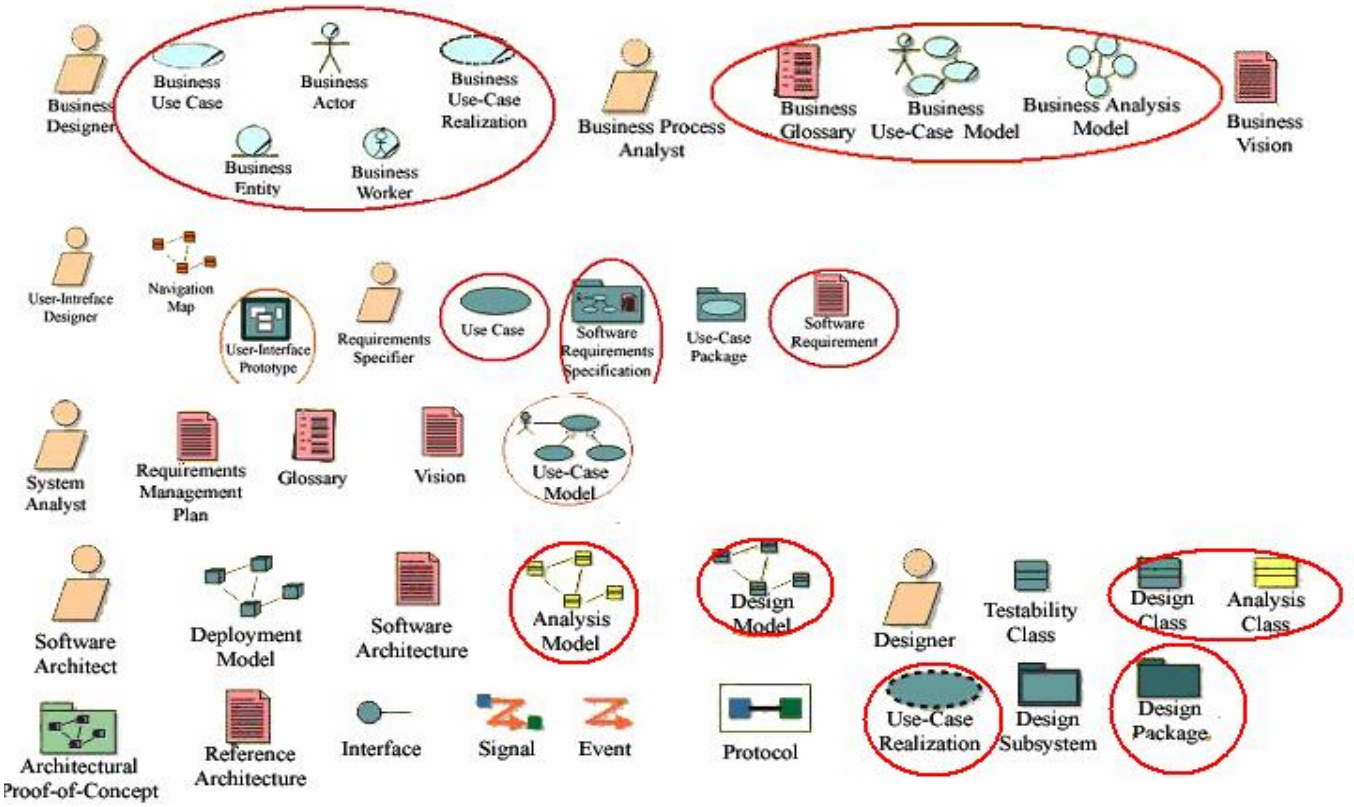


Figura 5 Roles y Artefactos generados en los diferentes Flujos de Trabajo.

En el flujo de trabajo Implementación:

- Subsistema de implementación.
- Diagrama de despliegue.
- Diagrama de compontes.
- Código fuente.

1.16 Patrones

Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación[19] (Craig, 1999).

[19] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos*"

Antes de saber que son los Patrones de Casos de Uso se hace necesario saber que es un **Caso de Uso** por sí solo para tener mayor comprensión del tema luego, según definiera Craig Larman un Caso de Uso es: *"una descripción de un proceso de principio a fin relativamente amplia, descripción que suele abarcar muchos pasos o transacciones; normalmente no es un paso ni una actividad individual del proceso"* [20] (Craig, 1999).

1.16.1 Patrones de Casos de Uso

Un patrón es una solución aplicable a un problema que se presenta una y otra vez en el desarrollo de distintas aplicaciones y en distintos contextos. Los principales patrones conocidos o más utilizados en los casos de usos son:

- **Patrón CRUD:** (Creating, Reading, Updating and Deleting) consiste en un caso de uso llamado "Información CRUD" o "Administrar Información". Modela todas las operaciones que se pueden realizar sobre una parte de información, tal como crearla, visualizarla, buscarla, actualizarla y eliminarla.
- **Actores múltiples:** Rol común (Multiple Actors: Common Role) se tienen dos actores que juegan el mismo papel hacia el caso de uso. Este rol es representado por otro actor, heredado por los actores que comparten este rol.

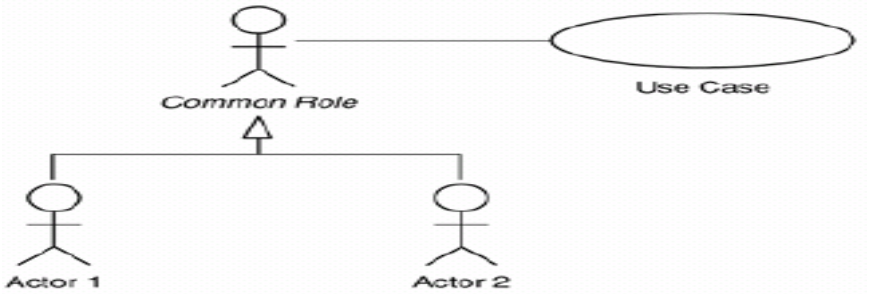


Figura 6 Representación de Actores Múltiples (Rol Común)

1.16.2 Patrones de Diseño.

Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una abstracción

[20] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

de una solución en un alto nivel, es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces[21] (Craig, 1999).

Patrones GoF (Creacionales)

- El patrón **Abstract Factory** (fabricación abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- **Singleton** (instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones GoF (Estructurales)

- **Adapter** (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- **Decorator** (Envoltorio): Añade funcionalidad a una clase dinámicamente.

Patrones GoF (Comportamiento)

- **Chain of Responsibility** (Cadena de responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- **Command** (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- **Observer** (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

Patrones GRASP: Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos y no son más que parejas de problema solución con un nombre, que codifican buenos principios y sugerencias relacionadas con la asignación de responsabilidades[22] (Craig, 1999). Se refieren a cuestiones y aspectos fundamentales del diseño, como son:

[21] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

[22] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

- **Experto:** La interrogante de este patrón radica en cuál es el principio fundamental en virtud de la asignación de responsabilidades en el diseño orientado a objeto. A la hora de asignar responsabilidades en un modelo de clases es fundamental hacerlo de forma adecuada pues el sistema sería más fácil de entender, mantener y ampliar además de que permitiría reutilizar los componentes en futuras aplicaciones. Por lo que el patrón propone como respuesta:
  - ✓ Asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- **Creador:** Este patrón se enfoca en solucionar la problemática de quien debería ser responsable de crear una nueva instancia de alguna clase. En un sistema OO una actividad muy frecuente es la creación de objetos, por lo que se hace necesario controlar la asignación de responsabilidades a través de un principio general. Por lo que el patrón propone como solución:
- Agregar a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos: B agrega los objetos de A, B contiene los objetos de A, B registra las instancias de los objetos de A, B utiliza específicamente los objetos de A, B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado.
- **Alta Cohesión:** Con este patrón se pretende solucionar el dilema de cómo mantener la complejidad dentro de los límites manejables. Específicamente la cohesión funcional es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Por lo que el patrón propone como solución:
  - ✓ Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- **Bajo Acoplamiento:** Este patrón radica en solucionar la problemática de cómo dar soporte a una dependencia escasa y a un aumento de la reutilización. El acoplamiento mide la fortaleza de la conexión entre una clase y otras. Es decir que una clase con bajo (o débil) acoplamiento no depende de muchas otras. Por lo que el patrón plantea como solución:
  - ✓ Asignar una responsabilidad para mantener bajo acoplamiento.
- **Controlador:** Este patrón surge ante la inconsistencia de quién debería encargarse de atender un evento del sistema, y propone como solución:
  - ✓ Asignar la responsabilidad del manejo de los eventos de un sistema a una clase, permitiendo mayor mantenimiento y reutilización.

1.16.3 Patrones de Arquitectura

Los patrones de arquitectura son patrones que se desarrollan en la fase de diseño inicial y son relacionados a la interacción de objetos dentro o entre niveles arquitectónicos. Este patrón resuelve problemas arquitectónicos, adaptabilidad a requerimientos cambiantes así como de acoplamiento[23] (Craig, 1999).

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Los patrones de arquitectura representan el nivel más alto en el sistema de patrones. Ayudan a especificar la estructura fundamental de una aplicación. Cada actividad de desarrollo es gobernada por esta estructura; por ejemplo, el diseño detallado de los subsistemas, la comunicación y colaboración entre diferentes partes del sistema. Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global.[24] (Wellicki, 2007)

Para la realización de este trabajo se utilizó el patrón **Modelo Vista Controlador (MVC)**, este es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web (como la de este trabajo), donde la *vista* es la página HTML y el código que provee de datos dinámicos a la página. El *modelo* es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el *controlador* es el responsable de recibir los eventos de entrada desde la vista.

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información (o sea se encarga de mostrar todo lo referente a visualizar la información).

---

[23] **Craig Larman UML y Patrones:** "*Introducción al Análisis y Diseño Orientado a Objetos"*

[24] Welicki, L. *Patrones y Antipatrones: una Introducción* 2007 [cited 2007 diciembre]; Available from: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317/default.aspx#M15

Controlador: Controla el flujo entre la vista y el modelo (los datos). Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

Básicamente el flujo que sigue el MVC es:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón)
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. *Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.*
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

1.16.4 Cómo se relaciona el patrón MVC con el framework Spring

Como la mayoría de los framework que implementan el patrón **MVC**, Spring tiene implementado un Servlet que realiza las tareas de Front Controller, esto significa que cada uno de los request que son realizados por el usuario, pasa a través de este Servlet. El nombre que recibe este Servlet es

Dispatcher Servlet. Y como se indicó anteriormente, todos los request que son realizados por los distintos usuarios pasan por este componente.

En la figura que a continuación se muestra la imagen indica que mediante el Request se llega al Dispatcher y este es responsable de delegar a otro componente el procesamiento del request. Para obtener el nombre del componente que recibirá el request, Spring utiliza lo que se denomina *Handler Mapping*, este tiene por función determinar cuál será el Controller que recibirá el request.

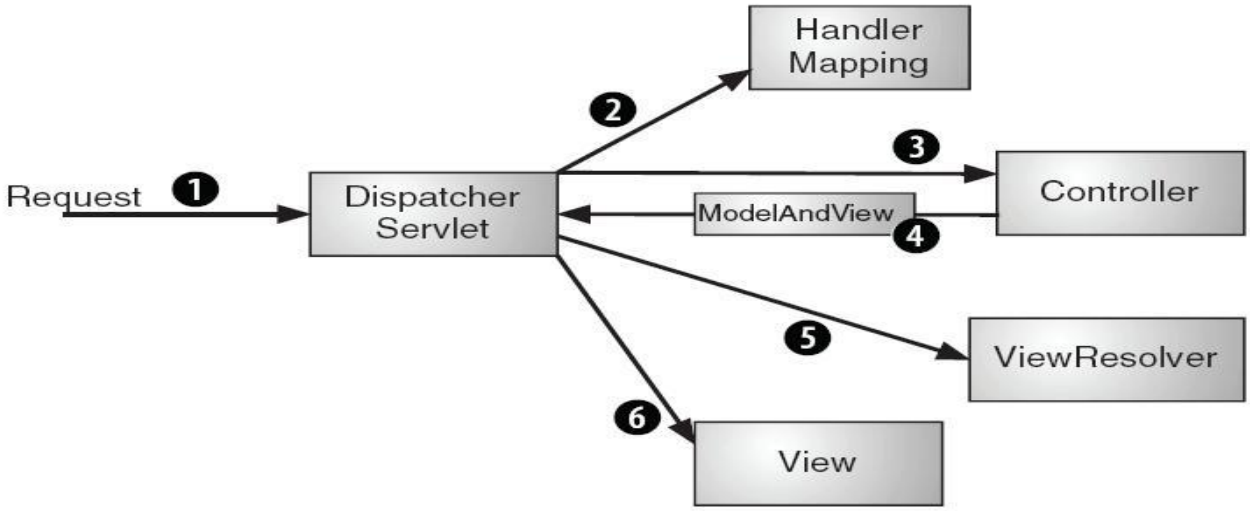


Figura 7 Muestra de cómo se desarrolla el MVC

El Handler Mapping como se dijo tiene el objetivo de indicar al Dispatcher cuál será el componente que debe recibir el request enviado por el usuario. Por lo cual el Dispatcher Servlet, le pregunta a uno o más Handler Mapping cuál será el Controller que recibirá el Request. Dentro del Spring existen varios Handler Mapping los cuáles tienen distintas capacidades de poder mapear a los controladores.

Ejemplo de esto es el ***SimpleUrlHandlerMapping*** el mismo *mapea* controladores a URL basándose en una colección de propiedades que se definen en el Spring Application Context; así como el ***ControllerClassNameHandlerMapping***, también este *mapea* controladores a URL utilizando el Controller Class Name.

1.17 Conclusiones del Capítulo

En el presente Capítulo se realizó un estudio al estado del arte concerniente al CMI. Se analizaron además las tecnologías, metodologías y herramientas a utilizar con el fin de dar solución a la problemática planteada, de donde se concluyó que se utilizarán aquellas que por alguna u otra razón se consideraron las más indicadas para el desarrollo de la aplicación, teniendo en cuenta diversos factores y tratando de cumplir al máximo las exigencias requeridas.

Capítulo 2: Características del Sistema

En este capítulo se realiza una descripción del objeto de estudio. Se utiliza el modelo de negocio para describir los procesos del negocio, se definen las funcionalidades del software a través de los requisitos funcionales y no funcionales; los actores y el diagrama de casos de usos del sistema, finalmente se realiza la descripción de los casos de usos.

2.1 Descripción del Problema a resolver

La UCI cuenta con un Centro de Consultoría en el cual existen numerosos proyectos productivos, a los cuáles no se les está controlando de forma eficiente su rendimiento debido a que se realiza el mismo de forma manual, por lo que se hace engorroso dicho proceso y se requiere de mucho tiempo para ello. Todo esto crea entonces la posibilidad de una mala gestión de sus indicadores, pues puede ocurrir pérdida de información una vez se esté llevando a cabo dicho proceso, citando también el retraso en cuanto a tiempo que ocasiona que se realice de forma manual. Por otra parte la dirección del Centro de Consultoría no cuenta con todos los datos necesarios y presentados de forma adecuada para proceder a una toma de decisiones correcta y acertada de la manera más rápida posible debido a que no puede mantenerse contacto grafico visual del estado de los indicadores en tiempo real.

2.2 Solución Propuesta

Se propone la realización de una aplicación Web que posibilite mejorar el proceso de toma de decisiones estratégicas en el Centro de Consultoría, así como poder gestionar los indicadores mediante las perspectivas, mostrar el rendimiento de forma gráfica a través de un Dashboard del estado actual de los indicadores y mantener un control adecuado de los mismos. Dicha aplicación deberá permitir adicionar, modificar, mostrar y eliminar una serie de datos enmarcados en diversas funcionalidades, necesarias para de esta forma llevar a cabo un mejor proceso de gestión de la información en el Centro de Consultoría, así como permitir la autenticación de los usuarios.

2.3 Modelo del Negocio

Un **modelo de negocio** (también llamado diseño de negocio) es el mecanismo por el cual un negocio trata de generar ingresos y beneficios. Implica tanto el concepto de estrategia como el de implementación. Es un resumen de cómo una compañía planifica servir a sus clientes lo cual se logra

definiendo los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos. El Modelamiento del negocio tiene como objetivos:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Derivar los requerimientos del sistema que va a soportar la organización.
- Lograr una comunicación efectiva entre los usuarios y el equipo de proyecto con el objetivo de llegar a un entendimiento de lo que hay que hacer, es la clave del éxito en la producción de un software.

En pos de lograr un mayor entendimiento del negocio se utilizó la técnica de la entrevista, entrevistando a quién sería el cliente, lo que supuso una mejor comprensión del problema, pues atendiendo sus solicitudes se lograría un producto con mayor calidad.

2.3.1 Reglas del Negocio

- El CMI tiene cuatro perspectivas.
- Cada indicador pertenece a una perspectiva (que puede pertenecer a un Proyecto o al Centro de Consultoría).
- Los indicadores pertenecen a una y solo una perspectiva.
- Los indicadores pertenecen al Centro de Consultoría o a los proyectos.
- Cada indicador podrá ser medido solamente según lo que se indique en su frecuencia de medición.

2.3.2 Actor del Negocio

Cuando se modela un actor del negocio se modela al rol que juega este al interactuar con el negocio y beneficiarse de sus resultados. Es un individuo o grupo de individuos, organización, máquina o sistema de información externo al negocio y que interactúa con él.

Tabla 1 Actores del Negocio

Actor	Descripción
Directivo General	Es el responsable de organizar y hacer que se lleven a cabo cabalmente las actividades en el negocio.

2.3.3 Trabajadores del Negocio

El trabajador del negocio define el rol de un individuo o grupo de individuos, máquina o sistema automatizado que participa directamente en los procesos que se llevan a cabo en el negocio, realizan

las actividades y son propietarios de elementos, pero no se benefician de forma alguna con los resultados del proceso.

Tabla 2 Trabajadores del Negocio

Trabajador	Descripción
Responsable de Área	Encargado de verificar la existencia o no de los indicadores definidos así como su medición para hacer una verificación de su estado en el área definida.
Responsable de Estrategia	Encargado de verificar el estado actual de los indicadores así como de emitirlo.

2.3.4 Diagrama de Caso de Uso del negocio

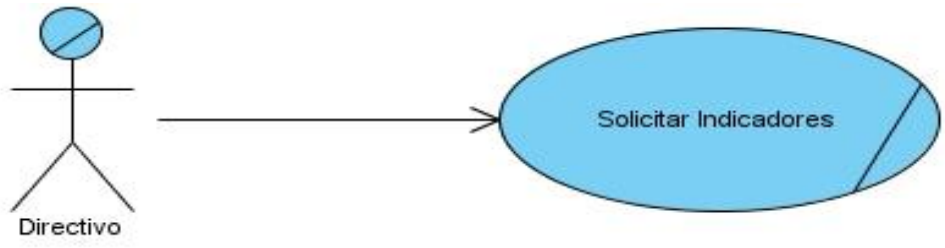


Figura 8 Diagrama del Caso de Uso del Negocio.

2.3.5 Realización de Caso de Uso del Negocio

El objetivo principal que se persigue cuando se detallan los casos de uso del negocio es la descripción de su flujo de sucesos, es tratar de saberlo todo, desde como comienza hasta cómo termina este, inmiscuyendo la forma en que interactúa con los actores, lo que supondrá una mayor comprensión de los procesos de negocio para con los clientes y desarrolladores.

Se describen en detalle las actividades dentro del negocio y quiénes son los encargados de realizarlas. Por otra parte, gracias a los diagramas de actividades se puede mostrar el flujo de los procesos de forma gráfica. Mediante las calles, es posible especificar las responsabilidades de los actores y trabajadores del negocio y a través del flujo de objetos cómo es que se utilizan las entidades del negocio.

2.4 Diagrama de Actividades

El diagrama de actividades (DA) es un grafo que contiene estados en los que puede hallarse una actividad y muestra una serie de actividades que deben ser realizadas para lograr los objetivos del negocio. El mismo se divide en calles donde cada una de ella representa el actor y trabajador que está llevando a cabo la actividad y muestra la utilización de las entidades del negocio.

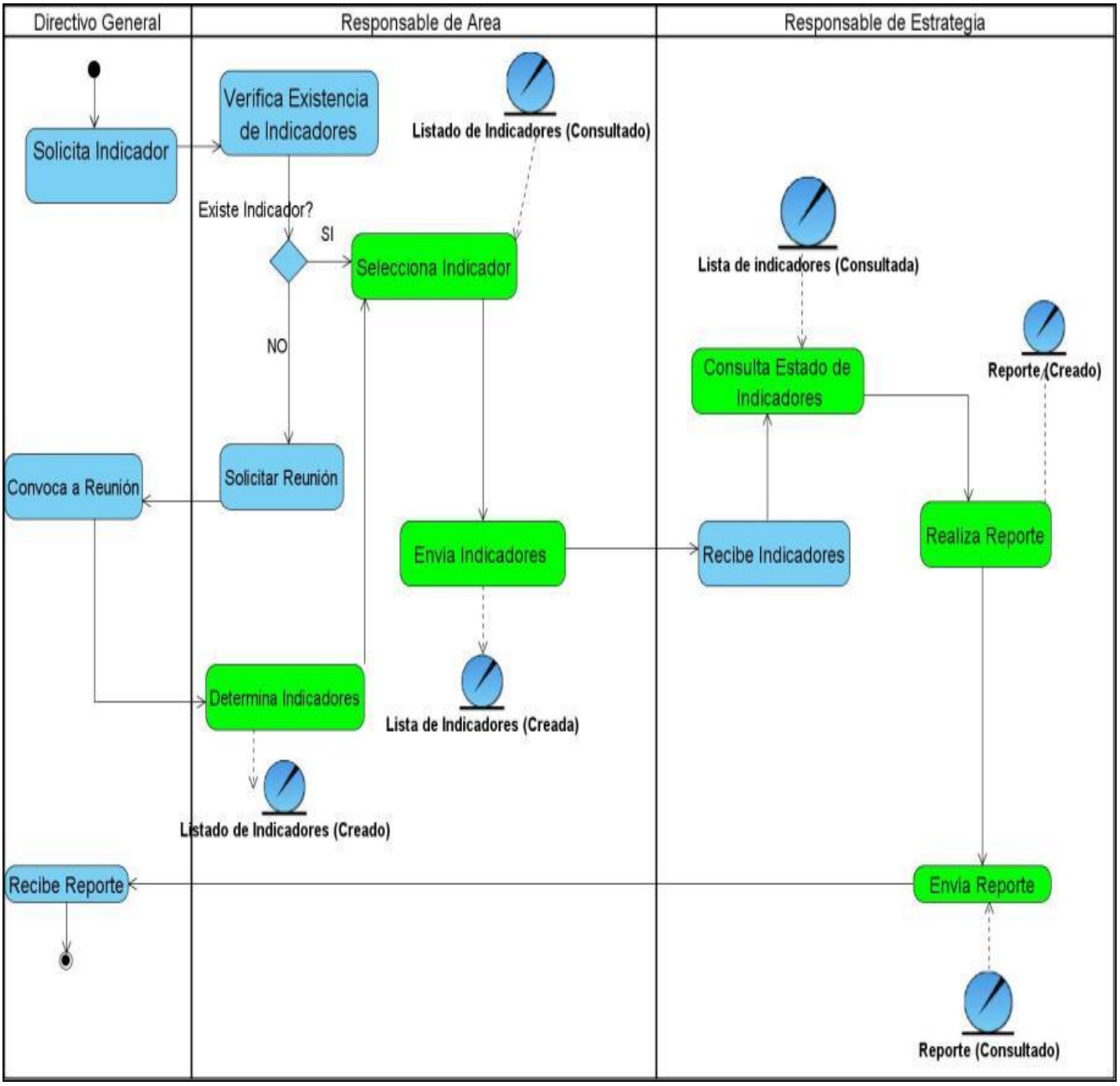


Figura 9 Diagrama de Actividades.

2.5 Modelo de Objeto

Se conoce como modelo de objetos del negocio, al modelo interno a un negocio, que muestra la relación entre trabajadores y entidades del mismo, perteneciente al flujo de trabajo de modelamiento del negocio. Describe cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo.

Figura 10 Diagrama de Modelo Objeto.

2.6 Levantamiento de Requisitos

Este flujo de trabajo es sumamente importante, pues es aquí donde se define qué es lo que debe hacer exactamente el sistema que se construye. Los requisitos son una especie de contrato que se debe cumplir, para que los usuarios finales comprendan y acepten lo que se especifique. Los requerimientos se clasifican en funcionales y no funcionales.

2.6.1 Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos permiten determinar, de una manera clara, responsabilidades que se propone el sistema. De acuerdo con los objetivos planteados se enumeran a continuación los requisitos funcionales y no funcionales del sistema. Se identificaron como requisitos funcionales los siguientes:

RF_1. Autenticar Usuario
RF_2. Adicionar Usuario.
RF_3. Modificar Usuario.
RF_4. Visualizar Usuario.
RF_5. Eliminar Usuario.
RF_6. Adicionar Proyecto.
RF_7. Visualizar Proyecto.
RF_8. Modificar Proyecto.

RF_9. Eliminar Proyecto.

RF_10 Adicionar Perspectiva.

RF_11. Eliminar Perspectiva.

RF_12. Modificar Perspectiva.

RF_13. Adicionar Indicador.

RF_14. Visualizar Indicador.

RF_15. Modificar indicador.

RF_16. Adicionar Institución.

RF_17. Eliminar Institución.

RF_18. Modificar Institución.

RF_19. Visualizar Indicador.

RF_20. Visualizar Institución.

RF_21. Visualizar Proyecto.

RF_22. Visualizar Perspectiva.

RF_23. Consultar Dashboard.

RF_24. Realizar Medición.

RF_25. Buscar Indicadores.

RF_26. Visualizar Listado de Indicadores.

RF_27. Buscar Usuarios.

RF_28. Visualizar Listado de Usuarios.

RF_29. Buscar Institución.

RF_30. Visualizar Listado de Instituciones.

RF_31. Buscar Perspectivas.

RF_32. Visualizar Listado de Perspectivas.

RF_33. Buscar Proyectos.

RF_34. Visualizar Listado de Proyectos.

2.6.2 Requerimientos no Funcionales

"Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen que el producto sea atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una

vez que sepamos lo que el sistema debe hacer, podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser"[25] (Ivar Jacobson, 2000)

Se identificaron los siguientes RNF:

Interfaz externa

1. El sistema interactúa con el usuario mediante una interfaz Web muy fácil de utilizar.

2. Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.

Portabilidad

3. El sistema tendrá la posibilidad de ser multiplataforma.

Hardware

4. La aplicación requiere estar instalada en una PC Pentium IV o superior con 2 GHz de velocidad de microprocesador como mínimo, que cuente a su vez con 1 Gb de memoria RAM tanto en el servidor de BD como en el Servidor Web, además con 160 Gb en el servidor de BD y 80 en el Servidor Web y una tarjeta de red de 100Mbps (pues todos los accesos al sistema se realizarán a través de la red) ya que es necesario que el sistema se desempeñe con un mejor rendimiento y eficiencia.

Software

5. Se debe disponer en el servidor con Windows XP, Windows 2000 Server o GNU/Linux.

Se utilizará como lenguaje de programación: Java y como gestor de Base de Datos: PostgreSQL así como el Servidor Web deberá ser Apache Tomcat en la versión 2.0.

6. Navegador compatible o superior con Internet Explorer 4.0, Mozilla Firefox 2.0.

Confidencialidad

7. La información manejada por el sistema deberá estar protegida de acceso no autorizado.

Seguridad

8. Garantizar que la información sea accedida de acuerdo a los privilegios que tienen los usuarios.

9. Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Funcionalidad

10. El sistema debe tener una mínima cantidad de páginas para ejecutar todas las funciones posibles (preferentemente que estén relacionadas).

Confiabilidad

11. Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

[25] **Jacobson, Booch, Grady y Rumbaugh 2000.** "*El proceso unificado de desarrollo de software"*.

2.7 Modelado del Sistema

2.7.1 Actores del Sistema

"Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado"[26] (Ivar Jacobson, 2000)

Representan terceros fueras del sistema que interactúan con este. Estos suelen corresponderse con trabajadores o actores del negocio.

A continuación se hace referencia a los actores del sistema, y se da una breve descripción de su función dentro del sistema:

Tabla 3 Actores del Sistema

Actores	Descripción
Administrador	Tiene todos los privilegios sobre la aplicación ya que es el responsable de manejar correctamente la información con la que contará el sistema. Es el responsable de adicionar, modificar, eliminar y visualizar los datos que se manejan desde la BD, así como de dar los permisos necesarios a los usuarios para poder acceder a la aplicación.
Usuario	Tiene acceso a un grupo menor de funcionalidades que el administrador, pero cuenta con los privilegios necesarios para navegar en el sitio accediendo a los vínculos que se le definieron como accesibles, y así lograr un mayor aprovechamiento del sistema, realizando sobre el mismo una serie de operaciones básicas, como visualizar la mayor parte de la información existente y realizar reportes para ver el estado de los indicadores.

2.7.2 Modelo de Casos de Uso del Sistema

El modelo de casos de uso no es más que un modelo del sistema que está compuesto por actores, casos de uso y la relación que existe entre ellos. Representa un esquema que recoge las funcionalidades del negocio que se automatizan y determina el uso que tendrá desde el punto de vista del usuario (actor), que su construcción es en base a las necesidades del mismo.

[26] **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* . s.l. : Pearson Education, 2000.

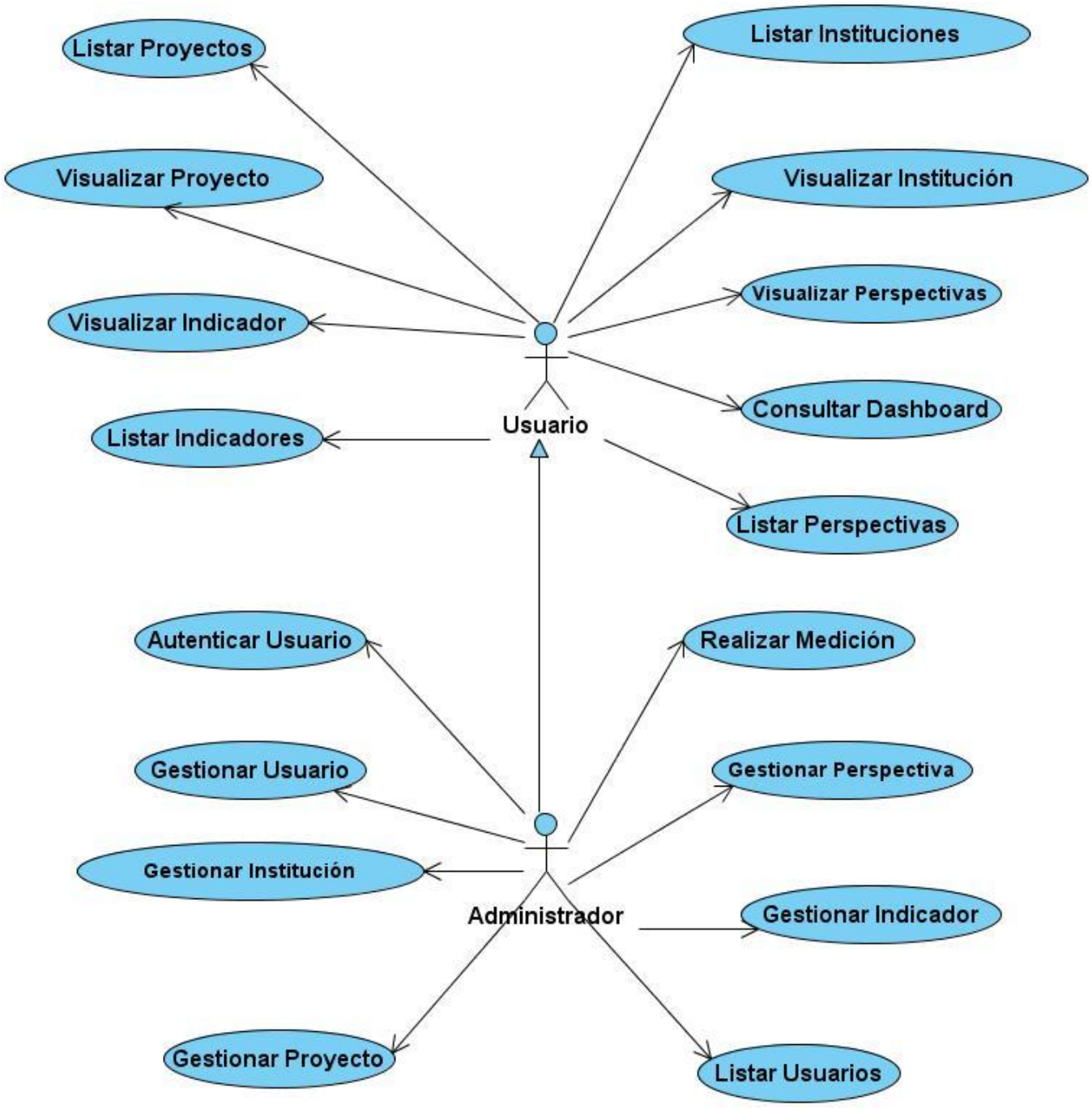


Figura 11 Diagrama de Caso de Uso del Sistema (DCUS).

2.7.3 Casos de uso del sistema

A continuación se presentan los casos de uso determinados para satisfacer los requerimientos funcionales del sistema y se da una descripción de los mismos:

2.7.3.1 Descripción textual de los CU del Sistema.

Tabla 4 Descripción del CU Autenticar Usuario

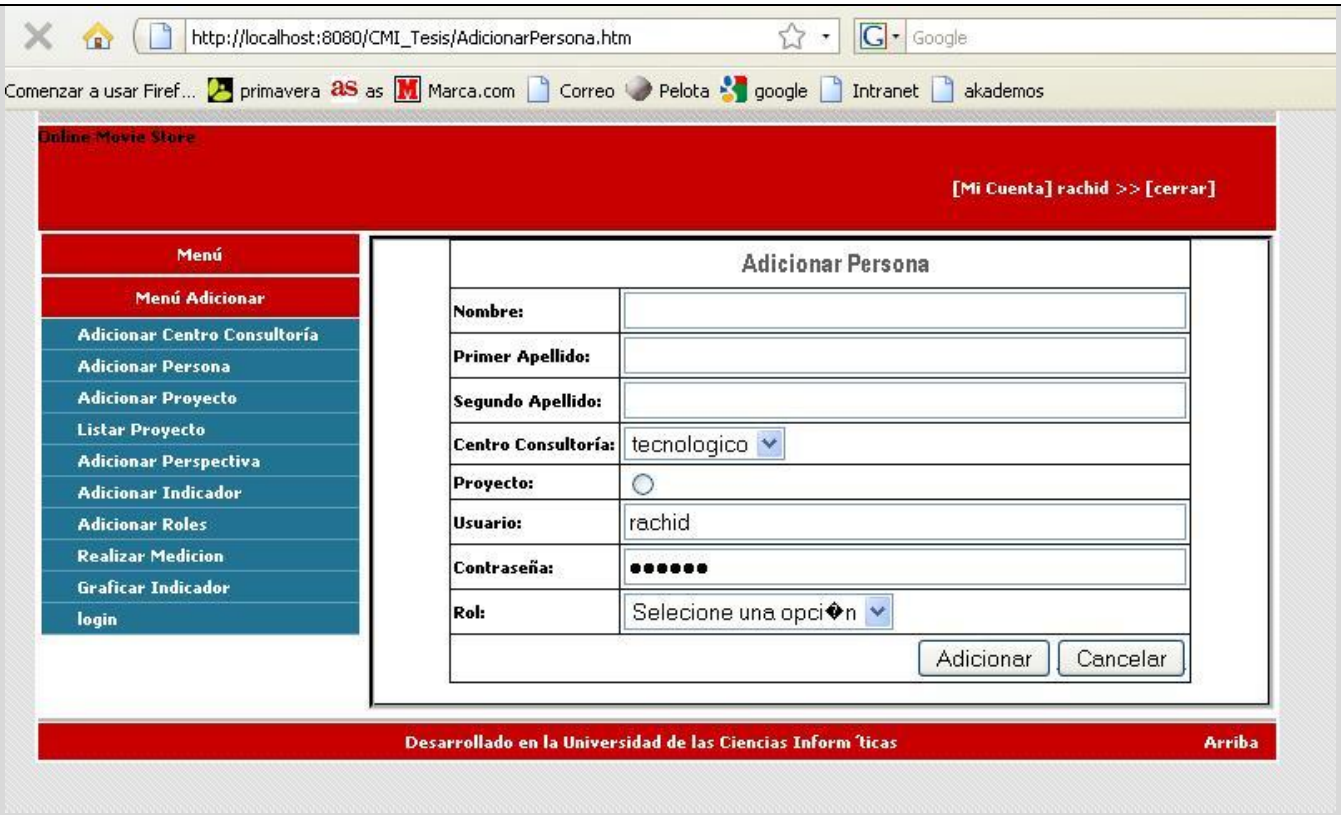
Caso de Uso	
CU_1	Autenticar Usuario
Propósito	Permitirle al usuario el acceso a la aplicación.
Actores	Usuario.
Resumen	Cuando el administrador o usuario se autentifican se verificarán sus privilegios y tendrá acceso a la información correspondiente según haya sido su autentificación.
Pre-Condiciones	Debe existir el usuario registrado en la BD, este tiene que haber sido previamente adicionado a la misma por un Administrador.
Post-Condiciones	El usuario tiene acceso solo a la información que le corresponde, si se autenticó como administrador tiene acceso a toda la información.
Referencia	R.F_1
Prioridad:	Crítico
Sección 1: Autenticar Usuario	
Acción del Actor	**Respuesta del Sistema**
1. El usuario accede a la interfaz principal de la aplicación para autenticarse.	1.2. La aplicación muestra un formulario de solicitud de los datos al usuario.
1.3. El usuario introduce en el sistema sus datos.	1.4. La aplicación verifica que los datos introducidos sean correctos o que el usuario exista en la BD.
	1.5. El sistema da acceso a las secciones que le correspondan según el privilegio de autenticación.
Flujos alternativos	
Acción del Actor	**Respuesta del Sistema**

	1.4.1 En caso de que los datos introducidos no sean correctos o el usuario no exista en la BD el sistema mostrará un mensaje de error y regresará al paso 1.2

Prototipo no Funcional de Interfaz de Usuario

Tabla 5 Descripción del CU Gestionar Persona

Caso de Uso	
CU_2	Gestionar Usuario
Propósito	El administrador gestiona los datos del usuario que tendrá permiso para usar la aplicación.
Actores	Administrador
Resumen	El administrador puede adicionar, eliminar, modificar y visualizar los datos del usuario actualizándose así la BD.
Pre-Condiciones	Tiene que existir al menos un Centro de Consultoría registrado en la BD. A excepción del Adicionar Usuario en los restantes tres escenarios, los usuarios deberán estar previamente listados.
Post-Condiciones	Se actualiza la BD.
Referencias	RF_ 2, RF_ 3, RF_ 4, RF_5
Prioridad	Crítico
Sección 1: Adicionar Usuario	
Acción del Actores	**Respuesta del Sistema**
1.1 El Administrador selecciona en el sistema la opción adicionar usuario.	1.2 La aplicación muestra un formulario solicitando al Administrador que introduzca los datos del nuevo usuario.
1.3 El Administrador introduce en el sistema los datos del nuevo usuario.	1.4 La aplicación comprueba que los datos estén escritos correctamente y que el usuario no exista en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos del usuario se han adicionado satisfactoriamente.
Flujos alternativos	
Acción del Actores	**Respuesta del Sistema**
	1.4.1 Si los datos del usuario no están escritos correctamente o este ya existe en la BD el sistema muestra un error y regresa al paso 1.2
Prototipo no Funcional de Interfaz de Usuario	



Sección 2: Eliminar Usuario	
Acción del Actor	**Respuesta del Sistema**
2.1 El Administrador selecciona la opción Eliminar Usuario a partir de la interfaz Listar Usuarios.	2.2 El sistema elimina el usuario seleccionado y se actualiza luego la BD.
	2.3 El sistema muestra la interfaz Listar Usuarios actualizada.
Sección 3: Modificar Usuario.	
Acción del Actor	**Respuesta del Sistema**
3.1 El Administrador selecciona la opción Modificar Usuario partiendo de la interfaz Listar Usuarios.	3.2 El sistema muestra un formulario con los datos del usuario antes de ser modificados.
3.3 El Administrador modifica los datos del usuario seleccionado.	3.4 Se actualiza la BD y se muestra la interfaz Listar Usuarios actualizada.
Sección 4: Visualizar Usuario	
Acción del Actor	**Respuesta del Sistema**

4.1 El administrador selecciona la opción de visualizar los datos de un usuario existente en la BD a partir de la interfaz Listar Usuarios.	4.2 El sistema muestra un formulario con los datos del usuario seleccionado para ser visualizada por el administrador, dichos datos podrán ser solamente visualizados.
	4.3 El sistema vuelve a la interfaz de Listar usuarios.

Tabla 6 Descripción del CU Gestionar Proyecto

Caso de Uso	
CU_3	Gestionar Proyecto
Propósito	El Administrador gestionará los datos de los proyectos.
Actores	Administrador.
Resumen	El administrador puede adicionar, modificar y eliminar los datos de un proyecto haciendo las respectivas actualizaciones en la BD.
Pre-Condiciones	Debe Existir al menos un Centro de Consultoría registrado en la BD. A excepción del Adicionar Proyecto en los restantes dos escenarios, los proyectos deberán estar previamente listados.
Post-Condiciones	Se actualiza la BD.
Referencias	R.F_6, R.F_7. RF_8
Prioridad	Crítico
Sección 1: Adicionar Proyecto	
Acción del Actores	**Respuesta del Sistema**
1. El Administrador selecciona la opción de adicionar proyecto.	1.2 La aplicación muestra una interfaz para que el Administrador introduzca los datos del proyecto.
1.3 El Administrador introduce en el sistema los datos del Proyecto.	1.4 La aplicación comprueba que los datos estén escritos correctamente y que el proyecto no exista en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos del proyecto se han adicionado satisfactoriamente.

Flujos alternativos	
Acción del Actores	**Respuesta del Sistema**
	1.4.1 Si los datos del proyecto no están escritos correctamente o el proyecto ya existe en la BD el sistema muestra un mensaje de error y regresa al paso 1.2
Sección 2: Eliminar Proyecto.	
Acción del Actor	**Respuesta del Sistema**
2.1 El Administrador selecciona la opción Eliminar Proyecto a partir de la interfaz Listar Proyecto.	2.2 El sistema elimina el proyecto seleccionado y se actualiza luego la BD.
	2.3 El sistema muestra la interfaz Listar Proyectos actualizada.
Sección 3: Modificar Proyecto	
Acción del Actor	**Respuesta del Sistema**
3.1 El Administrador selecciona la opción Modificar Proyecto partiendo de la interfaz Listar Proyectos.	3.2 El sistema muestra un formulario con los datos del proyecto antes de ser modificados.
3.3 El Administrador modifica los datos del proyecto seleccionado.	3.4 Se actualiza la BD y se muestra la interfaz Listar Proyectos actualizada.
Prototipo no Funcional de Interfaz de Usuario	

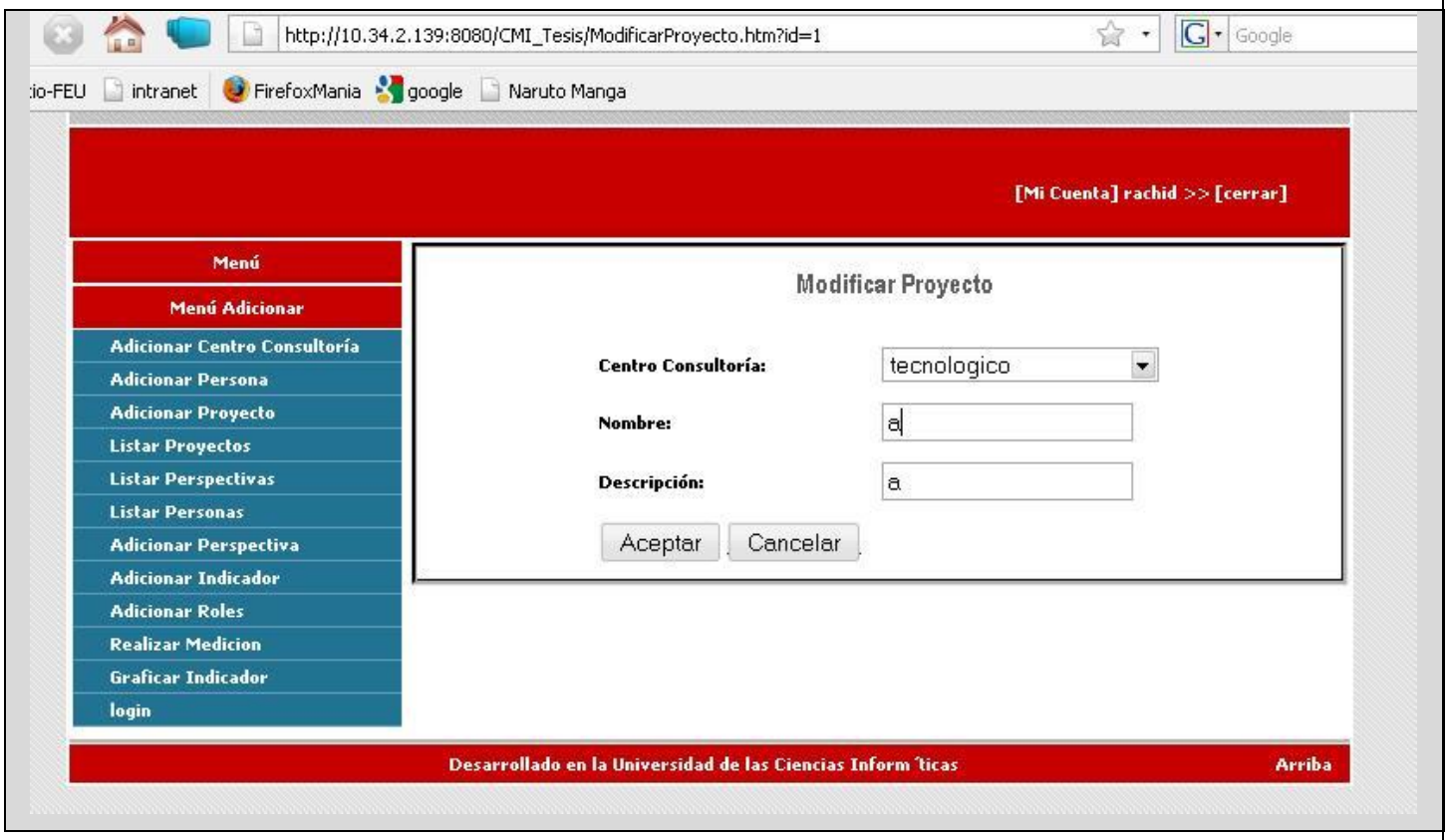


Tabla 7 Descripción del CU Consultar Dashboard.

Caso de Uso		
CU_13	Consultar Dashboard	
Propósito	Mostrar el estado de un indicador gráficamente.	
Actores	Usuario.	
Resumen	El usuario puede visualizar el estado de un indicador seleccionado de forma gráfica, mediante un conjunto de mediciones realizadas al mismo, así como obtener una descripción de su estado.	
Pre-Condiciones	El indicador debe existir en la lista de indicadores de la BD y debe tener al menos una medición realizada.	
Post-Condiciones	Se grafica el estado del indicador seleccionado, mostrándose además un reporte.	
Referencias	RF_23.	
Prioridad	Crítico	
Acción del Actor		**Respuesta del Sistema**

1.1 El usuario selecciona la opción Consultar Dashboard.	1.2 La aplicación muestra una interfaz con un formulario solicitando al usuario que entre los datos necesarios para graficar el estado del indicador.
1.3 El usuario introduce en el formulario los datos requeridos y los envía.	1.4 El sistema verifica que los datos hayan sido entrados correctamente.
	1.5 El sistema muestra la interfaz con la grafica referente al estado del indicador, así como un reporte del estado del mismo.
Flujos alternativos	
	1.4.1 Si el usuario introduce mal los datos el sistema muestra un mensaje de error y regresa al paso 1.2
Prototipo no Funcional de Interfaz de Usuario	

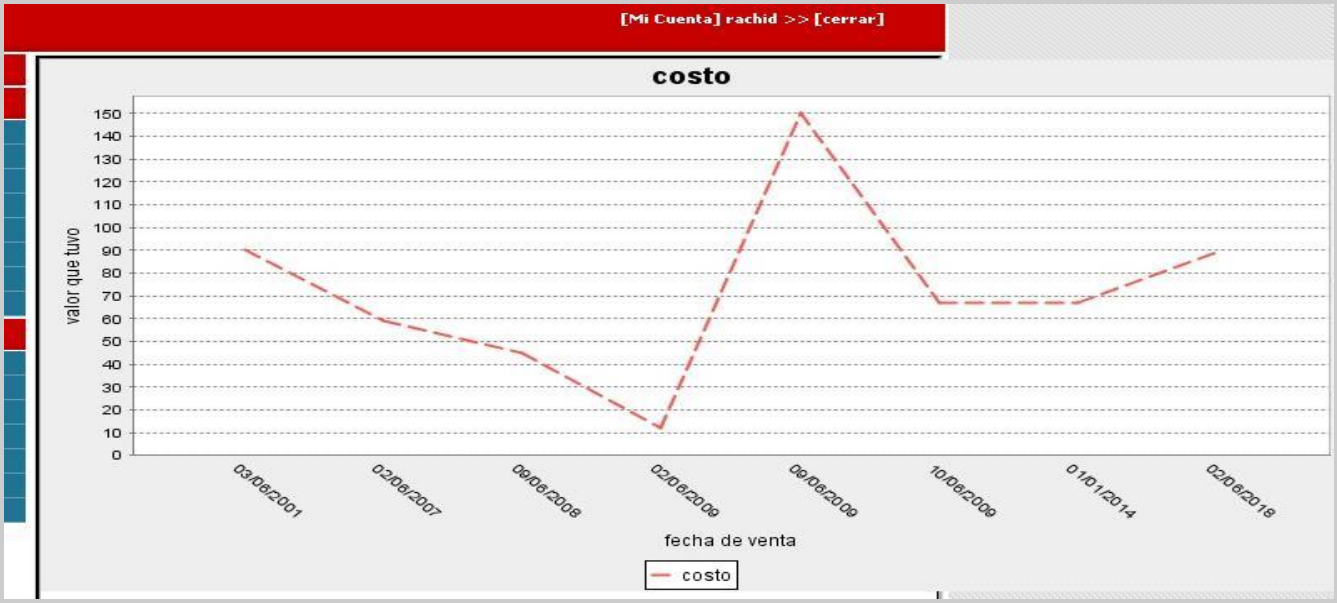


Tabla 8 Descripción del CU Realizar Medición

Caso de Uso	
CU_14	Realizar Medición
Propósito	Realizar una medición a un indicador seleccionado.
Actores	Administrador
Resumen	El administrador realiza una medición a un indicador previamente seleccionado
Pre-Condiciones	El indicador debe existir en la lista de indicadores de la BD.

Post-Condiciones	Se le realiza una nueva medición al indicador seleccionado.
Referencias	RF_24.
Prioridad	Crítico

Acción del Actor	Respuesta del Sistema
1.1 El administrador selecciona la opción Realizar Medición.	1.2 La aplicación muestra una interfaz con un formulario solicitando al administrador que entre los datos necesarios para realizar la medición.
1.3 El administrador introduce en el formulario los datos requeridos.	1.4 El sistema verifica que los datos hayan sido entrados correctamente o que no exista la medición en la BD.
	1.5 Se actualiza la BD.
Flujos alternativos	
	1.4.1 Si el usuario introduce mal los datos o existe la medición previamente en la BD la aplicación muestra un mensaje de error y regresa al paso 1.2

Prototipo no Funcional de Interfaz de Usuario

Tabla 9 Descripción del CU Listar Indicadores

Caso de Uso

CU_15	Listar Indicadores
Propósito	Mostrar una lista de todos los indicadores existentes en la BD.
Actores	Usuario.
Resumen	El usuario visualiza en una lista los indicadores existentes en la BD.
Pre-Condiciones	Debe existir en la BD al menos un Indicador.
Post-Condiciones	Se muestra un listado con los indicadores existentes en la BD.
Referencias	RF_25, RF_26.
Prioridad	Crítico

Acción del Actores	**Respuesta del Sistema**
1. El usuario selecciona la opción de Listar Indicadores del menú principal.	1.2 La aplicación muestra una interfaz con una lista de todos los indicadores existentes en la BD.
Prototipo de Interfaz	

Notar que el resto de las descripciones de los Casos de Uso están en los anexos.

2.8 Conclusiones del Capítulo: En este capítulo se obtuvo un mayor entendimiento del proceso de negocio a partir de un mejor análisis del problema en cuestión en el Centro de Consultoría, definiendo así los requisitos funcionales y no funcionales, lo que permitió identificar las funcionalidades con las que contaría el sistema (34), las que darían respuesta luego a las necesidades del problema, así como los casos de uso (17), que fueron relacionados mediante un diagrama de casos de uso del sistema. En el mismo se muestran de forma general los distintos objetos existentes en el negocio. Por otra parte se realizó una descripción detallada de los casos de uso del sistema logrando así un mayor acercamiento a lo que el sistema debería cumplir.

Capítulo 3: Diseño del Sistema

En este capítulo se pondrá de manifiesto cómo se utilizaron algunos patrones de diseño mencionados en el capítulo uno para la realización del trabajo, se hará referencia a todo lo relacionado con traducir los requisitos a una descripción de cómo desarrollar el sistema, o sea se evidenciarán los artefactos generados correspondientes al flujo de trabajo en cuestión para la solución desde el punto de vista informático, se mostrará el modelado visual de su desarrollo y se presentará además la arquitectura del mismo, así como una adaptación del diseño, haciéndolo corresponder con su ambiente de implementación desde el punto de vista de los distintos diagramas; por otra parte quedará mostrado el modelo físico de la base de datos y la realización del modelo de despliegue.

3.1 Aplicación dada a algunos de los Patrones de Diseño utilizados en el desarrollo del sistema.

Patrón Alta Cohesión

Spring como framework proporciona una *Alta Cohesión* entre clases, pues estas son creadas con el fin de cumplir responsabilidades estrechamente relacionadas sin que se realice un trabajo enorme. Lo que quedó evidenciado en la creación de la interfaz IndicadorServ con su respectiva implementación en IndicadorServImpl, cumplimentando lo anteriormente planteado.

Patrón Singleton

Se pone de manifiesto ya que a través de un objeto específico se puede acceder a una única clase, y desde otras clases utilizar dicho objeto de forma global, proporcionando así mayor accesibilidad a dichas instancias. Ejemplo:

En las Clases controladoras se utilizan objetos que son instancias únicas de las diferentes clases servicios (entiéndase, en la clase MedTestCont se utiliza el objeto indicadorServ que es propio de la clase servicio del mismo nombre).

Patrón DAO (Data Access Object)

Se utiliza para la realización de este trabajo el patrón **DAO** evidenciándose en el código del proyecto donde tiene un paquete propio que permite contar con diversas fuentes de datos (BD, archivos, servicios) de tal forma que se encapsula la manera de acceder a la fuente de datos. Este trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los

datos, o de cuál es la fuente de almacenamiento. Este patrón separa la persistencia de datos del resto de funcionalidades del sistema.

Patrón Inyección de Dependencia

Otro patrón muy importante para la realización de este trabajo fue el de **Inyección de Dependencia,** pues su función básicamente radica en resolver las dependencias de cada clase (atributos) generando los objetos en tiempo de ejecución (cuando se arranca la aplicación) y luego inyectarlos en los demás objetos que los necesiten a través de métodos set o bien a través del constructor, pero estos objetos se instancian una vez, se guardan en una factoría y se comparten por todos los usuarios (al menos en el caso de Spring) evitando tener que andar extendiendo clases o tumbar el servidor de la BD.

Se hace notar que en los anexos se encuentran las demás descripciones de los patrones utilizados.

3.1.1 Arquitectura

"La arquitectura del software proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa. Además, el diseño de datos es una parte integral para la derivación de la arquitectura del software. La arquitectura marca decisiones de diseño tempranas y proporciona el mecanismo para evaluar los beneficios de las estructuras de sistema alternativas"[27] (Pressman, 2005)***.***

La arquitectura no es un software operacional propiamente dicho, sino que es la representación que provee al ingeniero de software analizar la efectividad del diseño para la consecución de los requisitos fijados, además permite considerar las diferentes variantes arquitectónicas y ayuda a disminuir los posibles riesgos que pudiera correr el desarrollo de software.

[27] Pressman, R.S., *Ingienería del Software. Un enfoque práctico.* V ed. Vol. I. 2005.

3.1.2 Arquitectura utilizada

La arquitectura que se utilizó para la realización de la aplicación fue **La arquitectura Modelo Vista Controlador (MVC)** introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el *Modelo*, las *Vistas* y los *Controladores* se tratan como entidades separadas; esto hace que cualquier cambio producido en el *Modelo* se refleje automáticamente en cada una de las *Vistas*.

A continuación se presenta una figura donde se muestra la *arquitectura MVC* en su forma más general. Hay un *Modelo*, múltiples *Controladores* que manipulan ese *Modelo*, y hay varias *Vistas* de los datos del *Modelo*, que cambian cuando cambia el estado de ese *Modelo*.

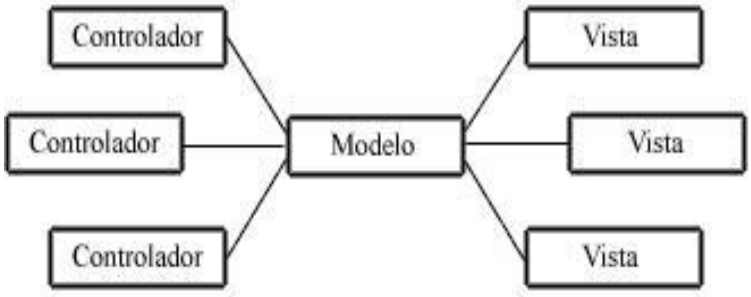


Figura 12 Arquitectura general del MVC

Este modelo de arquitectura presenta varias ventajas dentro de las cuales se pueden señalar:

- Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado
- Hay una API muy bien definida; cualquiera que use la API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

3.2 Modelo de Análisis.

Luego de haberse definido los requerimientos funcionales del sistema los diagramas de clases del análisis pasan a dar una descripción más específica de lo que es la estructura de las clases que lo conforman, igual quedan definidas las relaciones existentes entre ellas. El Modelo de Análisis ayuda a descomponer la implementación en partes más sencillas que puedan ser desarrolladas por diferentes equipos. En el caso particular del desarrollo de este sistema no se consideró relevante la idea de realizar Modelo de Análisis, pues ya se tiene experiencia en el modelado con UML y se decidió como mejor opción dedicar el esfuerzo en el diseño de las clases que se iban a implementar.

3.3 Modelo de Diseño

"El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación"[28] (Ivar Jacobson, 2000)

Se evidenció en el presente trabajo la utilización del Modelo de Diseño a la hora de confeccionar los Diagramas de Clases del Diseño (DCD), para la realización de dichos diagramas se utilizó el patrón Modelo Vista Controlador (MVC) debido a que el framework utilizado en este trabajo (Spring) utiliza este patrón; permitiendo la separación de la vista de la aplicación de las clases del modelo, todo esto debido a que se usa una capa controladora que es la encargada de fomentar los cambios en dicha vista y en el propio modelo.

[28] **Jacobson, Booch, Grady y Rumbaugh 2000.** "*El proceso unificado de desarrollo de software*".

3.4 Diagramas de Clases del Diseño.

Los diagramas de clases del diseño (DCD) describen gráficamente las especificaciones de las clases de software y de las interfaces (por ejemplo, las de Java) en una aplicación, o sea se utilizan con el fin de modelar aquellas relaciones que tienen lugar entre las diferentes clases. A la hora de confeccionar dichos diagramas se utilizó el patrón de diseño MVC ya que fue utilizada la capa Web modelo-vista-controlador del framework Spring, en ella se usa una capa controladora, que es la encargada de fomentar los cambios en dicha vista y en el propio modelo.

Un DCD contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

En el modelado de la aplicación aparece un subsistema que representa los componentes del framework Spring y recibe como nombre: *Componentes Spring* y otro subsistema encargado del mapeo objeto-relacional llamado *Hibernate*.

A continuación se mostrarán algunos de los diagramas de clases del diseño más importantes utilizados en la realización de este trabajo. Notando que el resto de los DCD se encuentran en los anexos.

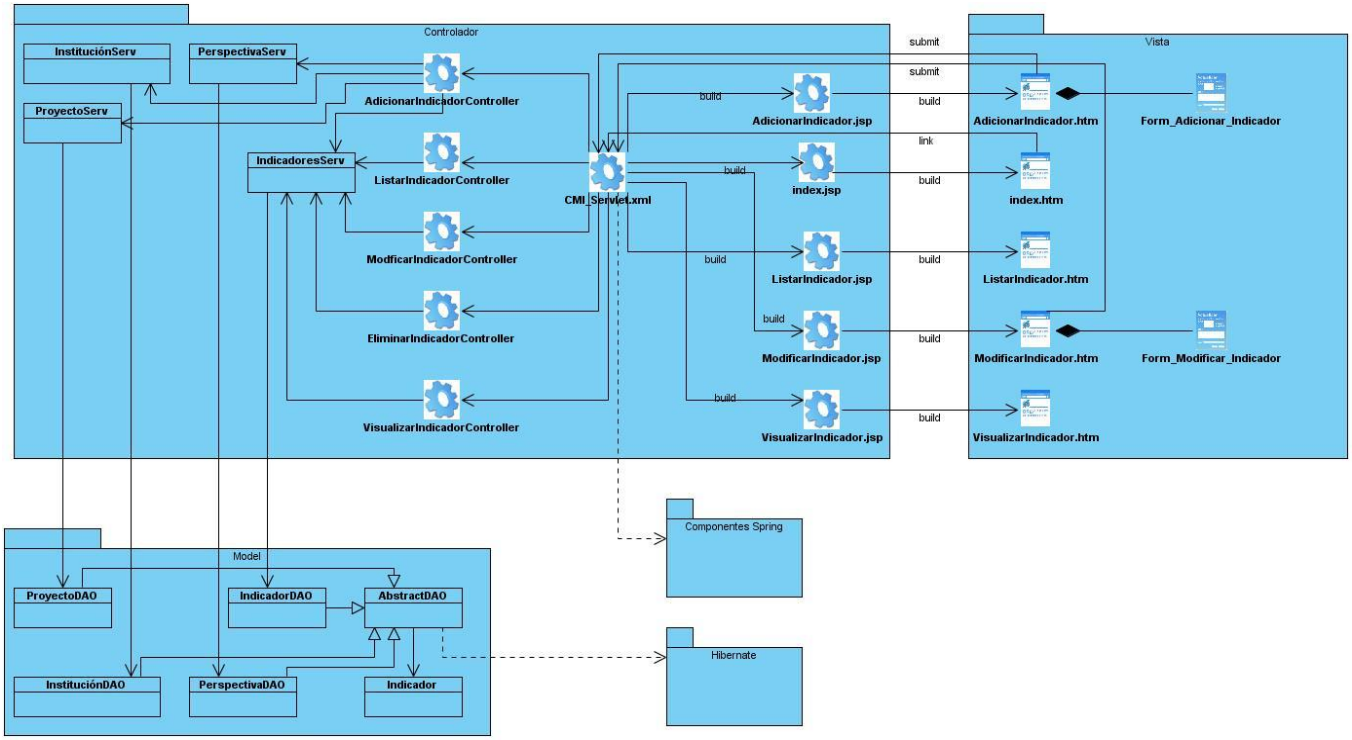


Figura 13 Diagrama de Clases del Diseño “Gestionar Indicador”

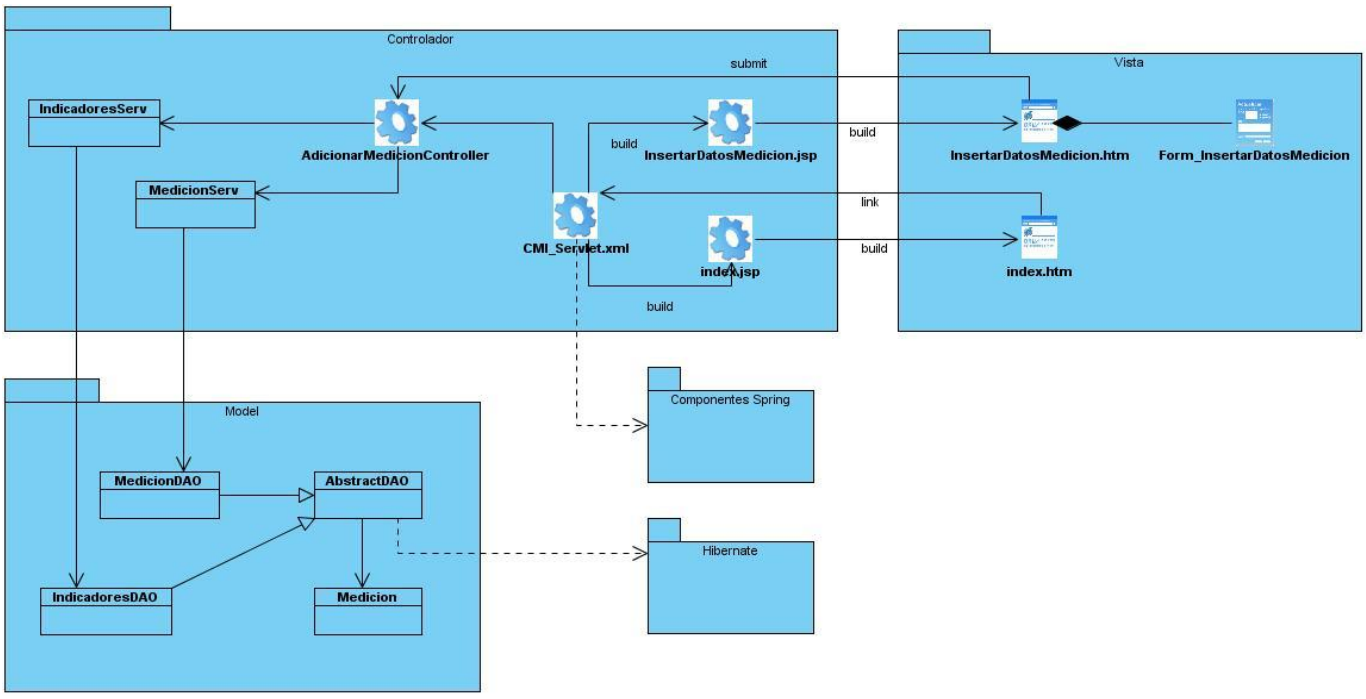


Figura 14 Diagrama de Clases del Diseño Realizar Medición

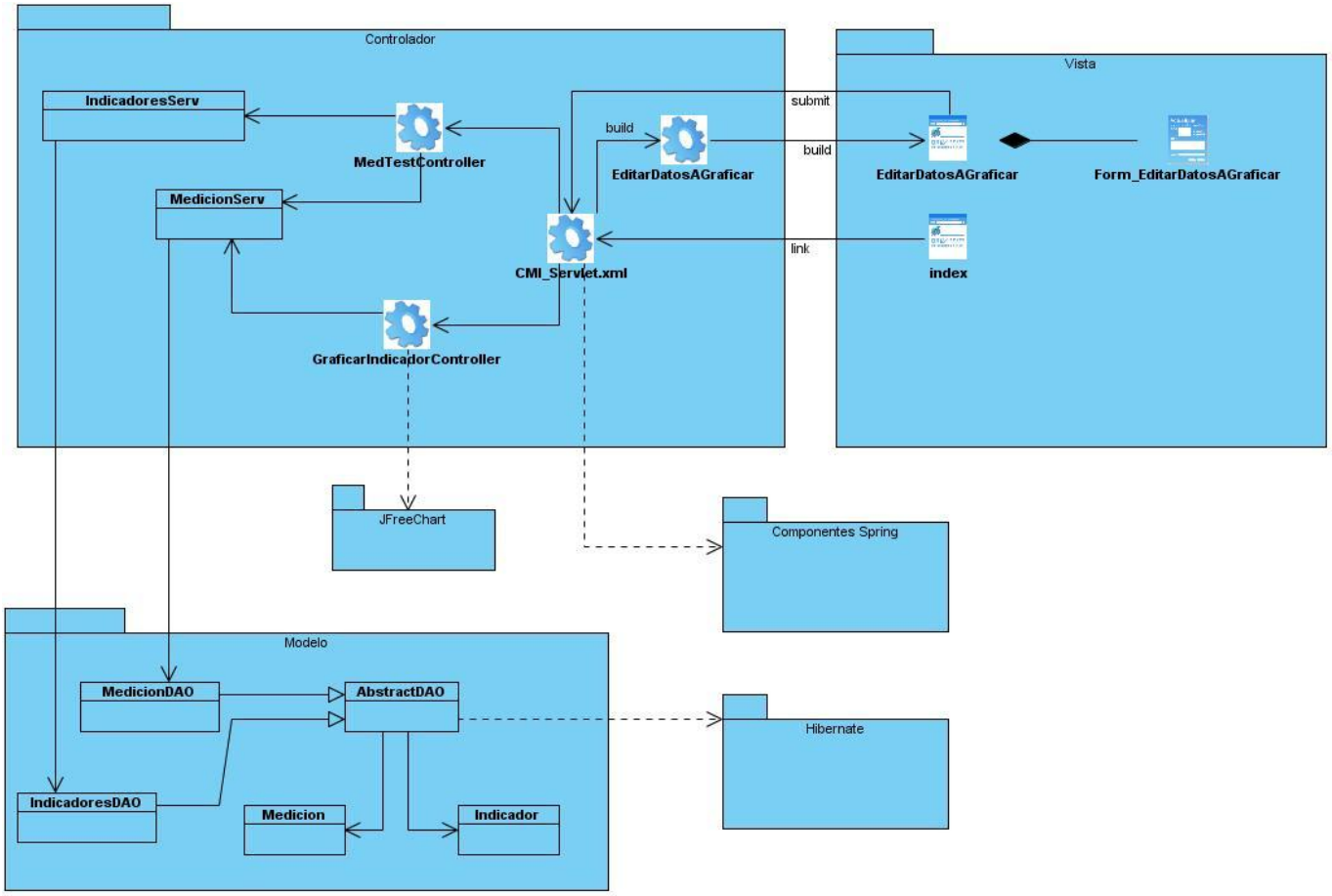


Figura 15 Diagrama de Clases del Diseño Consultar Dashboard

3.5 Diagramas de Interacción. Diagramas de Secuencia. Descripciones

Los diagramas de interacción se dividen en dos tipos de diagrama UML, los *diagramas de secuencia* y los *diagramas de colaboración*. Para modelar los aspectos dinámicos de este sistema se utilizó *diagramas de secuencia* por cada caso de uso.

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase, además contiene

detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

A continuación algunos diagramas de secuencia utilizados con sus respectivas descripciones:

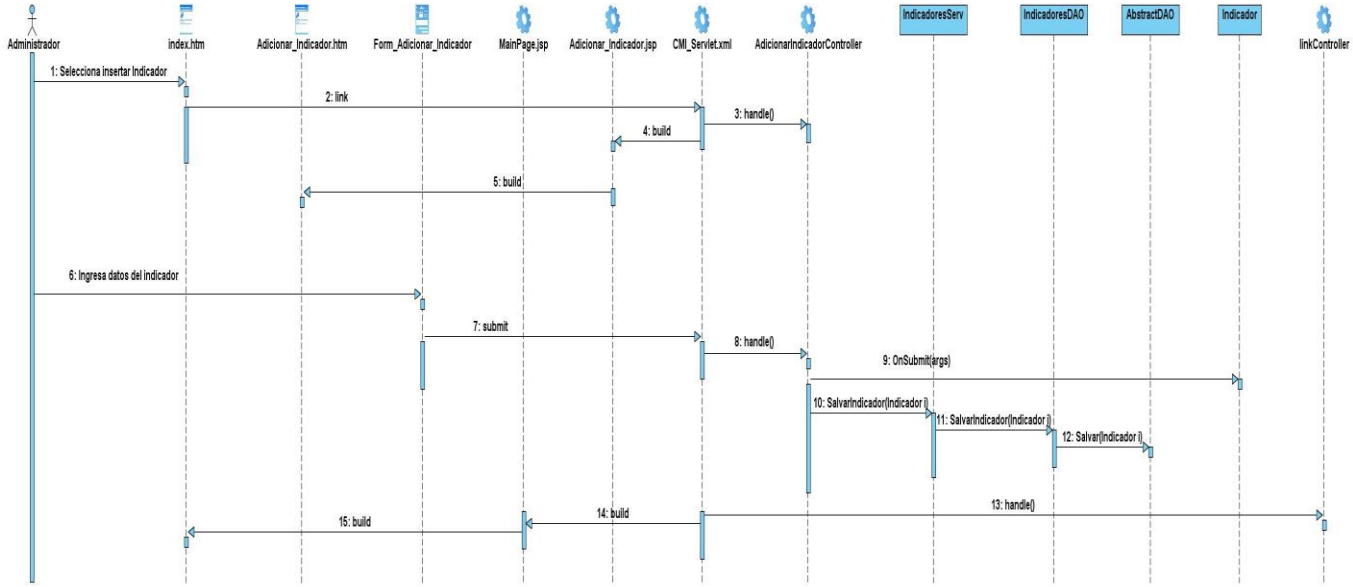


Figura 16 Diagrama de Secuencia del CU Gestionar_Indicador, escenario Insertar_Indicador

La figura 16 muestra el diagrama correspondiente al CU Gestionar Indicador en su escenario insertar. El flujo se inicia con la petición por parte del usuario en la página cliente index mediante el botón Adicionar Indicador, mediante un link el mismo ordena al Dispatcher Servlet que muestre la página cliente Adicionar Indicador, haciendo uso del controlador AdicionarIndicadorController el cual devuelve al Dispatcher los datos que le permiten a la JSP adicionar_indicador.jsp que genere la página HTM correspondiente; haciendo uso luego del formulario contenido en la página ingresa los datos del indicador que se adicionará, enviando los datos a través de un submit al Dispatcher-Servlet que es el encargado de buscar el controlador que dé respuesta a la nueva petición en este caso AdicionarIndicadorController, este a su vez hace uso de los servicios ofrecidos por la clase IndicadorServ e Indicador DAO, que es una clase que hereda de AbstractDAO que es quién hace las operaciones correspondientes en la BD, para el ejemplo crear el nuevo indicador y almacenarlo. Por último el Dispatcher-Servlet selecciona la página controladora linkController que es la que tiene los datos que permitirán a la Index.jsp generar su .htm correspondiente y termina el escenario.

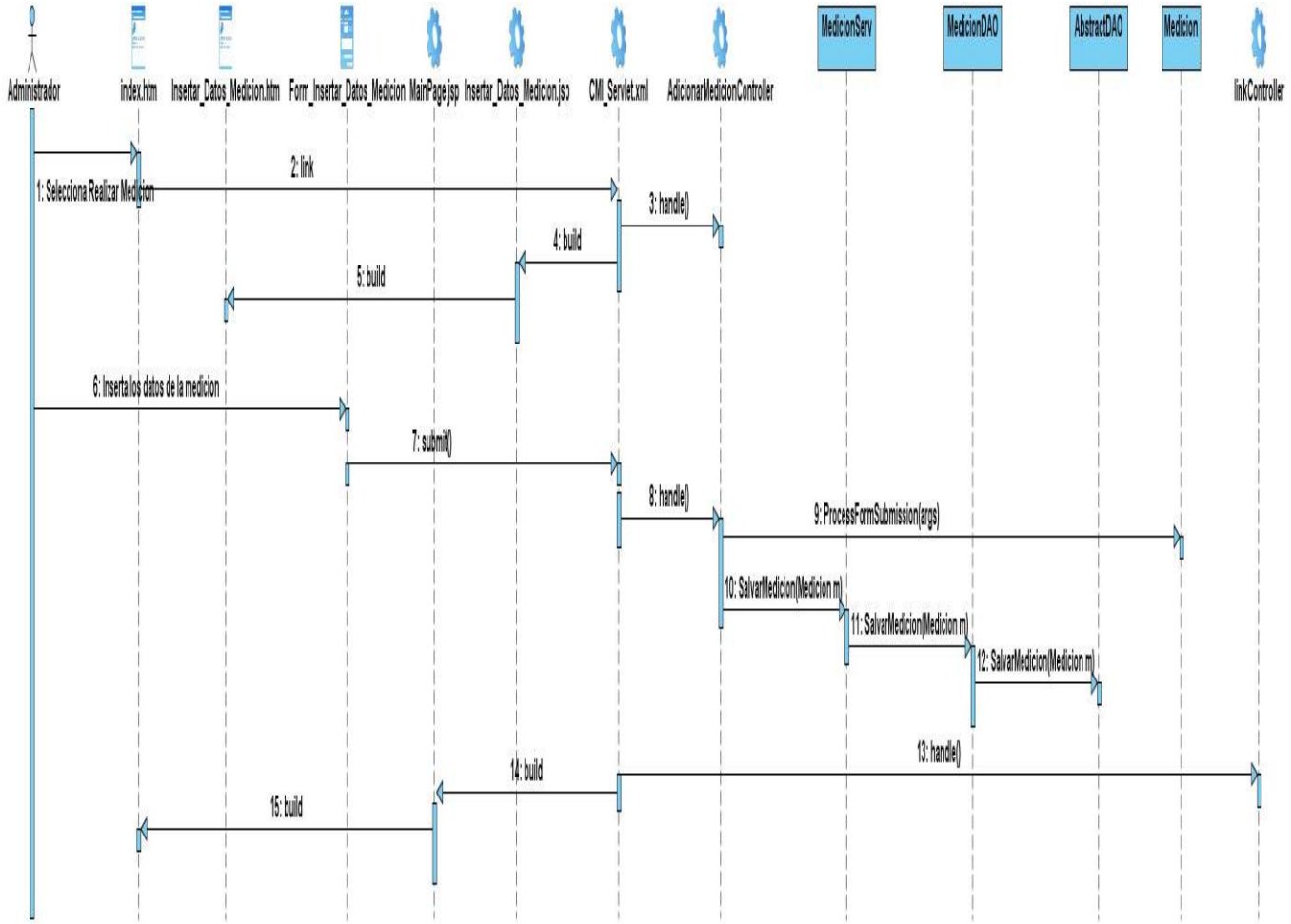


Figura 17 Diagrama de Secuencia del CU Realizar_Medición

La figura 17 muestra el diagrama de secuencia del CU Realizar Medición, el cual da inicio una vez que el Administrador selecciona la opción Realizar_Medición en la página Index.htm, la misma envía un link al Dispatcher-Servlet y este selecciona al controlador AdicionarMedicionController que es el encargado de devolver los datos necesarios para construir la página Editar_Datos_Medición.jsp y esta a su vez genera la .htm correspondiente. Una vez ocurrido esto el Administrador inserta los datos de la medición en el formulario, los cuáles son enviados mediante un submit al Dispatcher para a través del AdicionarMedicionController, lo que permite mediante el método **onSubmit** () crear la medición, salvarla con la clase MedicionServ que a su vez mediante la MedicionDAO haciendo uso de la AbstractDAO salva dicha medición. Finalmente se hace uso de la clase controladora linkController para que el Dispatcher-Servlet pueda construir mediante la Index.jsp su .htm correspondiente.

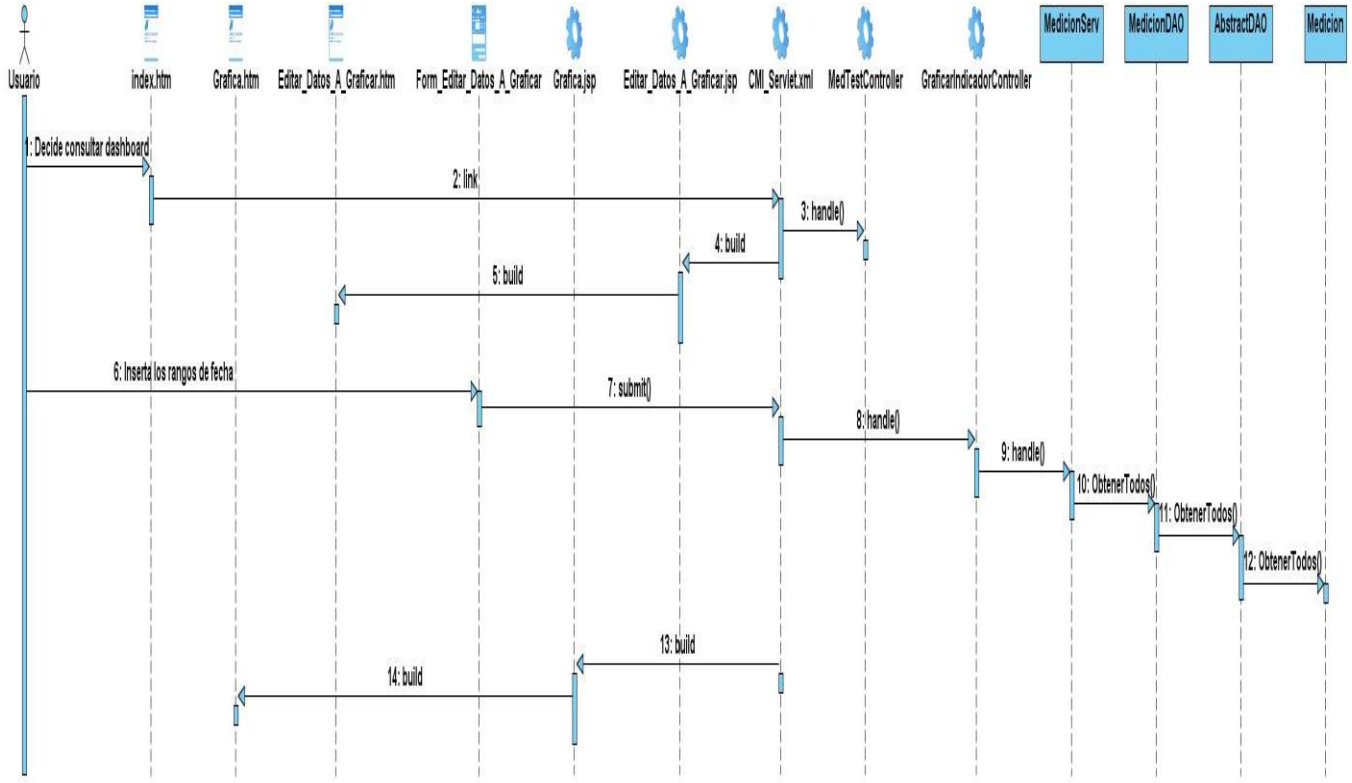


Figura 18 Diagrama de Secuencia de Consultar Dashboard

En la figura 18 se muestra el diagrama de secuencia correspondiente al Caso de uso Consultar Dashboard, donde el usuario le indica a la aplicación que desea ver de forma gráfica como se ha comportado el estado de un indicador basado en las mediciones realizadas al mismo; se inicia cuando el Administrador selecciona la opción Consultar Dashboard en la página cliente Index.htm, acto seguido se envía un link al Dispatcher-Servlet el cual selecciona el controller encargado de responder la petición (MedTestController) para construir la vista deseada, en este caso la pagina cliente Insertar_Datos_a_Graficar.htm, en esta página el usuario selecciona los rangos de fecha en los que se realizó las mediciones, al indicador en cuestión al que se le hará la medición así como el tipo de gráfico que utilizará, una vez todo configurado se le ordena a la aplicación que muestre el gráfico por lo que nuevamente el Dispatcher-Servlet selecciona la clase controladora encargada de ello, en este caso GraficarIndicadorController, el que le proporciona los datos necesarios al Dispatcher-Servlet para que construya la pagina cliente Grafica.

Notar: el resto de los diagramas se encuentran en los anexos con sus respectivos DCD.

3.6 Concepción del Mapa de Navegación

Los mapas de navegación proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. La **cuestión clave de un mapa de navegación** es que el usuario acude a él **porque en la página principal no ha encontrado** la información que está buscando.

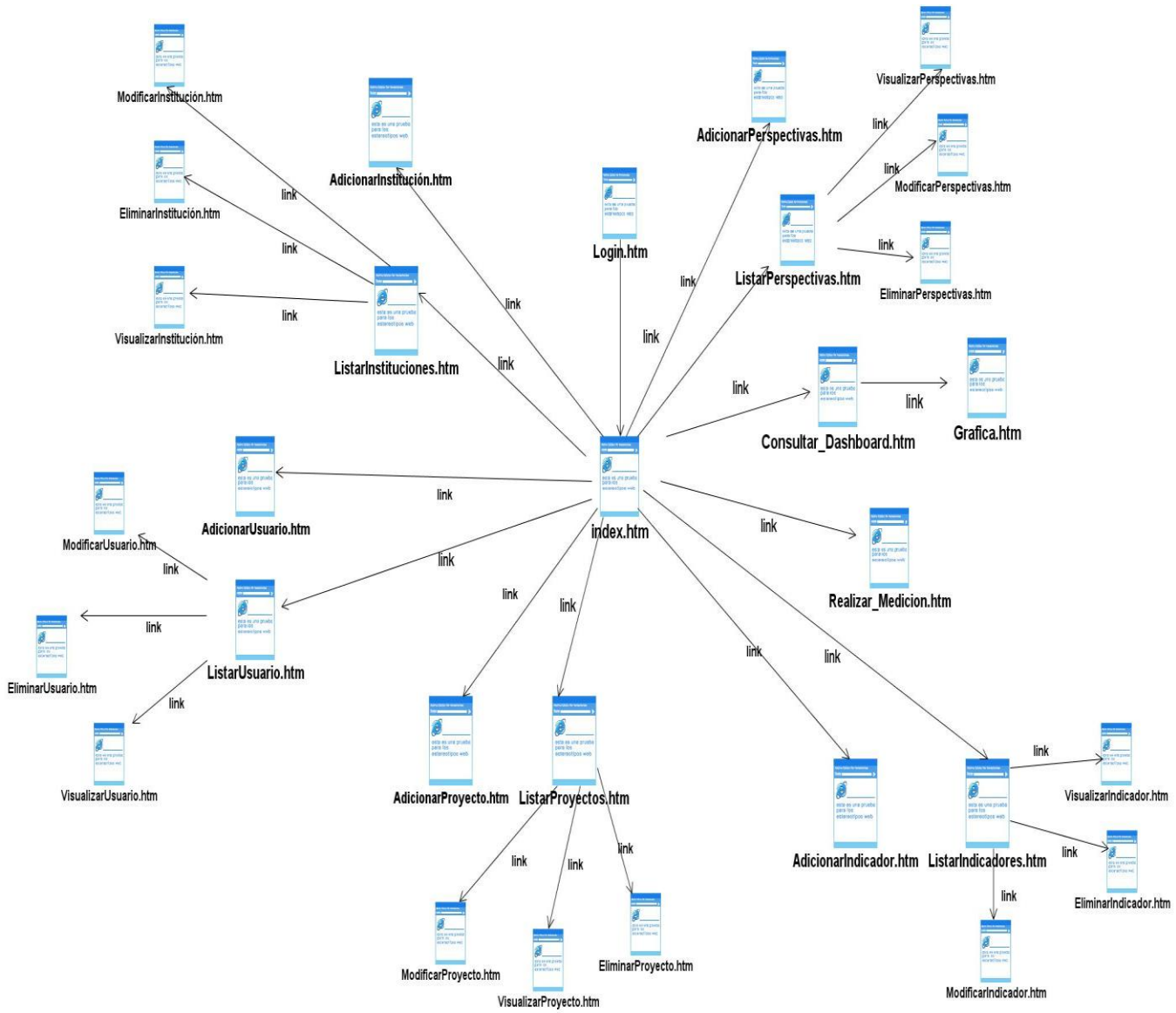


Figura 19 Mapa de navegación del sistema

3.7 Vista Lógica

Vista Lógica: Describe la visión lógica del sistema presentando la arquitectura y los elementos más relevantes dentro de la misma, describe además las partes significativas desde el punto de vista de la arquitectura del modelo de diseño, como su división en subsistemas y paquetes. Además de eso, para cada paquete significativo, ella muestra su división en clases y utilitarios de clase. Presenta las clases significativas desde el punto de vista de la arquitectura y describe sus responsabilidades, así como algunas relaciones, operaciones y atributos de gran importancia.

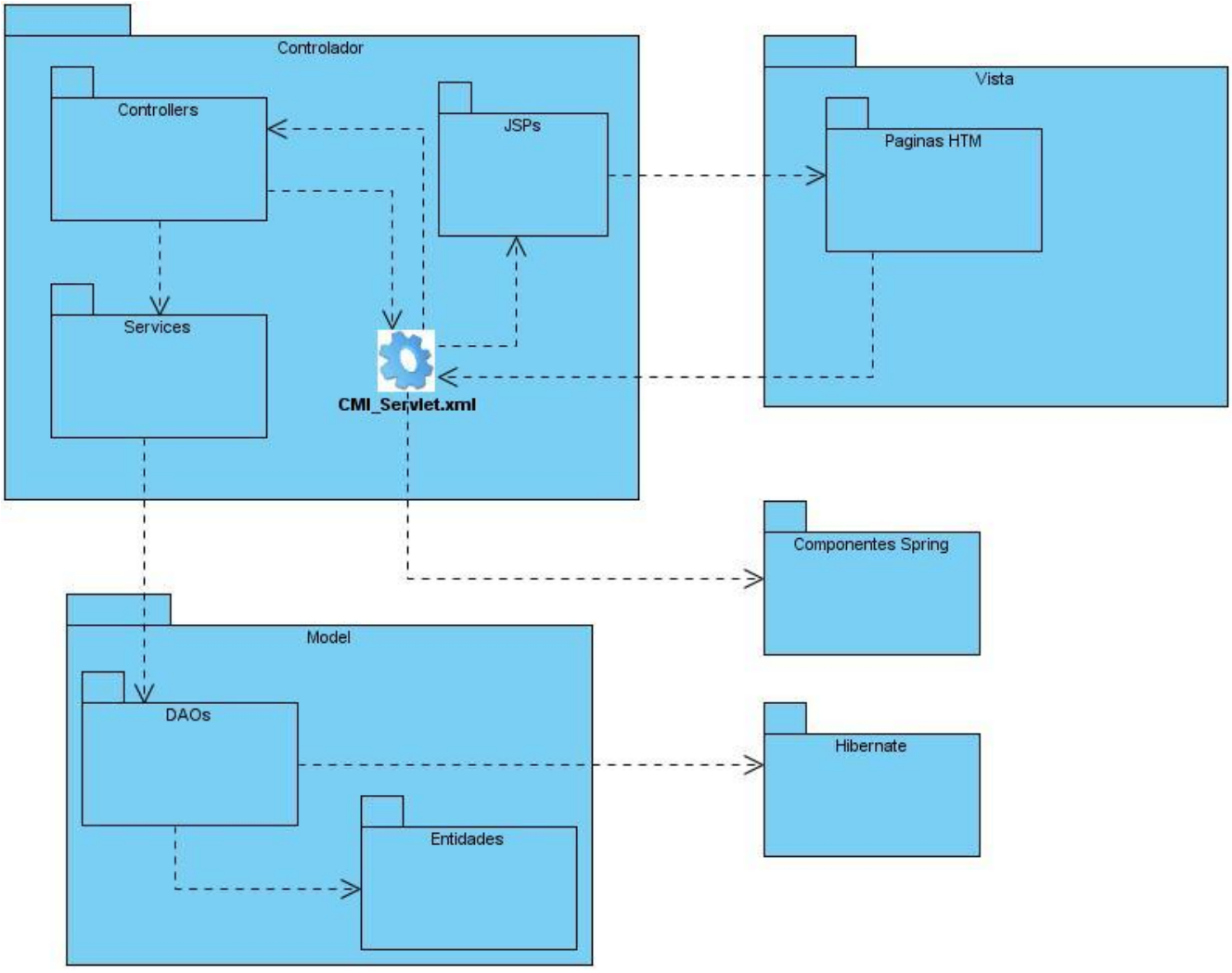


Figura 20 Vista Lógica del Sistema

A continuación se muestran las Interfaces de los servicios y sus respectivas implementaciones:

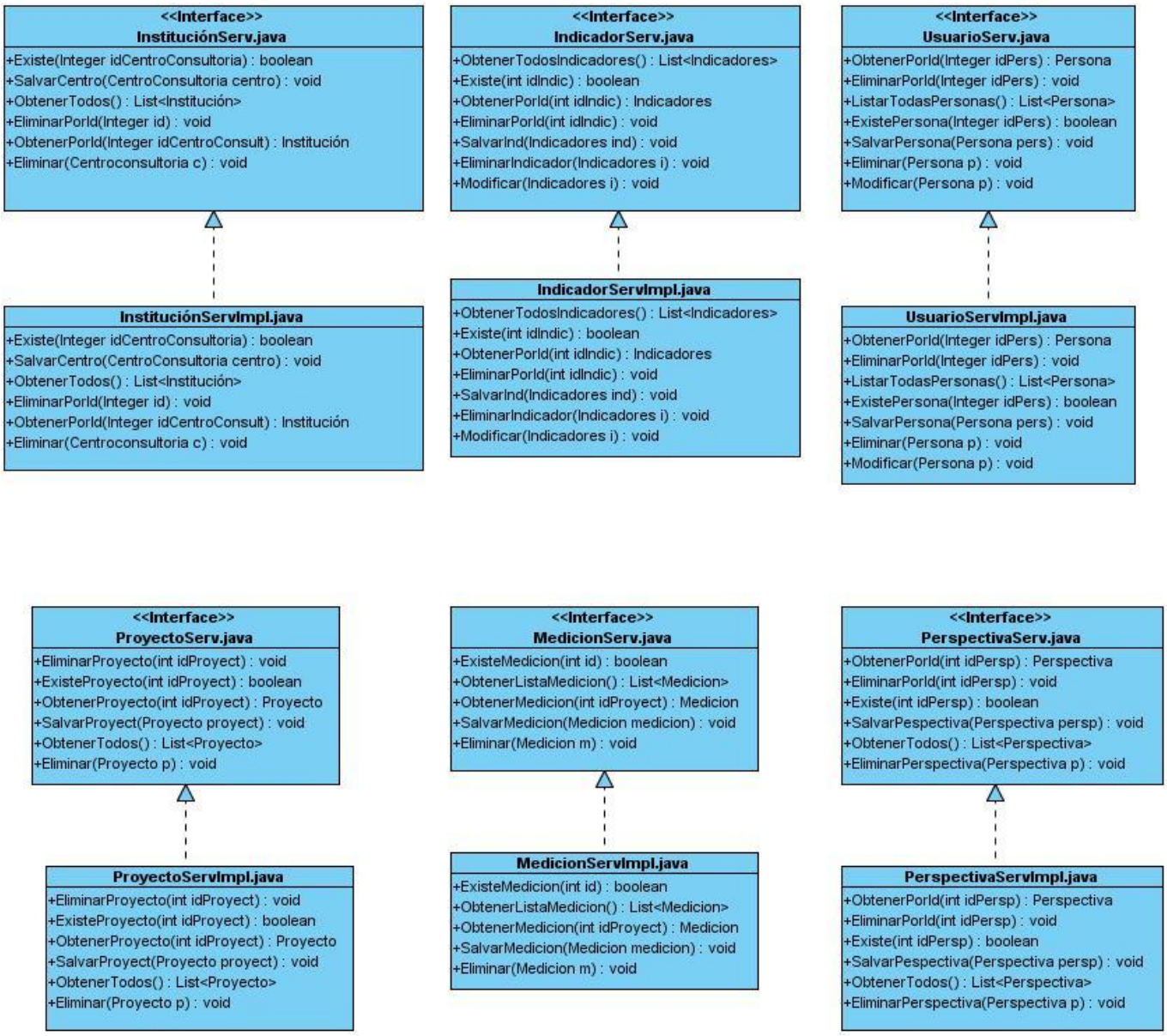


Figura 21 Clases del Servicio del Sistema

A continuación se muestra la capa de acceso a datos del sistema:

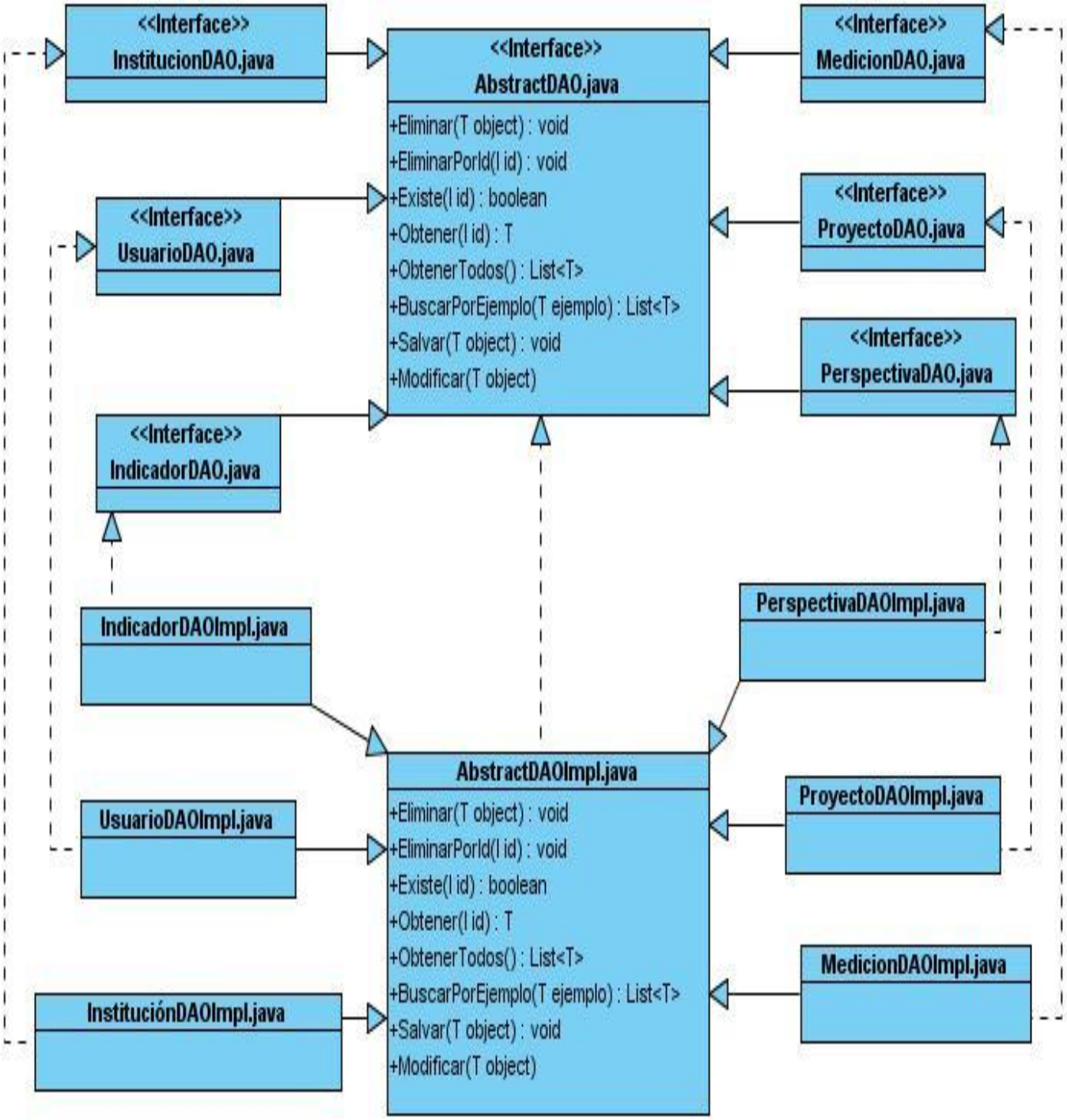


Figura 22 Capa de Acceso a Datos del Sistema

3.8 Diagrama de Clases Persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases que deben ser persistentes se definen durante el Diseño como tal del producto y es responsabilidad del diseñador su definición. Como regla se puede decir que: cuando una clase que está formada por otras clases es persistente, automáticamente las clases componentes también lo son. Por lo general las clases persistentes tienen como origen las clases clasificadas como entidad porque ellas modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto. El diagrama de clases persistentes describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

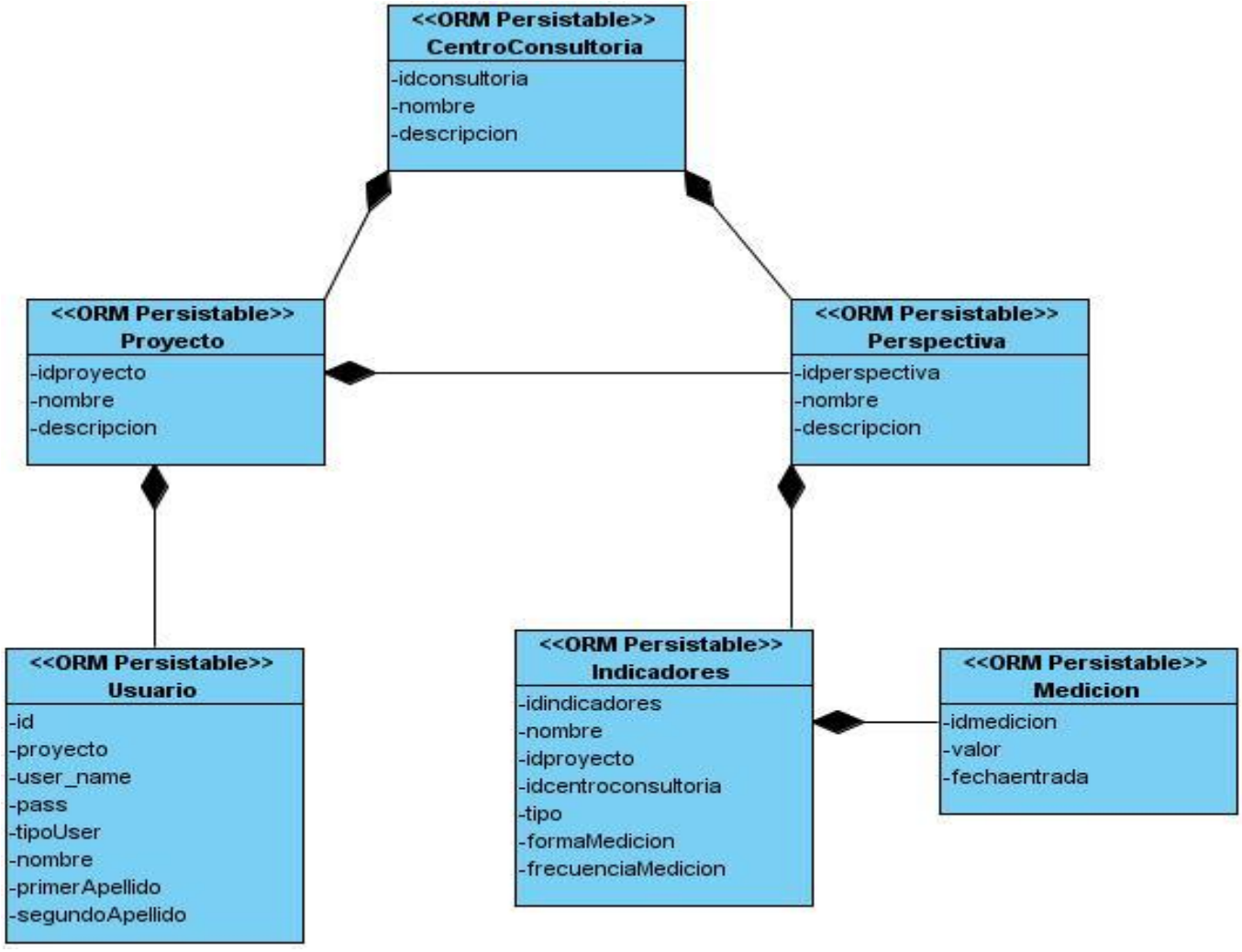


Figura 23 Diagrama de Clases Persistentes

3.9 Modelo de Datos

Un modelo de datos básicamente es *"un conjunto de conceptos, reglas y convenciones que nos permiten describir y en ocasiones manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos"*[29] Al producto del modelo de datos se le llama esquema (descripción de la estructura de la base de datos) y a los datos en concreto almacenados en la base de datos en ese momento, ocurrencia del esquema, lo que sería en Java una clase y una instancia de la clase.

El modelo de datos está formado por dos componentes, componente estática, relacionada con el lenguaje de definición de datos (LDD) y dinámica, relacionada con el lenguaje de manipulación de datos (LMD). La parte estática se refiere a la estructura y la dinámica a que operaciones puedo realizar sobre cada objeto.

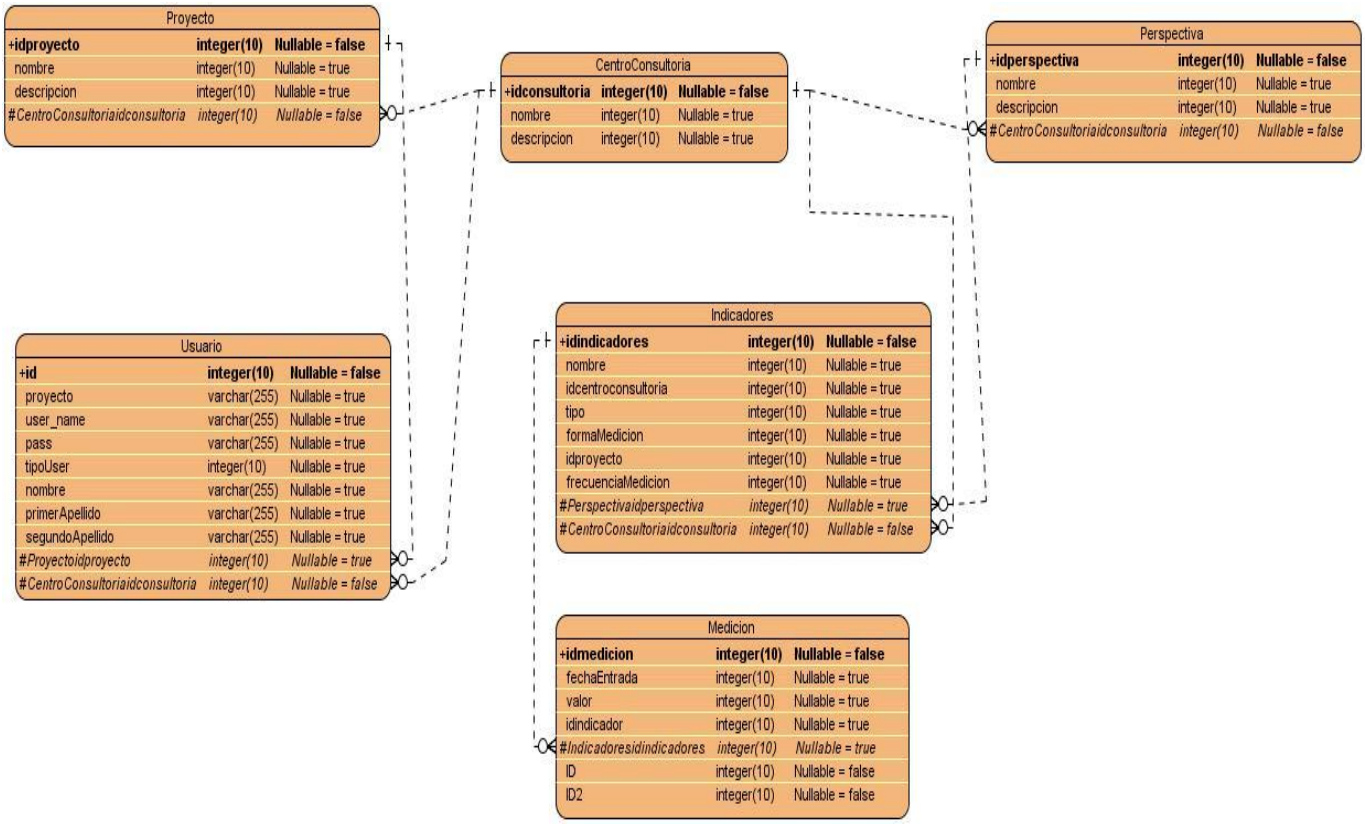


Figura 24 Diagrama de Modelo de Datos.

[29] Cito en http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/

3.9.1 Modelo físico de la Base de Datos

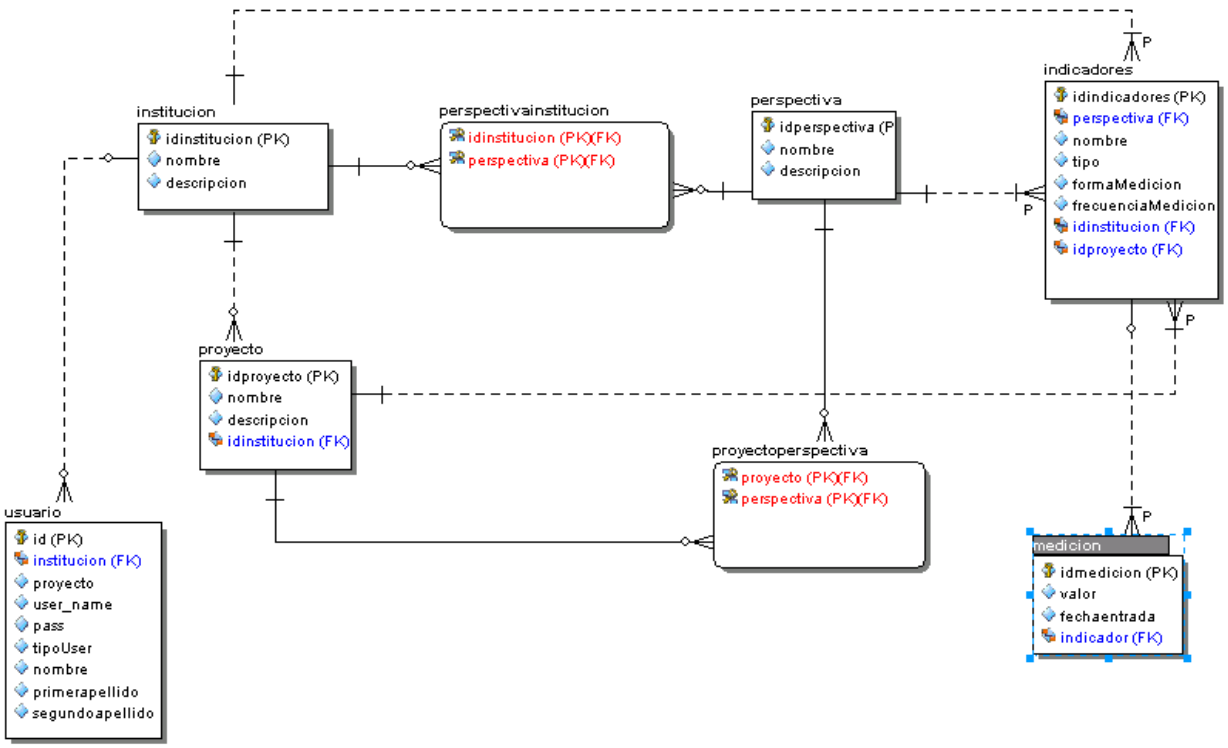


Figura 25 Modelo Físico de la Base de Datos

3.9.2 Descripción de las tablas de la Base de Datos

Descripción de la Tabla 10: "indicadores"

Nombre: indicadores		
Descripción: Almacena todos los indicadores pertenecientes a las perspectivas		
Atributo	**Tipo**	**Descripción**
idindicadores	integer	id del indicador de la tabla
perspectiva	Integer	id de la perspectiva a la que pertenece
nombre	varChar	nombre del indicador
tipo	varChar	almacena un tipo de indicador posible
formademedicion	varChar	forma de medición que se utiliza
frecuenciademedición	char	frecuencia con la que se mide el indicador
idconsultoria	integer	id de la consultoria si es que pertenece
idproyecto	integer	id del proyecto si es que pertenece
descripción	varChar	describe la función del indicador

Descripción de la Tabla 11: "institución"

Nombre: centroconsultoria		
Descripción: Almacena todas las instituciones existentes en la base de datos		
Atributo	**Tipo**	**Descripción**
idinstitucion	integer	id de la institución en cuestión
nombre	varChar	nombre de la institución
descripción	varChar	describe la función de la institución

Descripción de la Tabla 12: "medición"

Nombre: medicion		
Descripción: Almacena todos los centros de consultoria existentes en la base de datos		
Atributo	**Tipo**	**Descripción**
idmedicion	integer	id de la medición
valor	integer	valor de la medición
indicador	integer	indicador al que pertenece la medición
fechaentrada	date	fecha de entrada de la medición a la BD

Descripción de la Tabla 13: "usuario"

Nombre: usuario		
Descripción: Almacena todas los usuarios existentes en la base de datos		
Atributo	**Tipo**	**Descripción**
id	integer	guarda el id de la usuario
proyecto	integer	Id del proyecto al que pertenece
username	varChar	usuario de la persona
pass	varChar	guarda la contraseña de la persona
nombre	varChar	guarda el nombre de la persona
primerapellido	varChar	guarda el primer apellido de la persona
segundoapellido	varChar	guarda el segundo apellido de la persona

Descripción de la Tabla 14: "perspectiva"

Nombre: perspectiva

Descripción: Tiene las perspectivas existentes en la base de datos		
Atributo	**Tipo**	**Descripción**
idperspectiva	integer	id de la perspectiva
nombre	varChar	nombre de la perspectiva
descripción	varChar	describe la función de la perspectiva

Descripción de la Tabla 15: "proyecto"

Nombre: proyecto		
Descripción: Almacena todos los proyectos del centro de consultoria existentes en la base de datos		
Atributo	**Tipo**	**Descripción**
idproyecto	integer	id del proyecto en cuestión
nombre	varChar	nombre del proyecto
descripción	varChar	describe la función del proyecto
consultoria	integer	almacena el id del centro de consultoria al que pertenece

3.10 Diagrama de Despliegue

El Diagrama de Despliegue es un tipo de diagrama del UML que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.

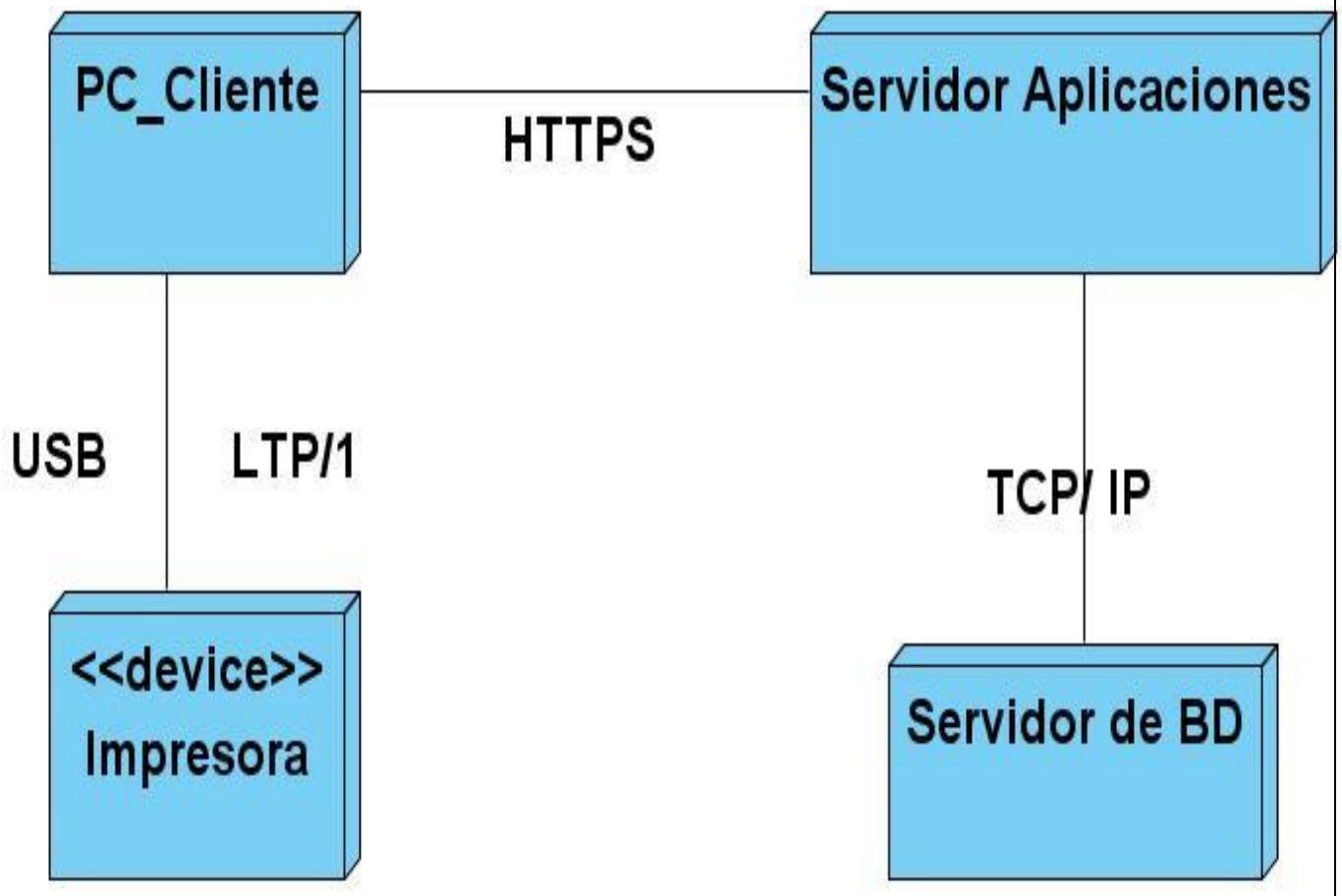


Figura 26 Diagrama de Despliegue

PC Cliente: Estación de trabajo desde donde interactúa el cliente con la aplicación.

Servidor Aplicaciones: Este contendrá la aplicación Web.

Servidor BD: Servidor donde se almacenarán los datos de la aplicación.

Impresora: Dispositivo mediante el cual se puede editar aquellos reportes que genera la aplicación.

3.11 Conclusiones del Capítulo:

En este capítulo se generaron los diferentes DCD, así como los diagramas de secuencia por cada uno de los escenarios de la aplicación, los cuáles brindan una panorámica de como el usuario interactúa con la misma, de igual manera se describieron las tablas de la BD. Se obtuvo también el diagrama de clases persistentes, el modelo de datos y el diagrama de despliegue que permite interpretar las interacciones entre los distintos componentes que forman parte de la aplicación.

Capítulo 4: Implementación del Sistema.

En el presente capítulo se desarrolla el modelo de implementación. Se muestran los diagramas de componentes, donde se detallan en componentes las diferentes clases y objetos utilizados en la implementación, para que se tenga una idea del funcionamiento de las mismas. Además se muestran ejemplos de código fuente de algunas clases que emplean el uso de los patrones de arquitectura y diseño.

4.1 Diagramas de Componentes.

"Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas"[30] (Ivar Jacobson, 2000)

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

Los diagramas de componentes fueron realizados por casos de uso, con el objetivo de facilitar la comprensión de éstos, se muestra como están distribuidos los componentes según el patrón arquitectónico Modelo-vista-Controlador (MVC) que utiliza Spring como paradigma en su organización interna.

A continuación una representación de los distintos diagramas de componentes utilizados, donde se muestran los que corresponden con los DCD mostrados en el capítulo tres, y el resto de estos diagramas de componentes están en los anexos.

[30] **Jacobson, Booch, Grady y Rumbaugh 2000.** "*El proceso unificado de desarrollo de software*".

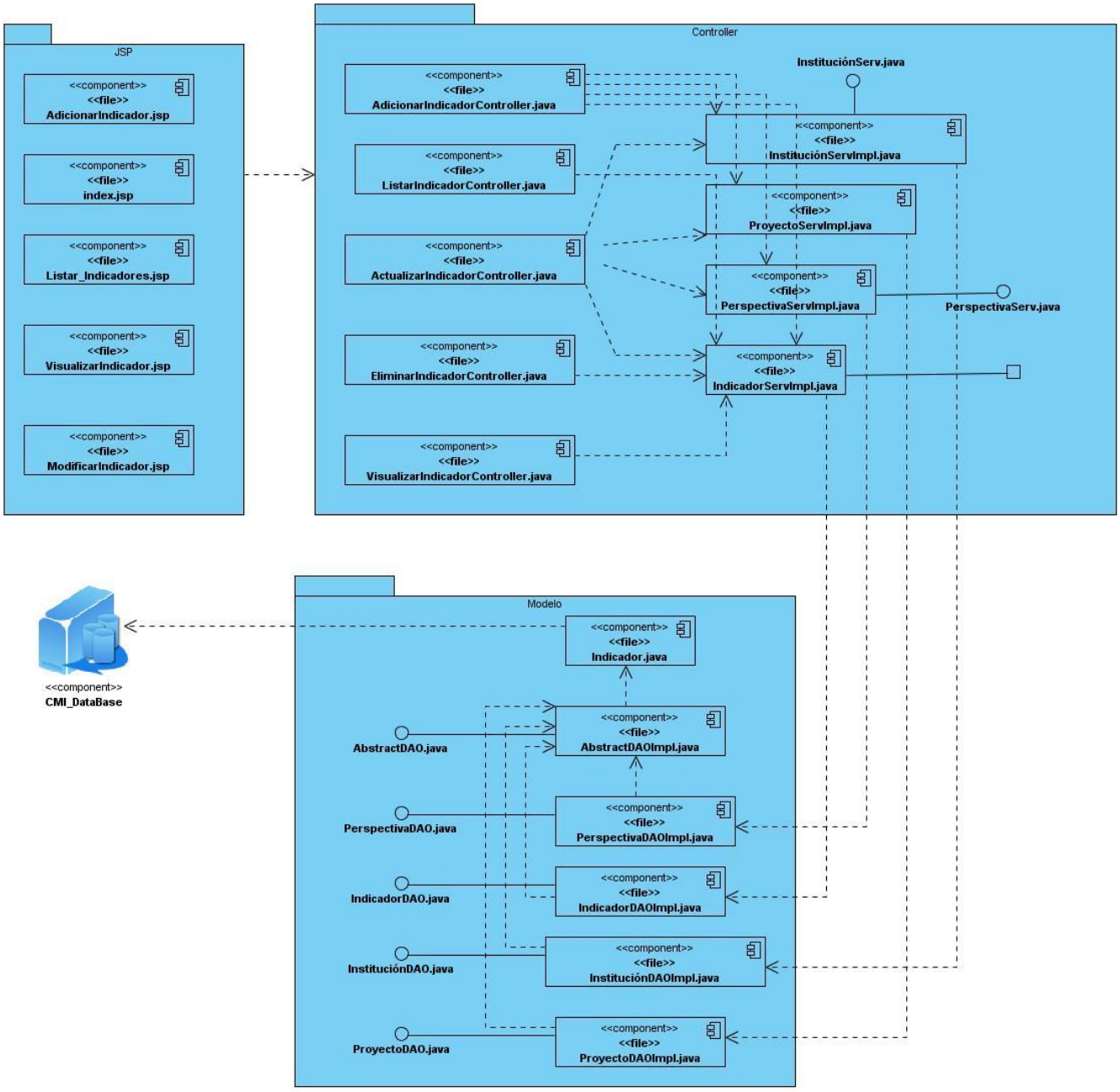


Figura 27 Diagrama de Componentes Gestionar_Indicador

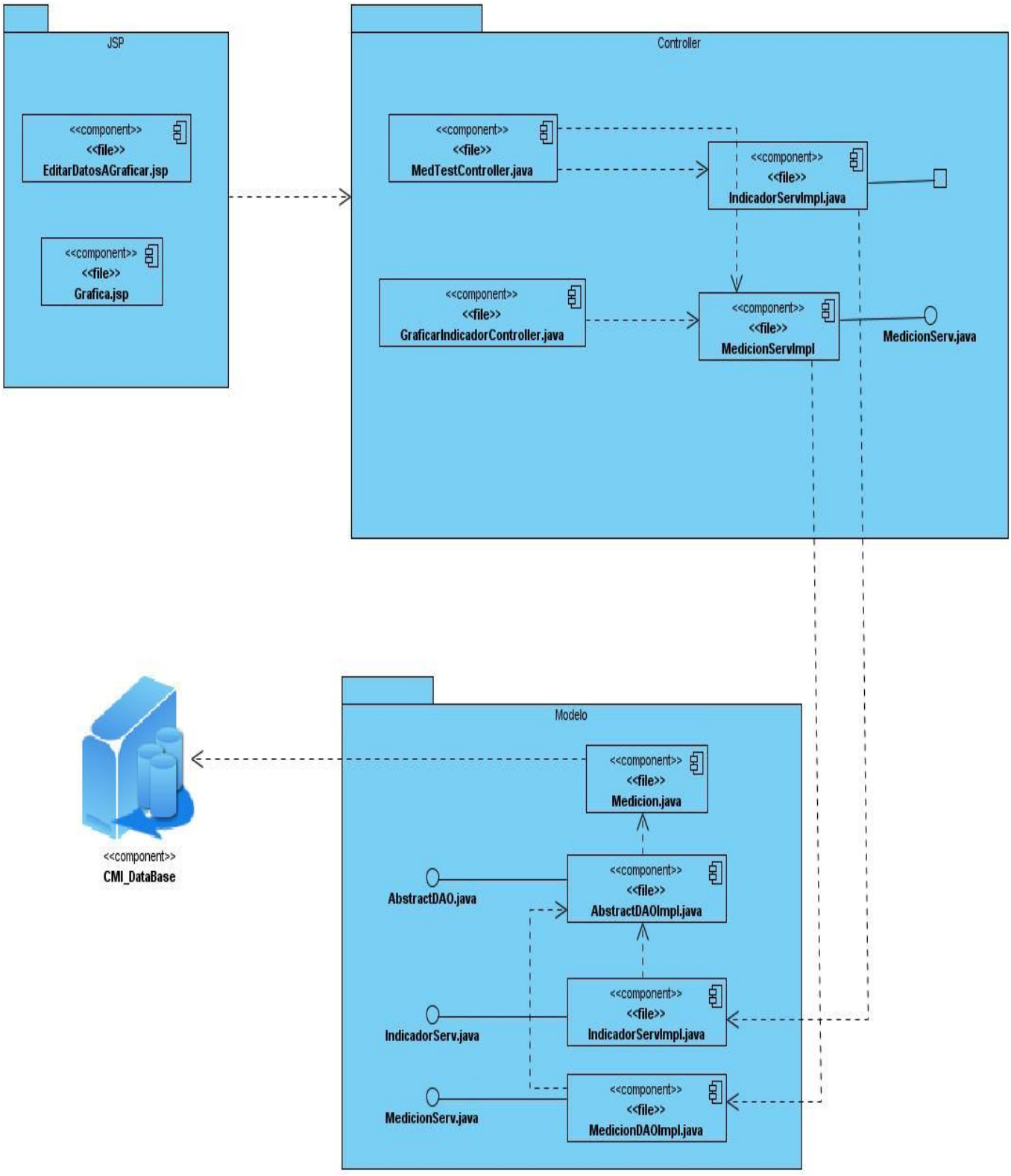


Figura 28 Diagrama de Componentes Consultar_Dashboard

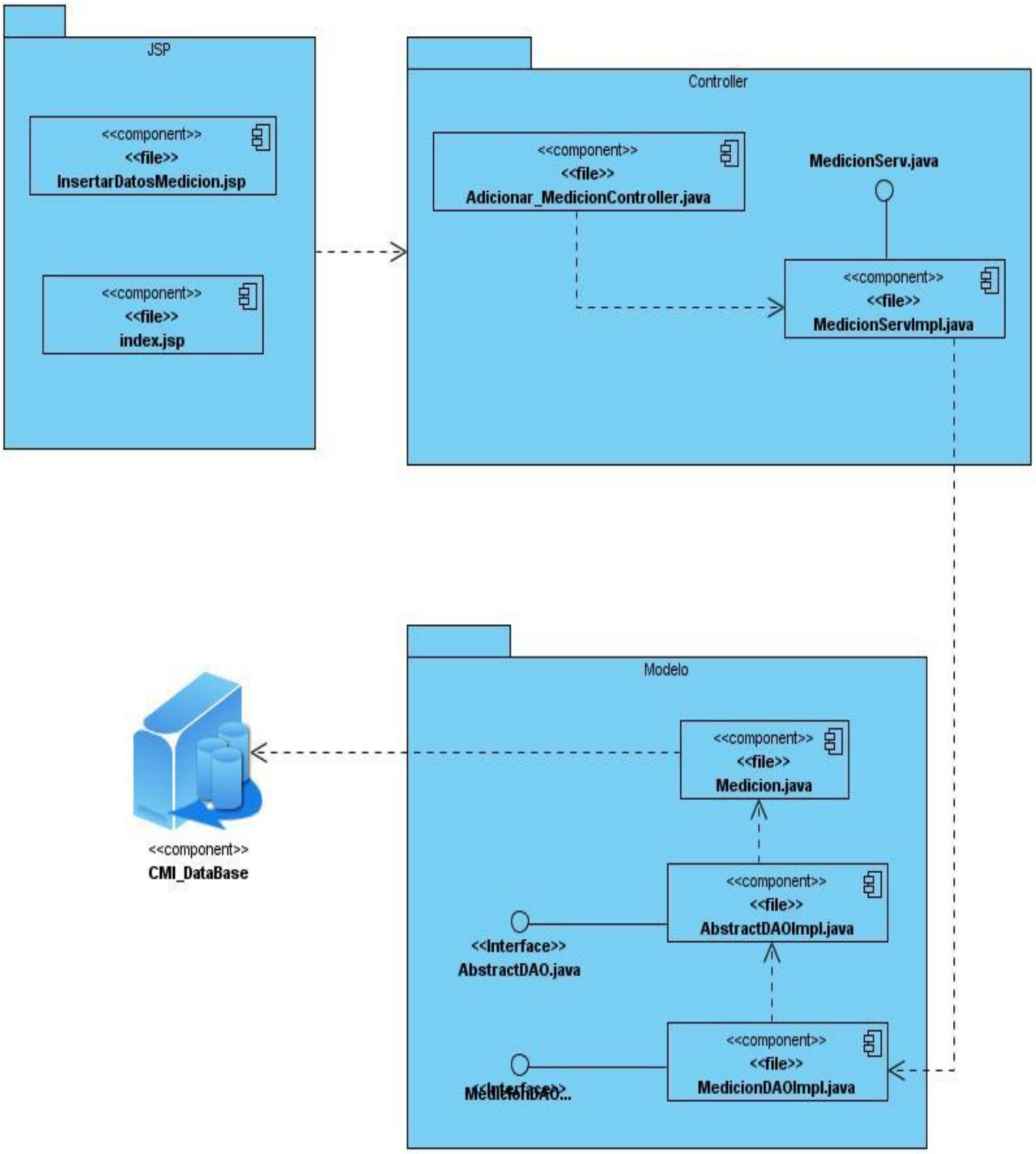


Figura 29 Diagrama de Componentes Realizar_Medición

4.2 Estándares de Codificación

Para la realización de este proyecto se siguieron ciertos estándares de codificación que a continuación serán expuestos:

Tabla 16 Estándares de Codificación Utilizados:

Indicador	Reglas de la nomenclatura utilizada	Ejemplo
Clases	Los nombres utilizados para las clases serán palabras completas, solo en letras mayúsculas y minúsculas, donde la primera letra de cada palabra será en mayúscula y el resto en minúsculas. Los nombres de las clases serán simples, descriptivos, y se podrán utilizar acrónimos o abreviaturas.	public class Indicadores public class IndicadorServ
Interfaces	Los nombres de interfaces deben seguir las mismas reglas indicadas para las clases.	
Métodos	Los métodos deben ser verbos, siguiendo la misma estructura que los nombres de las clases, y la primera letra de cada una de las palabras internas en mayúscula.	getIndicadorServ(); voidEliminar(T t);
Variables	Las variables deben estar escritas en letras minúsculas independientemente del número de palabras que estas contengan utilizando su respectivo tipo de datos.	int totalbuenas

4.2.1 Ejemplos de códigos de algunas de las funcionalidades utilizadas en la aplicación.

A continuación imágenes del código utilizado para la funcionalidad ***GraficarIndicador:***

```
package Controller;

import java.awt.BasicStroke;

public class GraficarIndicador{

    IndicadorServ indicadorServ;
    MedicionServ medicionServ;

    public MedicionServ getMedicionServ() {
        return medicionServ;
    }

    public void setMedicionServ(MedicionServ medicionServ) {
        this.medicionServ = medicionServ;
    }

    public IndicadorServ getIndicadorServ() {
        return indicadorServ;
    }

    public void setIndicadorServ(IndicadorServ indicadorServ) {
        this.indicadorServ = indicadorServ;
    }
```

Figura 30 Imagen de fragmento de código utilizado

```
public static String graficar(String indicador, List<Medicion> mediciones,
        String path) throws IOException {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    SimpleDateFormat formato = null;
    String fecha = null;
    Medicion aux = null;
    for (int i = 0; i < mediciones.size(); i++)
        for (int j = i + 1; j < mediciones.size(); j++)
            if (mediciones.get(i).getFechaentrada().after(
                    mediciones.get(j).getFechaentrada())) {
                aux = mediciones.get(i);
                mediciones.set(i, mediciones.get(j));
                mediciones.set(j, aux);
            }

    for (int i = 0; i < mediciones.size(); i++) {
        Medicion medicion = mediciones.get(i);
        formato = new SimpleDateFormat("dd/MM/yyyy");
        fecha = formato.format(medicion.getFechaentrada());
        dataset.addValue(medicion.getValor(), fecha, fecha);
    }
    JFreeChart chart = ChartFactory.createBarChart3D(indicador,
            "Fechas de Venta", "Valor de costo", dataset,
            PlotOrientation.VERTICAL, true, true, false);
```

Figura 31 Imagen de fragmento de código utilizado

```
    CategoryPlot plot = chart.getCategoryPlot();
    plot.setForegroundAlpha(0.7f);                // da nitidez al color de las barras
    plot.setRangeGridlinePaint(Color.black);      // tonifica el color de las
    plot.setOutlinePaint(Color.BLACK);            // lineas discontinuas detras
    plot.setWeight(100);                          //    de las graficas

    //Se coloca el rango que tendrá el eje Y para mostrar enteros solamente
    final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
    rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());

    //Permite pintar las lineas de los bordes de las barras del grafico
    BarRenderer3D renderer = (BarRenderer3D) plot.getRenderer();
    renderer.setDrawBarOutline(true);
    // set up gradient paints for series...
    GradientPaint gp0 = new GradientPaint(0.0f, 0.0f, Color.blue, 0.0f,
            0.0f, Color.blue);
    renderer.setSeriesPaint(0, gp0);
    renderer.setMaximumBarWidth(0.20);
    CategoryAxis domainAxis = plot.getDomainAxis();
    domainAxis.setCategoryLabelPositions(CategoryLabelPositions
            .createUpRotationLabelPositions(Math.PI / 4.0));
    String fileName = indicador + "" + System.currentTimeMillis();

    File file = new File(path + System.getProperty("file.separator")
            + fileName);

    ChartUtilities.saveChartAsJPEG(file, chart, 780, 412);

    return fileName;
}
```

Figura 32 Imagen de fragmento de código utilizado

A continuación se evidencia cómo funciona el método **onSubmit** de la clase ModificarIndicadorController, este método es llamado después de que ocurrió la validación y en el

objeto BindException no hubo ningún error, se ejecuta este método, el cual realiza la acción que se desea después que el usuario envíe el formulario, el objeto Command es el objeto que contiene todos los datos entrado desde el formulario y se castea para que se comporte según el tipo de objeto.

```
@Override
protected ModelAndView onSubmit(HttpServletRequest request,
        HttpServletResponse response, Object command, BindException errors)
        throws Exception {
    String mensaje = "Indicador insertado correctamente";
    if(update){
        mensaje = "Indicador actualizado correctamente";
    }
    update=false;
    String idPerspectiva = request.getParameter("idperspectiva");
    String idConsult = request.getParameter("idconsultoria");
    String idProyect = request.getParameter("idproyecto");
    String var = "seleccione";
    int idP=Integer.parseInt(idPerspectiva);
    Institucion institucion= null;
    Proyecto pro = null;

    if( idConsult.equals(var)){
        institucion=null;
    }
    else {
        institucion = institucionServ.ObtenerPorId(Integer.parseInt(idConsult));
    }

    if(idProyect .equals(var)){
        pro= null;
    }
    else {
        pro = proyectoServ.ObtenerProyecto(Integer.parseInt(idProyect));
    }

    Perspectiva perspectiva = perspectivaServ.ObtenerPorId(idP);
    Indicadores indicador = (Indicadores) command;
    indicador.setFormamedicion(request.getParameter("formamedicion"));
    indicador.setFrecuenciamedicion(request.getParameter("frecuenciamedicion"));
    indicador.setInstitucion(institucion);
    indicador.setProyecto(pro);
    indicador.setTipo(request.getParameter("tipo"));
    indicador.setNombre(request.getParameter("nombre"));
    indicador.setPerspectiva(perspectiva);
    indicador.setDescripcion(request.getParameter("descripcion"));
    indicadorServ.SalvarInd(indicador);
    RedirectView re = new RedirectView(getSuccessView());
    return new ModelAndView(re, "mensaje", mensaje);
}
```

Figura 33: Imagen de fragmento de código utilizado

4.2.2 Cómo se logró graficar en la Tesis

Para graficar como tal se utilizaron clases de la librería JFreeChart en aras de lograr realizar los gráficos con mayor precisión, se utilizaron las clases ChartFactory para definir el tipo de gráfico que se iba a utilizar así como algunos valores que tendría, ejemplo el ploteo (orientación (Vertical u Horizontal)), etiquetas para definir los ejes, y para decir si se iba o no a utilizar leyendas entre otras; se utilizaron otras clases como CategoryPlot también de JFreeChart que permitió a través de la llamada a objetos adaptar lo que sería la imagen de la gráfica (dígase color de las líneas y otros). Finalmente se utilizaron así mismo otras clases de la potente librería para customizar el gráfico y obtener una imagen más agradable a la vista del usuario.

4.2.3 Cómo se creó el gráfico que permitió pintar. Utilizando createBarChart3D

Se creó un objeto de tipo JFreeChart al cual se le dijo con qué tipo de gráfico pintaría, el que a su vez fue pedido a la clase ChartFactory de la librería JFreeChart, se le pasó entonces como parámetros los valores que utilizaría este gráfico para ser creado, que serían:

- El indicador al cual se le quiere hacer el gráfico.
- Lo que mediría el gráfico como tal (o sea valor por el eje Y, y fecha por el eje X).
- El objeto contenedor de los datos (dataset) que se pintarían, pertenecientes a la clase que se estaba utilizando para pintar el tipo de gráfico específico (en este caso CategoryDataset que es la clase que permite pintar gráficos de barras).
- El ploteo, que sería el encargado de definir la orientación de la gráfica (horizontal o vertical).
- Así como se le indicó si la gráfica tendría leyenda o no y si tendría dirección URL…entre otras

Ejemplo de código de cómo se creó uno de los tipos gráficos utilizados**:**

```
,
JFreeChart chart = ChartFactory.createBarChart3D(indicador,
        "Fechas de Venta", "Valor de costo", dataset,
        PlotOrientation.VERTICAL, true, true, false);
```

4.3 Validación de la Propuesta.

Una vez se terminó de implementar la herramienta propuesta, se constató que la misma cumplió todos los requerimientos funcionales y no funcionales presentados por el cliente, donde el principal RF fue el de Graficar Indicador, con el mismo mediante graficas se pudo dar seguimiento al estado de los diferentes indicadores. Corroborando lo anterior, se realizaron pruebas de sistema, en las que se evidenció el cumplimiento de la herramienta con los requerimientos del cliente.
Mediante prueba de aceptación se pudo validar que el software cumplió la calidad mínima requerida, lo que pudo constatarse a través de la satisfacción mostrada por el cliente.

4.4 Conclusiones del Capítulo

En el siguiente capítulo se identificaron los diagramas de componentes, lo cuáles posibilitaron mayor acercamiento y entendimiento del trabajo, se evidenciaron los distintos estándares de codificación utilizados para el trabajo de la aplicación, y se validó la propuesta del sistema.

Conclusiones Generales

Una vez finalizado el Trabajo de Diploma y luego de haber desplegado la herramienta propuesta se le dieron cumplimiento a todos los requisitos funcionales y no funcionales identificados en las entrevistas realizadas al cliente, lo que contribuyó a un mejor diseño e implementación de la aplicación, logrando así una primera versión del producto.

Se implementó un software de Cuadro de Mando Integral que dotó al Centro de Consultoría de una herramienta inteligente que le permitirá la gestión de sus indicadores de forma automatizada.

De esta forma se cumplió el objetivo general así como los objetivos específicos de la investigación del presente trabajo.

Recomendaciones

Dado que el tema de Cuadro de Mando Integral es sumamente amplio, importante e innovador, y que la aplicación solo se centró en las principales exigencias del cliente, se recomienda realizarle versiones posteriores al software donde implementen nuevas funcionalidades o modifiquen algunas de las existentes (ejemplo poder graficar con un mayor número de tipos de gráfico) de acuerdo a los distintos requerimientos donde será utilizado el software, lo que posibilite que este sea desplegado no solamente en el Centro de Consultoria de la UCI sino también en otras áreas de la Universidad.

Se recomienda igualmente se incluyan mecanismos basados en la inteligencia artificial para analizar la relación entre los indicadores y dotar a la herramienta de capacidad de predicción. Que las empresas cubanas implanten en mayor medida software referentes a CMI, lo que permita contribuir a un mejor proceso de toma de decisiones estratégicas en las mismas, y elevar el prestigio y la competitividad del sistema empresarial cubano.

Referencias Bibliográficas

"Balanced Scorecard: Measures that Drive Performance", Harvard Business Review. **Kaplan, Robert y Norton, David. 1992.** Boston : s.n., 1992.
A Balancing Act. **Rhom, Howard. 1992.** 1992.
Barrios, Armando González. 2001. *El Cuadro de Mando Integral en el Sistema Portuario Español.* 2001.
Batllori, Aguila. 1998. *La aplicación del Cuadro de Mando Integral a una Empresa Industrial.* 1998.
Candia, Diaz. 2001. *Tablero de comando en una empresa forestal PYME.* 2001.
Customer-Profiability Analysis. **Cooper, Robin y Kaplan, Robert. 1998.**
Dávila, Antonio. 1999. *Nuevas herramientas de control: El Cuadro de Mando Integral.* 1999.
Epstein, Richard y Westbrook, George. 2001. *Action-Profit-Linkage Model.* 2001.
Heskett, James, Sasser, Leb y Schlesinger, Leonard. 1997. *Service-Profit Chain.* 1997.
Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El proceso unificado de desarrollo de software.* . s.l. : Pearson Education, 2000.
Kaplan, Robert y Norton, David. 1996. *Using the Balanced Scorecard as a Strategic Management System.* 1996.
Larman, Craig. 1999. *UML y Patrones, Introducción al Análisis y Diseño Orientado a Objeto.* 1999.
Maldonado, Fernando Guzman. 2002. *Planeación estratégica: aplicación en la comisión federal de electricidad, de dirección de operaciones, subdirección de transmisión, transformación y control.* 2002.
Martínez, Ricardo. 2001. *Balancead Scorecard - Sistema de comunicación, control y aprendizaje estratégico.* 2001.
Pagan, Elda Conde. 2006. *El Cuadro de Mando Integral en Organizaciones no lucrativas.* 2006.
1998. *Resolución Económica del V Congreso del Partido Comunista de Cuba.* 1998.
Return on Quality. **Rust, Roland y Zahorit, Jhon. 1999.** 1999.
Sánchez, Carlos. 2004. *La Programación Extrema.* 2004.
Xodo y Nigro. 2001. *El CMI en el Ámbito Comunal.* 2001.
R Kasman, Bass Clements. *Software Architecture in Practice.* s.l.: Addison-Wesley, 1998.
Boston : s.n., 1992.
A. Silberschatz, H. F. Korth, S. Susarshan. 2006. *Fundamentos de bases de datos.* s.l. : McGraw-Hill, 2006. 994.
Argentino, Periódico. 2007. El Diario del Centro del País. [En línea] 2007.
El Cuadro de Mando Integral como herramienta de gestión ***Norton y Kaplan, 2000****.*
—. *Programación Java Server con J2EE Edición 1.3.*
El Balanced Scorecard ayudando a implantar la estrategia. **Fernández, Alberto. marzo 2004.** 42, marzo 2004.
http://www.deinsa.com/delphos.html. [En línea]
Kent, Beck. 2000. *Programación Extrema.* 2000.
Pressman, R. S. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.
Resolución Económica del V Congreso del Partido Comunista de Cuba. **1998.** 1998.
Rhom, Howard. 1992. *A Balancing Act.* 1992.
Silberschatz, A.
Sixtina. http://www.sixtina.com.ar. [En línea]
Wellicki, L. 2007. Patrones y Antipatrones: una introduccion. [En línea] 2007. http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ 3317/default.aspx#M15.

Bibliografía

1. Balanced Scorecard: Measures that Drive Performance
   http://www.mantenimientomundial.com/sites/mmnew/bib/notas/Amendola1.pdf
2. Using the Balanced Scorecard as a Strategic Management. System
   http://www.ceprede.com/forest/reuniones/docs/280405/acastilla_abr05.pdf
3. A Balancing Act
   http://www.managinf.com/sig.pdf
4. Balancead Scorecard - Sistema de comunicación, control y aprendizaje estratégico
   http://www.arearh.com/rrhh/balanced_scorecard2.htm
5. Service-Profit
   http://www.favaneves.org/arquivos/artigoextra6-5.pdf
6. Action-Profit-Linkage Model
   http://www.qa.au.edu/page2/research/BSCLinkingActionsToProfits.pdf
7. El cuadro de mando integral; Un instrumento de control
   http://www.monografias.com/trabajos43/cuadro-mando-integral/cuadro-mando-integral.shtml
8. Cuadro de Mando Integral. J. E. Pereira
   http://www.mercadeo.com/41_scorecard.htm
9. El Cuadro de Mando Integral en organizaciones no lucrativas
   http://www.monografias.com/trabajos31/cuadro-mando-integral/cuadro-mando-integral.shtml
10. Elda Conde Pagan “El Cuadro de Mando Integral en Organizaciones no lucrativas”
    http://www.gestiopolis.com/canales6/fin/herramienta-financiera.htm
11. Introducción a Extreme Programming http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf
12. El Proceso Unificado de Desarrollo de Software
    http://bibliodoc.uci.cu/pdf/reg00060.pdf

13. UML y Patrones Introducción al Análisis y Diseño Orientado a Objetos
http://bibliodoc.uci.cu/pdf/reg00061.pdf
14. Módulo Gestión de estrategias e indicadores del Cuadro de Mando Integral.
http://biblioteca.uci.cu/sbd/biuci/index.html
15. Arquitectura y Patrones de Diseño
http://teleformacion.uci.cu/course/view.php?id=43.
16. Delphos Arquitectura - Presentation Transcript
http://www.slideshare.net/controlgestionarmada/delphos-arquitectura
17. Soluciones Integradas S.A Balanced Scorecard
http://www.solintegra.com/volantes/Delphos%20tiro.pdf
18. Sixtina Balanced Scorecard Online
http://www.aicon.es/esp/030102.htm
19. ¿Qué es PostgreSQL?
http://www.sraoss.co.jp/PostgreSQL/introduction-en.php
20. Visual Paradigm para UML
http://www.todoprogramas.com/programalinux/visualparadigmforuml
21. Rational Unified Process
http://www.rational.com.ar/herramientas/rup.html
22. Programación Extrema http://www.clubdevelopers.com/prog/articulos/xp/downloads/xp.pdf
23. Rational Rose http://www.slideshare.net/vivi_jocadi/rational-rose
24. http://www.aulaclic.es/sql (Integridad Referencial)
25. http://msdn.microsoft.com/es-es/library/ms174875.aspx
26. http://msdn.microsoft.com/es-es/library/ms165911.aspx
27. http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml (Herramientas CASE)
28. http://www.definicion.org/indicadores-financieros
29. **Deinsa.** http://www.deinsa.com/delphos.html
30. http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/

31. Kaplan Robert y Norton David - "Using the Balanced Scorecard as a Strategic Management System, 1996", Harvard Business Review (Enero – Febrero 1996).

32. Rohm, Howard "A Balancing Act", Performe Magazine, vol. 2, issue 2.

33. Dávila, Antonio**.** ¨*Nuevas herramientas de control: El Cuadro de Mando Integral". (septiembre 1999).*

34. Kaplan, Robert S. y Norton, David P., ¨Balanced Scorecard: Measures that Drive Performance¨, Harvard Business Review, Boston, enero-febrero de 1992.

35. Ricardo Martínez "Balancead Scorecard - Sistema de comunicación, control y aprendizaje estratégico".

36. Resolución Económica del Partido Comunista de Cuba Editora Política, La Habana 1998 p.25

37. Sitio de la Asignatura Ingeniería de Software 2. *Conferencia 2: Arquitectura y Patrones de Diseño*

38. Teleformacion.uci.cu [En línea]

39. http://teleformacion.uci.cu/course/view.php?id=42/Introducción a la ingeniería de software.

40. Francisco Martínez Fernández "El cuadro de mando integral; Un instrumento de control"**.**

41. Craig Larman "UML y Patrones, Introducción al Análisis y Diseño Orientado a Objeto"

42. Kent Beck "Programación Extrema".

43. Ayuda extendida del Rational Rose Enterprise Edition 2003.

44. González Barrios, A. (2001). El CMI en el sistema portuario español: Alineando objetivos y estrategias de negocio. Dossier del II Congreso Internacional de Tablero de Comando (noviembre), Argentina.

45. Jaramillo Martínez, A. A. (2001). Aplicación del Balanced Scorecard en el sector educativo. Dossier del II Congreso Internacional de Tablero de Comando (noviembre), Argentina.

46. Díaz Candia, L. A. (2001). Tablero de comando en una empresa forestal PYME. Dossier del II Congreso Internacional de Tablero de Comando (noviembre), Argentina.

47. Xodo, D. & Nigro, O. (2001). El CMI en el Ámbito Comunal. Su aplicación a la actividad turística. Dossier del II Congreso Internacional de Tablero de Comando (noviembre), Argentina.

48. Martínez Rivadeneira, R. (2001). El Balanced Scorecard aplicado en áreas de logística.

49. Guzmán Maldonado, F. (2002). Planeación estratégica: aplicación en la comisión federal de electricidad, de dirección de operaciones, subdirección de transmisión, transformación y control (Premio Scorecard 2001). México.

50. Águila Batllori, S. (1998). La aplicación del Cuadro de Mando Integral a una Empresa Industrial. Harvard Deusto Finanzas & Contabilidad. No. 21 (enero-febrero). España. pp.27-34.

51. Sánchez, Carlos. [En línea] Universidad de Coruña, 28 de septiembre de 2004. [Citado el: 6 de marzo de 2008.] http://oness.sourceforge.net/proyecto/html/ch05.html. 4

52. Metodología para la Implantación de un Sistema de Medición del Rendimiento Empresarial. [Citado en el 2007].

53. UML y Patrones. Introducción al análisis y diseño orientado a objetos - Larman - Prentice Hall

ANEXOS CAPITULO I

Figura 30 Cuadro de Mando Integral y Perspectivas

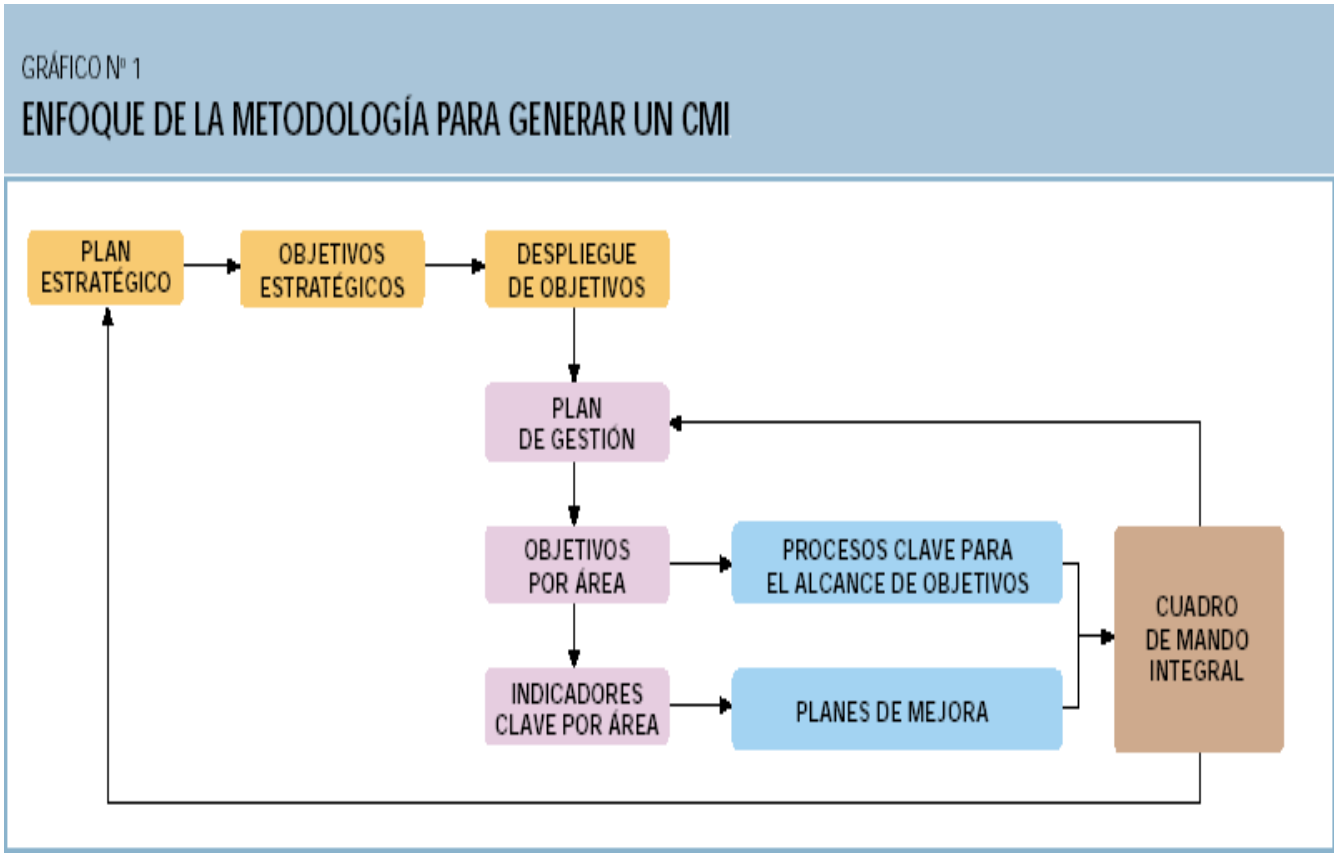


Figura 31 Enfoque de la Metodología para generar un CMI

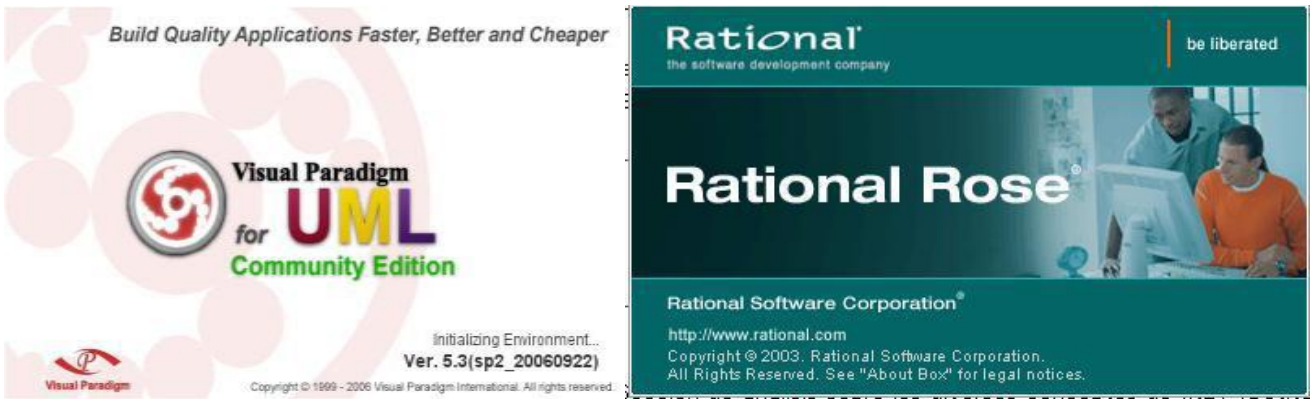


Figura 32 Herramientas Case

Figura 33 Gráfica BarChart de ejemplo

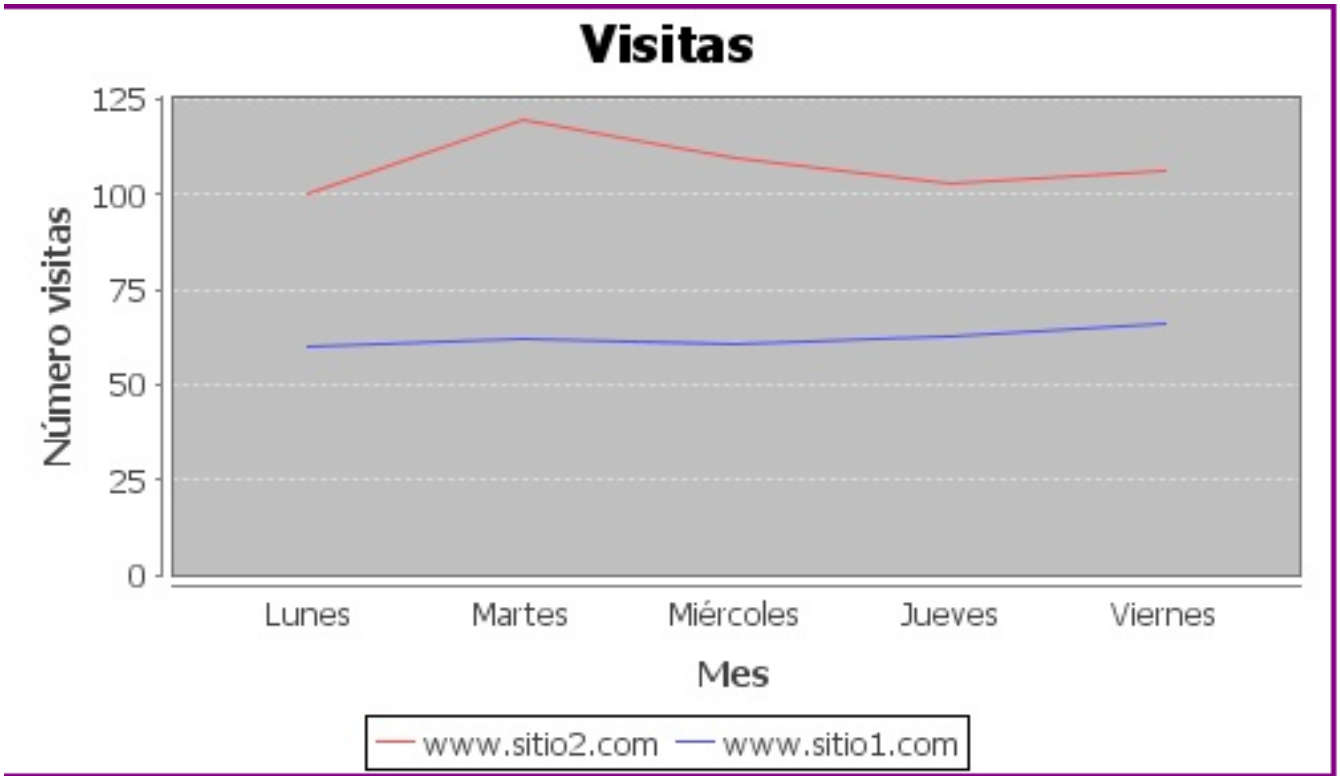


Figura 34 Gráfica LineChart de ejemplo

ANEXOS CAPITULO II

Descripción de los restantes Casos de Uso del sistema:

Tabla 18: Descripción del CU: Gestionar Perspectiva

Caso de Uso	
CU_4	Gestionar Perspectiva
Propósito	El administrador gestionará los datos de las perspectivas.
Actores	Administrador.
Resumen	El administrador puede adicionar, modificar y eliminar los datos de una perspectiva haciendo las respectivas actualizaciones en la BD.
Pre-Condiciones	Para poder gestionar una perspectiva debe existir al menos un Centro de Consultoría o un Proyecto registrado en la BD. Las perspectivas deben de estar listadas previamente.
Post-Condiciones	Se actualiza la BD.
Referencias	R.F_10, R.F_11, RF_12.
Prioridad	Crítico
Sección 1: Adicionar Perspectiva	
Acción del Actores	**Respuesta del Sistema**
1. El administrador selecciona la opción de Adicionar Perspectiva.	1.2 La aplicación muestra una interfaz para que el administrador introduzca los datos de la perspectiva.
1.3 El administrador introduce en el sistema los datos de la perspectiva.	1.4 La aplicación comprueba que los datos estén escritos correctamente y que la perspectiva no exista en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos de la perspectiva se han adicionado satisfactoriamente.
Flujos alternativos	
Acción del Actores	**Respuesta del Sistema**
	1.4.1 Si los datos de la perspectiva no están escritos correctamente o la perspectiva ya existe en la BD el sistema muestra un error y regresa al

	paso 1.2
Prototipo no Funcional de Interfaz de usuario	
	
Sección 2: Eliminar Perspectiva.	
Acción del Actor	**Respuesta del Sistema**
2.1 El administrador selecciona la opción Eliminar Perspectiva a partir de la interfaz Listar Perspectiva.	2.2 El sistema elimina la perspectiva de la BD y la actualiza.
	2.3 El sistema muestra la interfaz de Listar Perspectiva actualizada.
Sección 3: Modificar Perspectiva	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador selecciona la opción Modificar Perspectiva a partir de la interfaz Listar Perspectivas.	3.2 El sistema muestra un formulario con los datos de la perspectiva antes de ser modificada.
3.3 El administrador modifica los datos de la perspectiva.	3.4 El sistema actualiza la BD con los nuevos datos de la perspectiva.
	3.5 Se actualiza la interfaz Listar Perspectivas.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_5	Gestionar Indicador
Propósito	El Administrador gestionará los datos referentes a un indicador.
Actores	Administrador
Resumen	El administrador puede adicionar, modificar y eliminar los datos de un indicador haciendo las respectivas actualizaciones en la BD.
Pre-Condiciones	Debe existir al menos un Centro de Consultoría o un Proyecto registrado en la BD. A excepción del Adicionar y Visualizar Indicador en los restantes dos escenarios, los indicadores deberán estar previamente listados.
Post-Condiciones	Se actualiza la BD.
Referencias	R.F_13, R.F_14, RF_15.
Prioridad	Crítico

Sección 1: Adicionar Indicador	
Acción del Actores	**Respuesta del Sistema**
1.1 El Administrador selecciona la opción Adicionar Indicador.	1.2 El sistema muestra una interfaz para que el Administrador introduzca los datos del indicador que será adicionado.
1.3 El Administrador introduce los datos del nuevo indicador.	1.4 La aplicación comprueba que los datos estén escritos correctamente y que el indicador no exista en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos del indicador se han adicionado satisfactoriamente.

Flujos alternativos	
Acción del Actores	**Respuesta del Sistema**
	1.4.1 Si los datos del indicador no están escritos correctamente o el indicador ya existe en la BD el sistema muestra un mensaje de error y regresa al paso 1.2
Sección 2: Eliminar Indicador	
Acción del Actor	**Respuesta del Sistema**
2.1 El Administrador selecciona la opción de Eliminar Indicador a partir de la interfaz de Listar Indicador.	2.2 El sistema elimina los datos del indicador existentes en la BD y la actualiza.
	2.3 El sistema muestra la interfaz Listar Indicador actualizada.
Sección 3: Modificar Indicador	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador selecciona la opción Modificar Indicador a partir de la interfaz de Listar Indicador.	3.2 El sistema muestra un formulario con los datos del indicador antes de ser modificados.
3.3 El administrador modifica los datos del indicador.	3.3.1 El sistema muestra la interfaz Listar Indicadores actualizada.
Prototipo no Funcional de Interfaz de Usuario	

Tabla 19: Descripción del CU: Gestionar Institución

Caso de Uso	
CU_6	Gestionar Institución
Propósito	El administrador gestionará los datos de la Institución.
Actores	Administrador.
Resumen	El administrador puede adicionar, modificar y eliminar los datos de una Institución haciendo las respectivas actualizaciones en la BD.
Pre-Condiciones	A excepción del Adicionar Institución en los restantes tres escenarios, las instituciones deberán de estar listadas previamente.

Post-Condiciones	Se actualiza la BD.
Referencias	R.F_16, R.F_17, RF_18.
Prioridad	Crítico

Sección 1: Adicionar Institución	
Acción del Actores	**Respuesta del Sistema**
1. El administrador selecciona la opción de Adicionar Institución.	1.2 La aplicación muestra una interfaz para que el administrador introduzca los datos de la Institución.
1.3 El administrador introduce en el sistema los datos de la Institución.	1.4 La aplicación comprueba que los datos estén escritos correctamente y que la Institución no exista en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos de la Institución se han adicionado satisfactoriamente.
Flujos alternativos	
Acción del Actores	**Respuesta del Sistema**
	1.4.1 Si los datos de la Institución no están escritos correctamente o esta ya existe en la BD el sistema muestra un mensaje de error y regresa al paso 1.2
Sección 2: Eliminar Institución.	
Acción del Actor	**Respuesta del Sistema**
2.1 El administrador selecciona la opción Eliminar Institución a partir de la interfaz Listar Centro de Consultoría.	2.2 El sistema elimina los datos de la Institución existentes en la BD.
	2.3 El sistema muestra la interfaz Listar Instituciones actualizada.
Sección 3: Modificar Institución	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador selecciona la opción Modificar Institución a partir de la interfaz Listar Instituciones.	3.2 El sistema muestra un formulario con los datos de la Institución antes de ser modificados en la BD.
3.3 El administrador modifica los datos de la	3.4 El sistema actualiza los datos de la Institución

Institución y los envía.	en la BD.
	3.5 El sistema muestra la Interfaz Institución de forma actualizada.

Tabla 20: CU Visualizar Indicador

Caso de Uso	
CU_8	Visualizar Indicador
Propósito	Mostrar los datos específicos de un Indicador.
Actores	Usuario.
Resumen	El usuario puede visualizar los datos del indicador seleccionado.
Pre-Condiciones	El indicador debe estar en la lista de indicadores.
Post-Condiciones	Se visualizan los datos del Indicador seleccionado.
Referencias	RF_19.
Prioridad	Secundario

Acción del Actor	Respuesta del Sistema
1. El usuario selecciona la opción de Visualizar Indicador a partir de la interfaz Listar Indicador.	1.2 La aplicación muestra una interfaz con los datos del indicador seleccionado.
1.3 El usuario selecciona la opción regresar a la interfaz donde se encontraba.	1.4 El sistema muestra la interfaz seleccionada por el usuario.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_9	Visualizar Institución
Propósito	Mostrar los datos específicos de una Institución.
Actores	Usuario.
Resumen	El usuario visualiza los datos de la Institución seleccionada.
Pre-Condiciones	La Institución debe estar en la lista de Instituciones.
Post-Condiciones	Se visualizan los datos de la Institución seleccionada.
Referencias	RF_20.
Prioridad	Secundario

Acción del Actores	Respuesta del Sistema
1. El usuario selecciona la opción de Visualizar Institución a partir de la interfaz Listar Instituciones	1.2 La aplicación muestra una interfaz con los datos de la Institución seleccionada.

1.3 El usuario selecciona la opción regresar a la interfaz donde se encontraba.	1.4 El sistema muestra la interfaz seleccionada por el usuario.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_10	Visualizar Proyecto
Propósito	Mostrar los datos específicos de un Proyecto.
Actores	Usuario.
Resumen	El usuario puede visualizar los datos del proyecto seleccionado.
Pre-Condiciones	El proyecto debe estar en la lista de proyectos existentes en la BD.
Post-Condiciones	Se visualizan los datos del proyecto seleccionado por el usuario.
Referencias	RF_21.
Prioridad	Secundario

Acción del Actores	**Respuesta del Sistema**
1. El usuario selecciona la opción de Visualizar Proyecto a partir de la interfaz Listar Proyecto.	1.2 La aplicación muestra una interfaz con los datos de un proyecto específico.
1.3 El usuario selecciona la opción regresar a la interfaz donde se encontraba.	1.4 El sistema muestra la interfaz seleccionada por el usuario.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_11	Visualizar Perspectiva
Propósito	Mostrar los datos específicos de una Perspectiva.
Actores	Usuario.
Resumen	El usuario puede visualizar los datos de la perspectiva seleccionada.
Pre-Condiciones	La perspectiva debe estar en la lista de perspectivas existentes en la BD.
Post-Condiciones	Se visualizan los datos de la perspectiva seleccionada por el usuario.
Referencias	RF_22.
Prioridad	Secundario

Acción del Actores	**Respuesta del Sistema**
1. El usuario selecciona la opción de Visualizar Perspectiva a partir de la interfaz Listar Perspectiva.	1.2 La aplicación muestra una interfaz con los datos de la perspectiva seleccionada por el usuario.

1.3 El usuario selecciona la opción regresar a la interfaz donde se encontraba.	1.4 El sistema muestra la interfaz seleccionada por el usuario.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_12	Listar Instituciones
Propósito	Mostrar una lista de todas las Instituciones existentes en la BD.
Actores	Usuario.
Resumen	El usuario visualiza en una lista las Instituciones existentes en la BD.
Pre-Condiciones	Debe existir en la BD al menos una Institución.
Post-Condiciones	Se muestra un listado con las Instituciones existentes en la BD.
Referencias	RF_29, RF_30.
Prioridad	Crítico

Acción del Actores	**Respuesta del Sistema**
1. El usuario selecciona la opción de Listar Instituciones del menú principal.	1.2 La aplicación muestra una interfaz con una lista de todas las Instituciones existentes en la BD.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso	
CU_16	Listar Usuarios
Propósito	Mostrar una lista de todas las Personas existentes en la BD.
Actores	Usuario.
Resumen	El usuario visualiza en una lista todas las personas existentes en la BD.
Pre-Condiciones	Debe existir en la BD al menos una persona.
Post-Condiciones	Se muestra un listado con las personas existentes en la BD.
Referencias	RF_27, RF_28.
Prioridad	Crítico

Acción del Actores	**Respuesta del Sistema**
1. El usuario selecciona la opción de Listar Usuario del menú principal.	1.2 La aplicación muestra una interfaz con una lista de todos los Usuarios existentes en la BD.
Prototipo no Funcional de Interfaz de Usuario	

Caso de Uso		
CU_17	Listar Perspectivas	
Propósito	Mostrar una lista de todas las Perspectivas existentes en la BD.	
Actores	Usuario.	
Resumen	El usuario visualiza en una lista todas las perspectivas existentes en la BD.	
Pre-Condiciones	Debe existir en la BD al menos una perspectiva.	
Post-Condiciones	Se muestra un listado con las perspectivas existentes en la BD.	
Referencias	RF_31, RF_32.	
Prioridad	Crítico	
Acción del Actores		**Respuesta del Sistema**
1. El usuario selecciona la opción de Listar Perspectiva del menú principal.		1.2 La aplicación muestra una interfaz con una lista de todas las perspectivas existentes en la BD.
Prototipo no Funcional de Interfaz de Usuario		

Caso de Uso		
CU_19	Listar Proyecto	
Propósito	Mostrar una lista de todos los proyectos existentes en la BD.	
Actores	Usuario.	
Resumen	El usuario visualiza en una lista todos los proyectos existentes en la BD.	
Pre-Condiciones	Debe existir en la BD al menos un proyecto.	
Post-Condiciones	Se muestra un listado con todos los proyectos existentes en la BD.	
Referencias	RF_33, RF_34.	
Prioridad	Crítico	
Acción del Actores		**Respuesta del Sistema**
1. El usuario selecciona la opción de Listar Proyectos del menú principal.		1.2 La aplicación muestra una interfaz con una lista de todos los proyectos existentes en la BD.
Prototipo no Funcional de Interfaz de Usuario		

ANEXOS CAPITULO III

Descripción del resto de los patrones utilizados:

Patrón Abstract Factory

Este patrón se evidencia de forma directa para utilizar objetos de clases distintas sin mezclarlas entre sí, lo que se puede apreciar en la clase *Adicionar_IndicadorController* que utiliza objetos pertenecientes a otras clases. Ejemplo:

public class Adicionar_IndicadorController
IndicadorServ *indicadorServ****;***
PerspectivaServ *perspectivaServ****;***
CentroConsultoriaServ *centroConsultoriaServ****;***
ProyectoServ *proyectoServ****;***

Patrón Creador

Este patrón se pone de manifiesto una vez que se necesita realizar herencia en la aplicación, por ejemplo cuando decimos que una clase extiende de otra, esta que hereda utiliza los objetos del padre por lo que se produce entonces la utilización de este patrón.

Patrón Adapter (Adaptador)

Este patrón es el encargado de adaptar las interfaces para que puedan ser utilizadas por otras clases luego ya que de no ser así estas últimas no podrían hacerlo. Ejemplo:
La clase testMed.jsp que diseña la interfaz de la página que va a utilizar como tal, permite que luego esta pueda ser accedida sin problemas desde la CMI-servlet.xml mediante el mapeo:
<prop key="testMed.htm">ViewNameController</prop>

A continuación se muestra el resto de los diagramas:

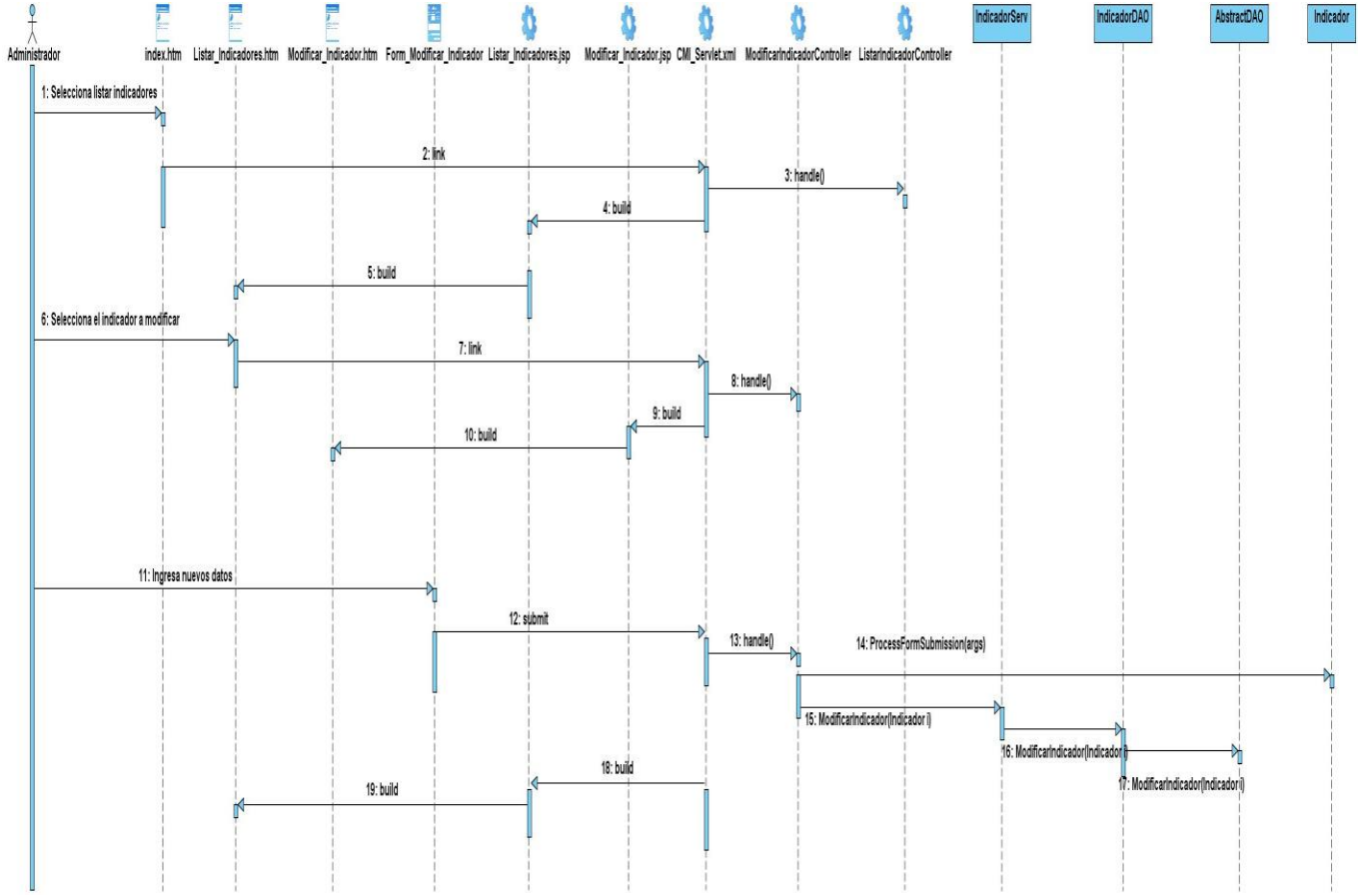


Figura 35 Diagrama de Secuencia de Gestionar Indicador, escenario “Modificar Indicador”

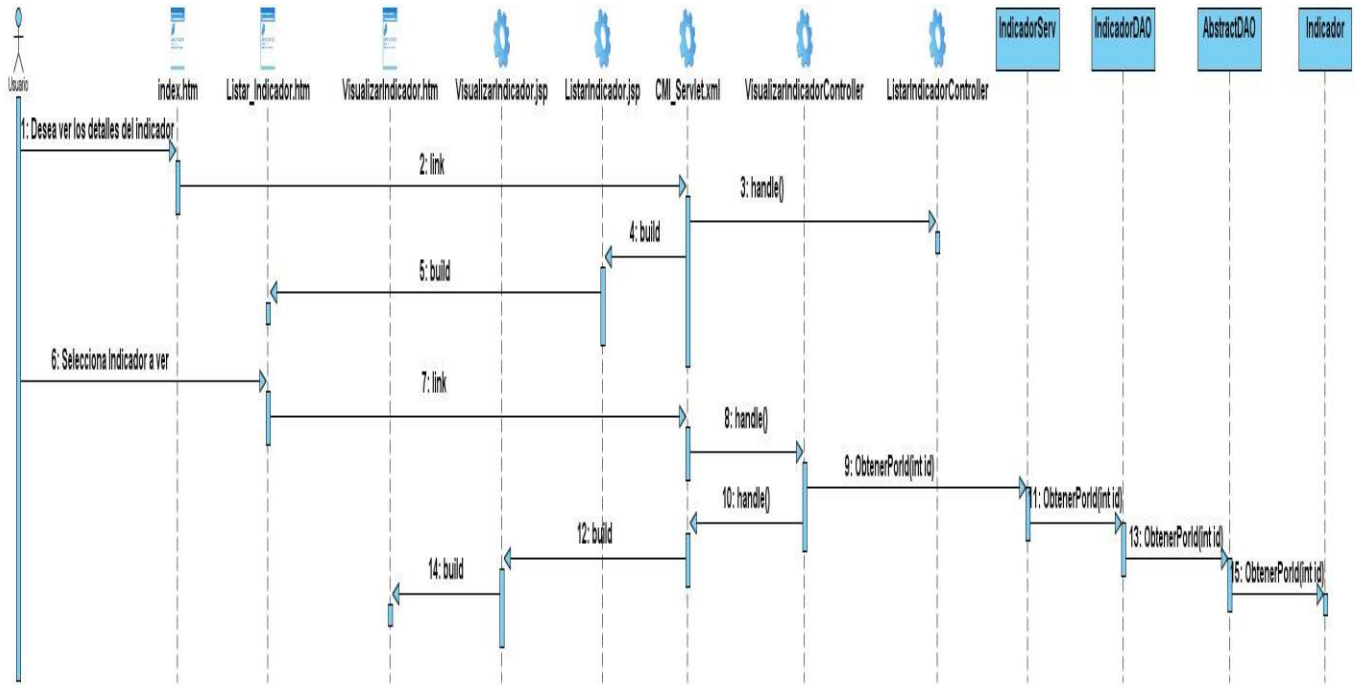


Figura 36 Diagrama de Secuencia de Gestionar Indicador, escenario “Visualizar Indicador”

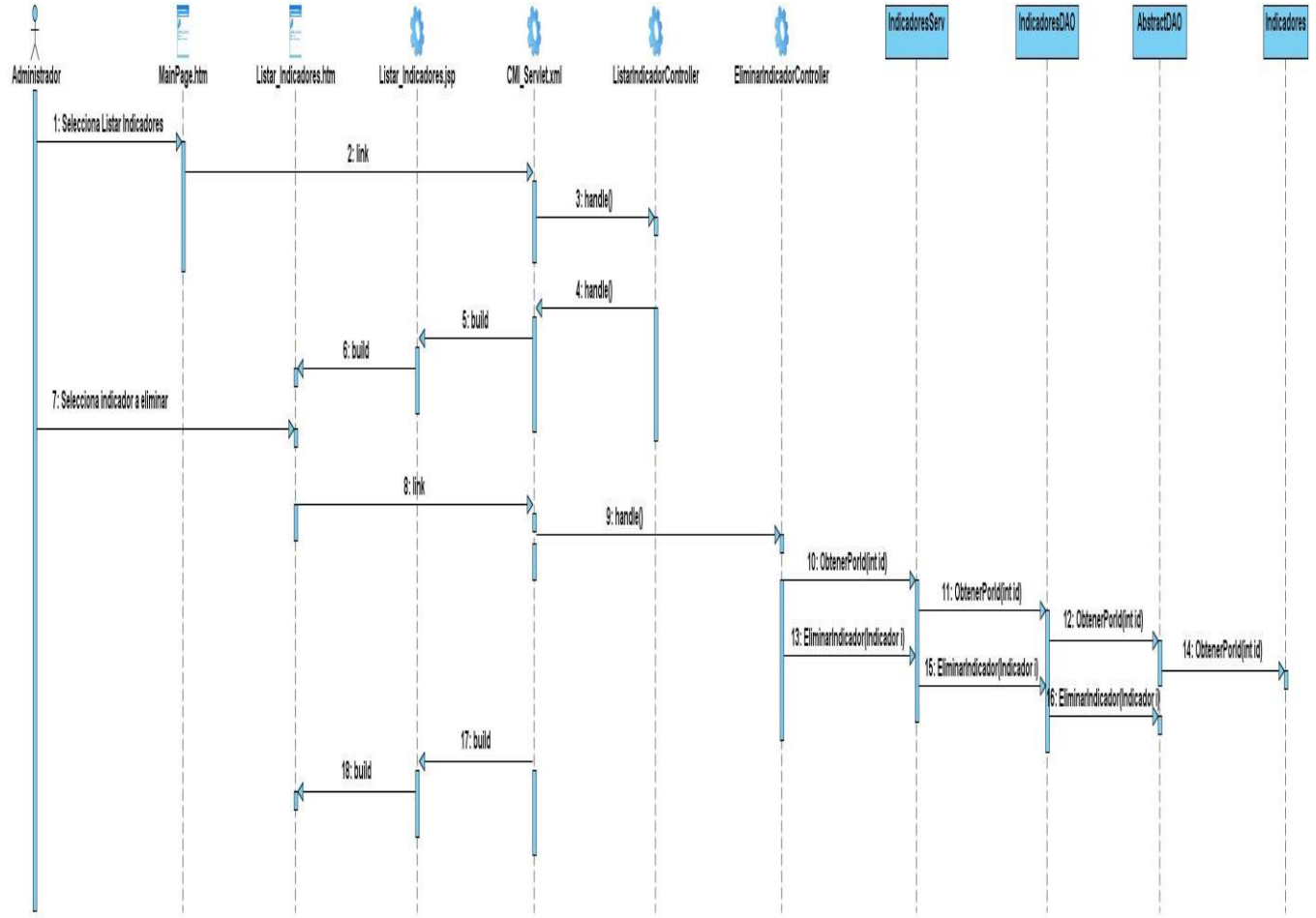


Figura 37 Diagrama de Secuencia de Gestionar Indicador, escenario “Eliminar Indicador”

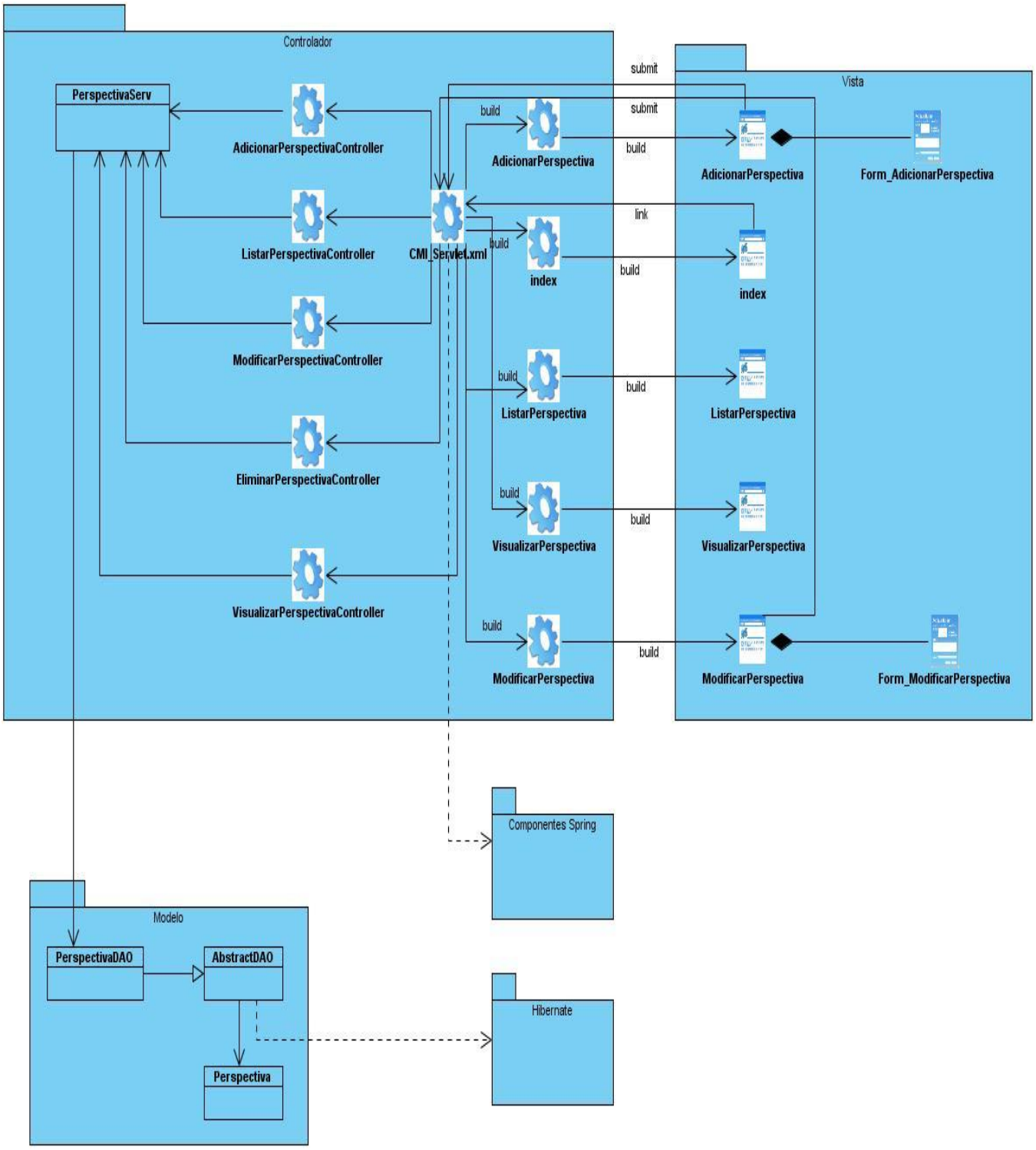


Figura 38 Diagrama de Clases del Diseño Gestionar Perspectiva

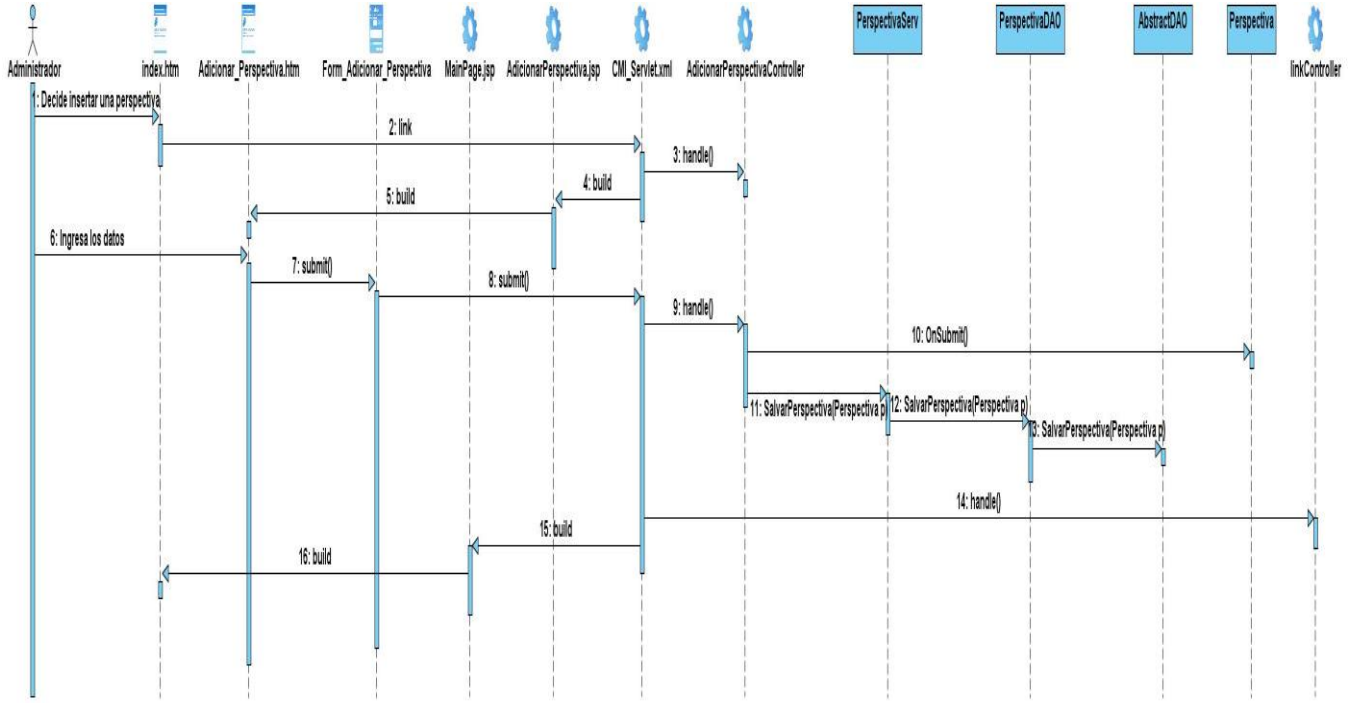


Figura 39 Diagrama de Secuencia Gestionar Perspectiva. Escenario Insertar_Perspectiva

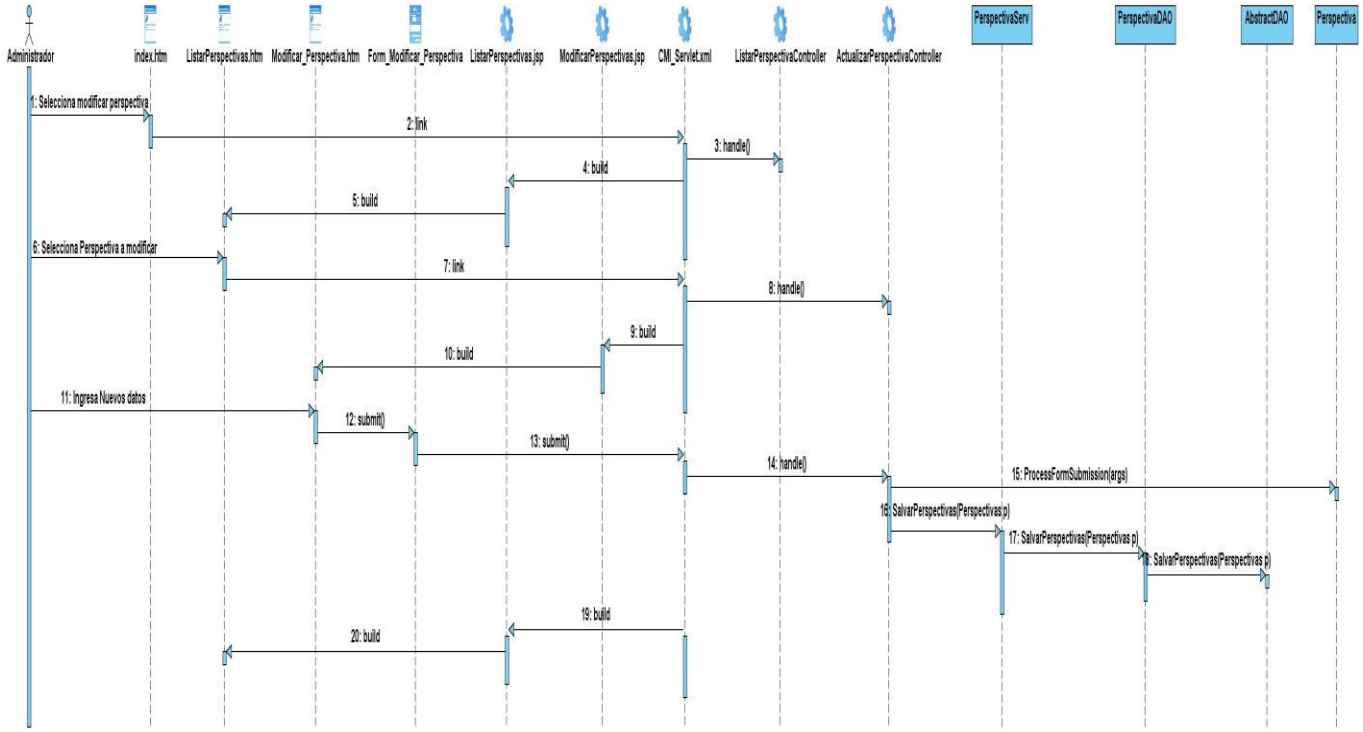


Figura 40 Diagrama de Secuencia Gestionar Perspectiva. Escenario Modificar_Perspectiva

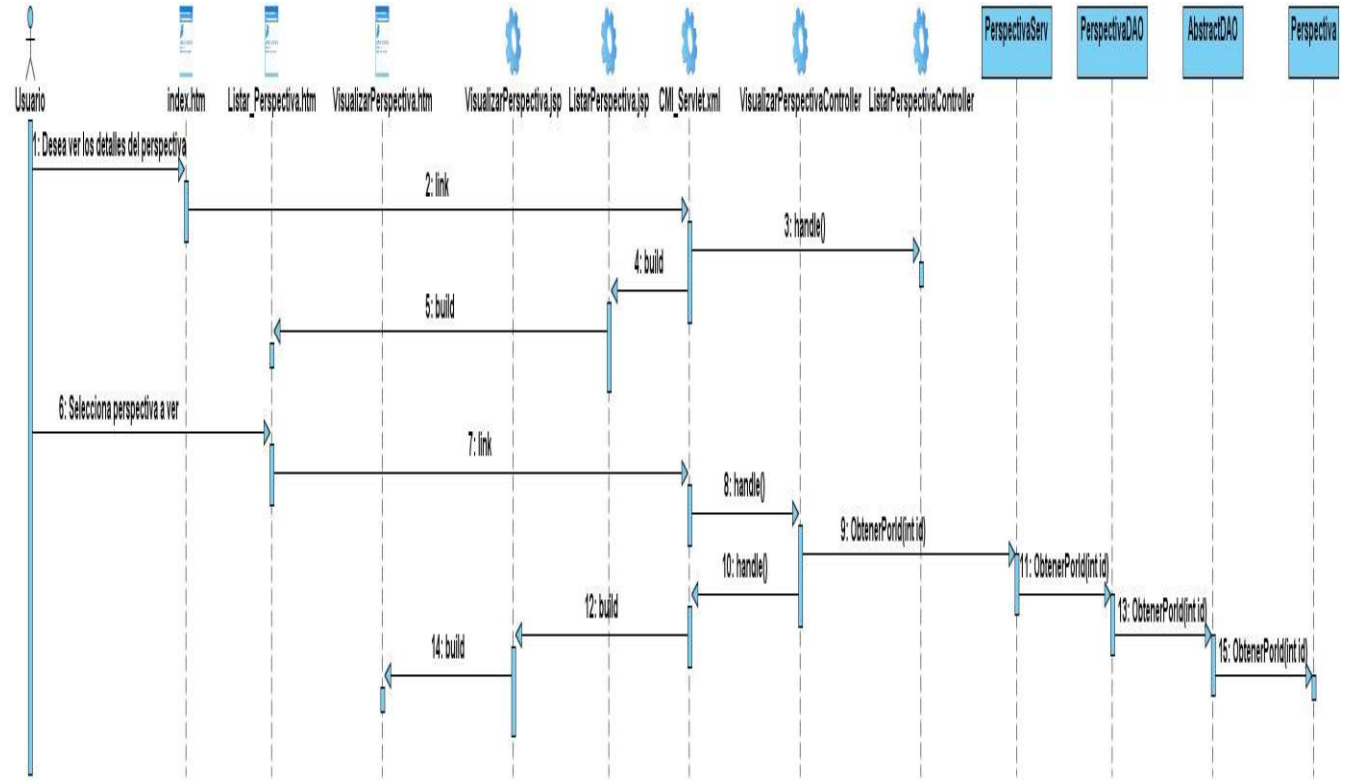


Figura 41 Diagrama de Secuencia Gestionar Perspectiva. Escenario Visualizar_Perspectiva

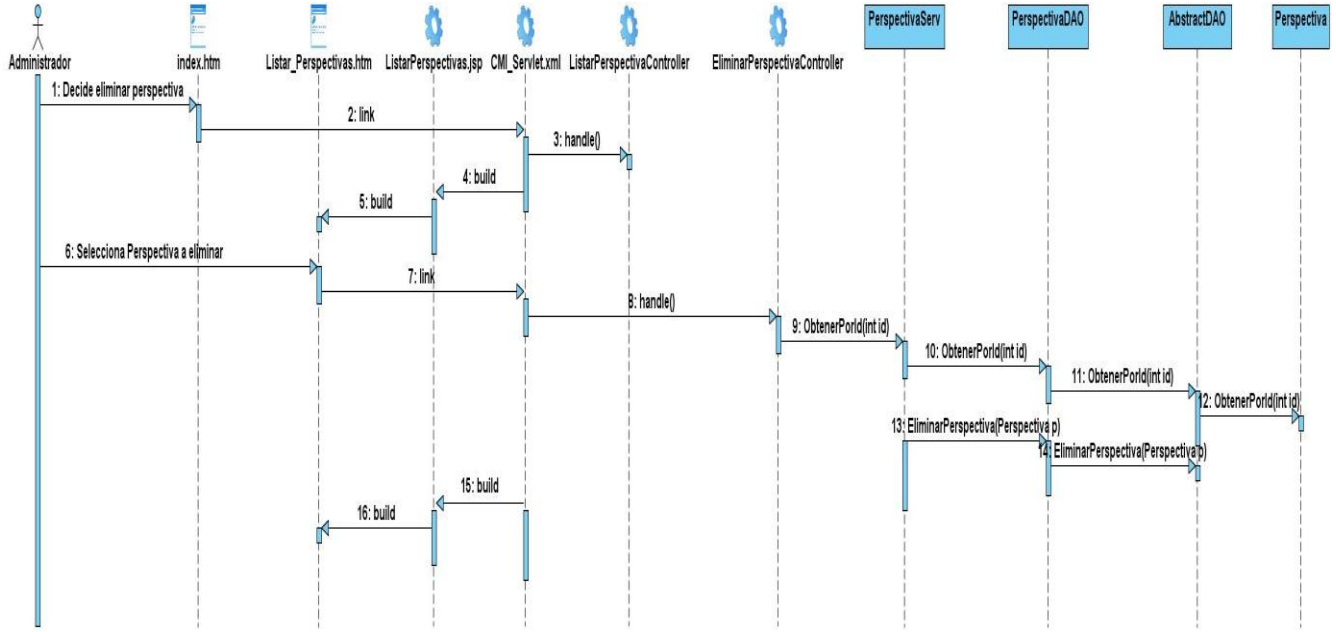


Figura 42 Diagrama de Secuencia Gestionar Perspectiva. Escenario Eliminar_Perspectiva

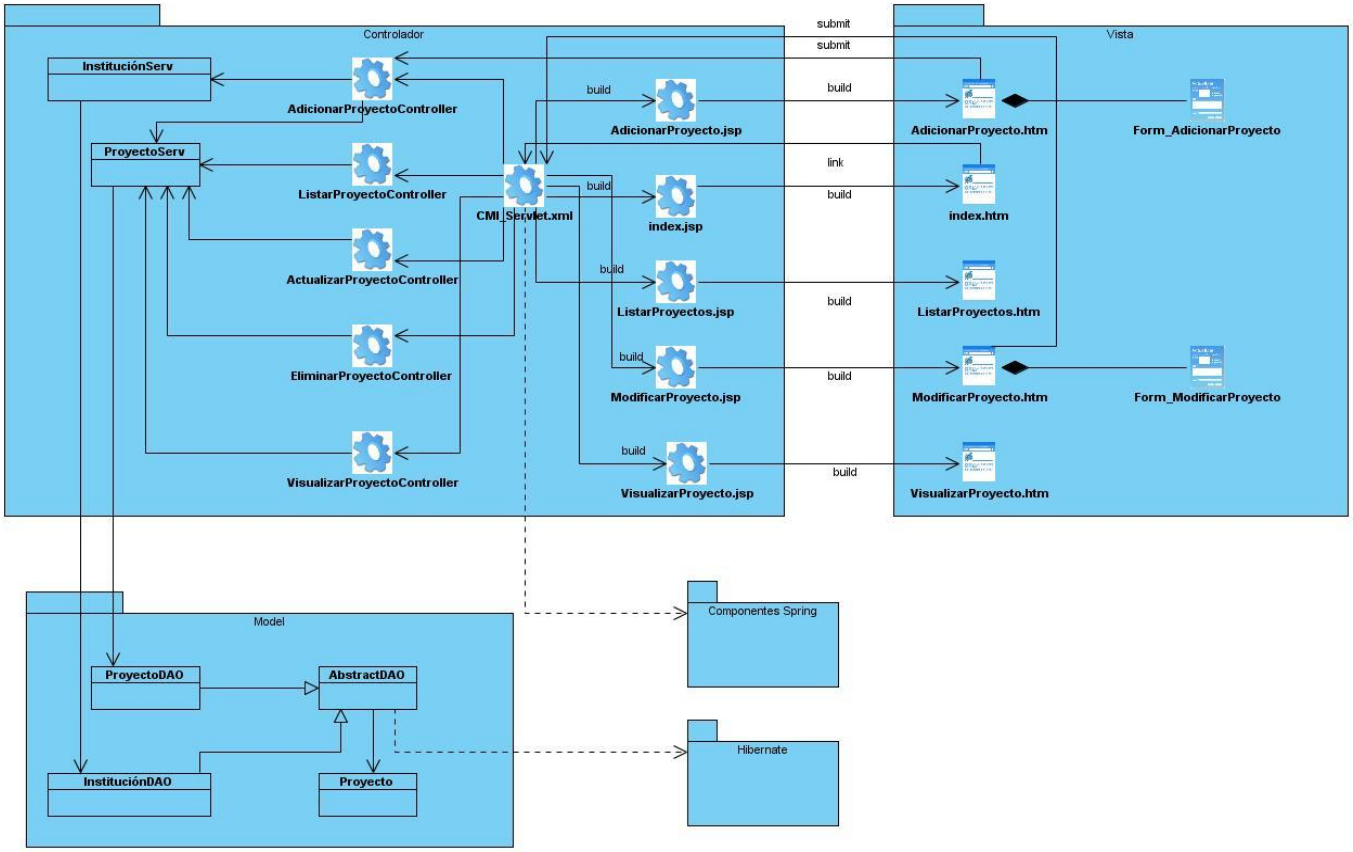


Figura 43 Diagrama de Clases del Diseño Gestionar Proyecto

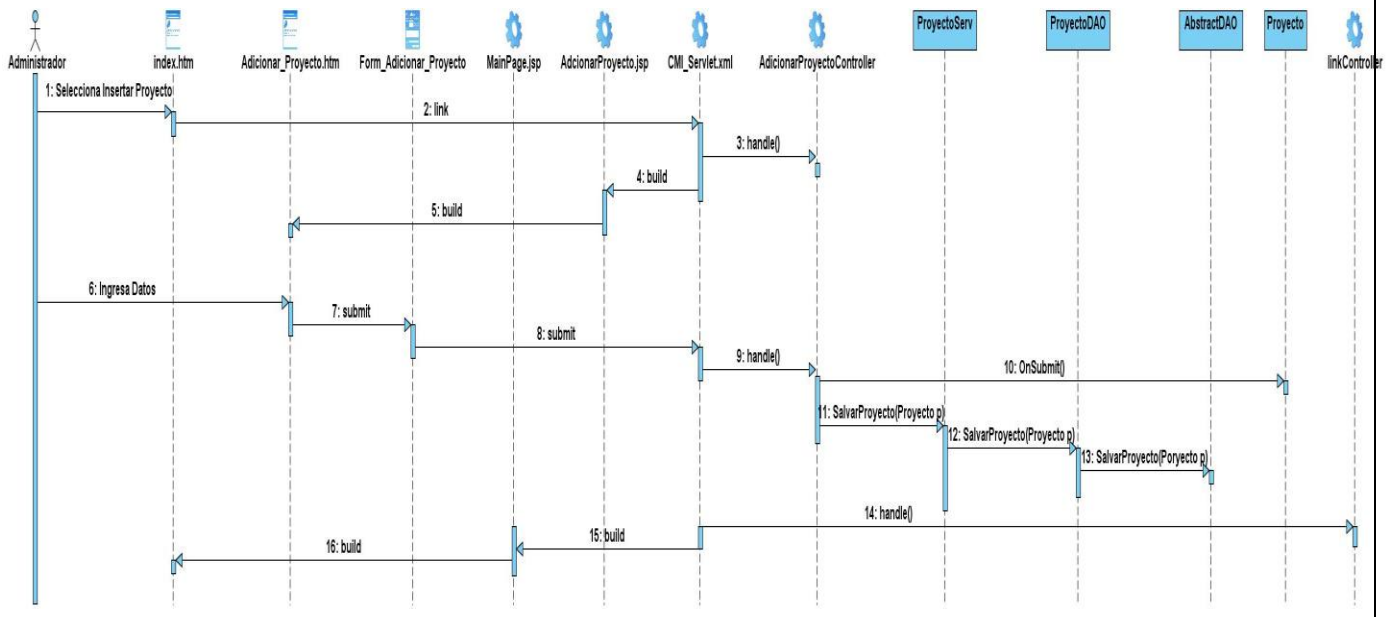


Figura 44 Diagrama de Secuencia Gestionar Proyecto. Escenario Insertar_Proyecto

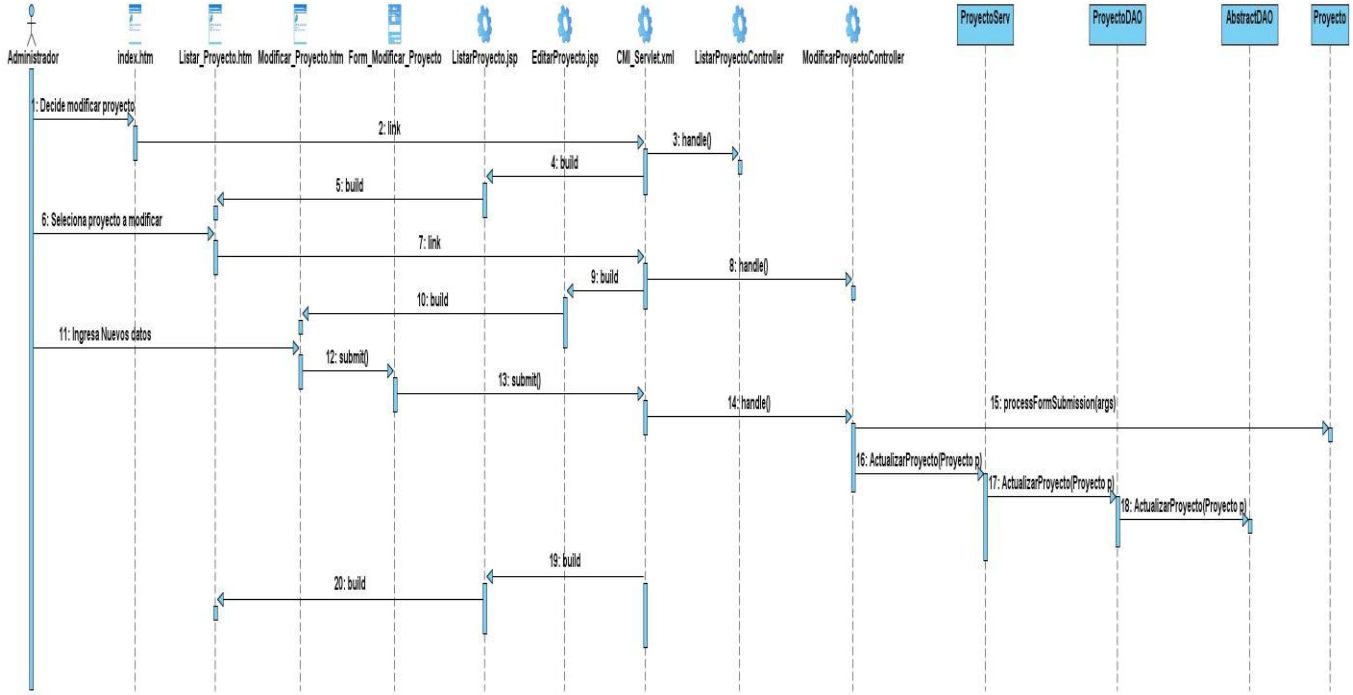


Figura 45 Diagrama de Secuencia Gestionar Proyecto. Escenario Modificar_Proyecto

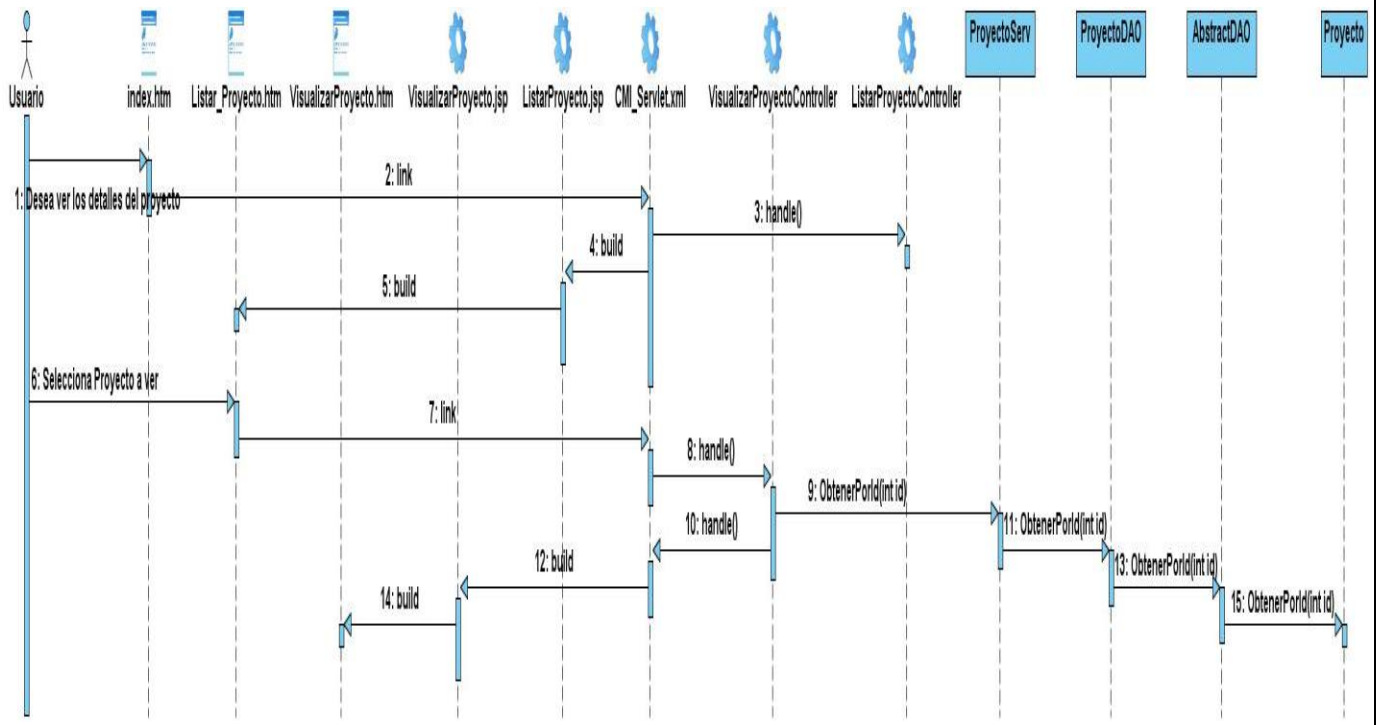


Figura 46 Diagrama de Secuencia Gestionar Proyecto. Escenario Visualizar_Proyecto

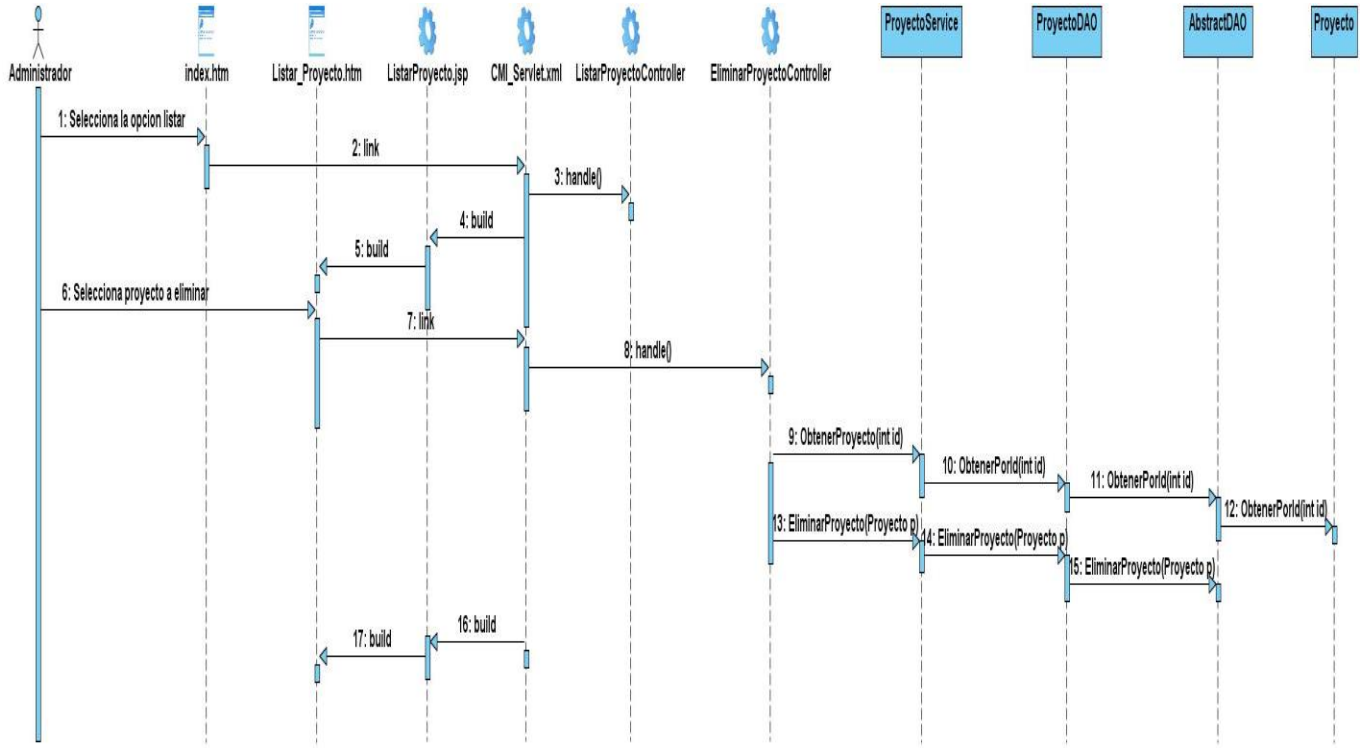


Figura 47 Diagrama de Secuencia Gestionar Proyecto. Escenario Eliminar_Proyecto

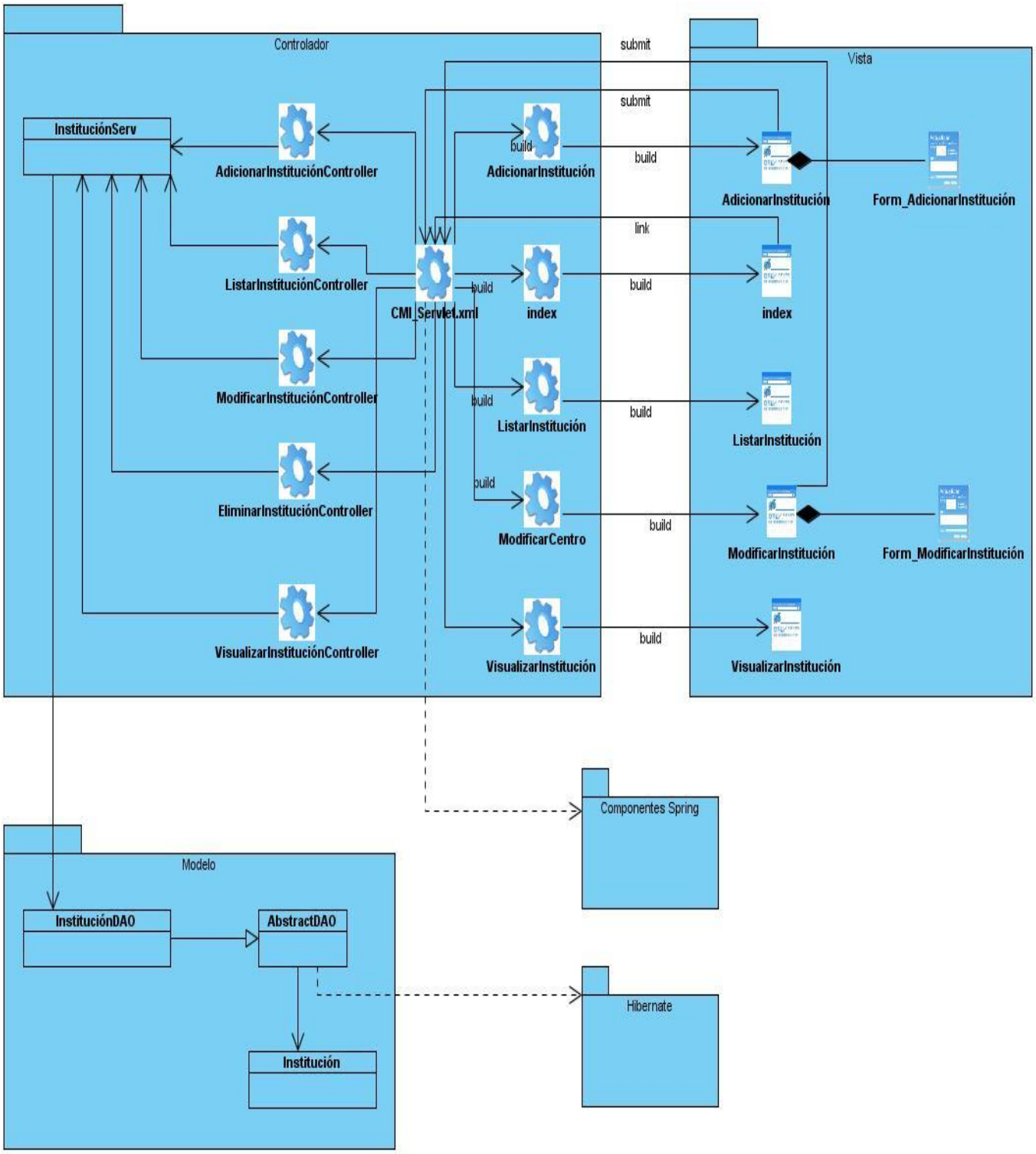


Figura 48 Diagrama de Clases del Diseño Gestionar Institución

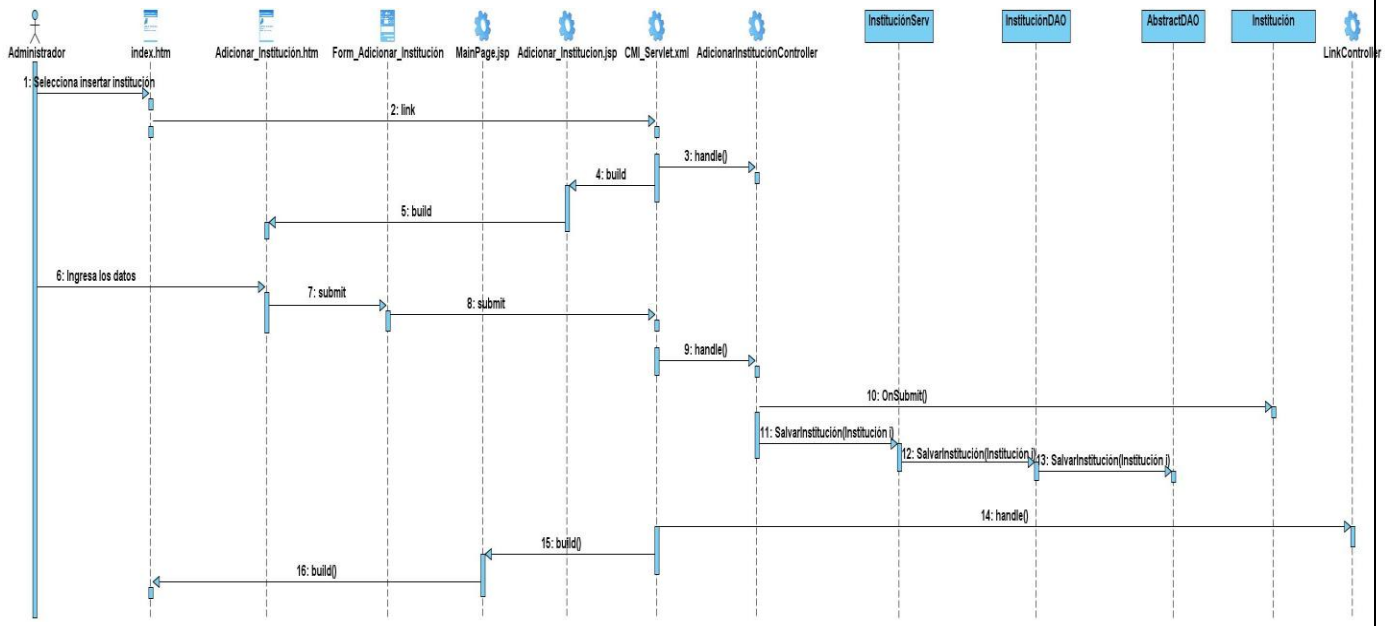


Figura 49 Diagrama de Secuencia Gestionar Institución. Escenario Insertar_ Institución

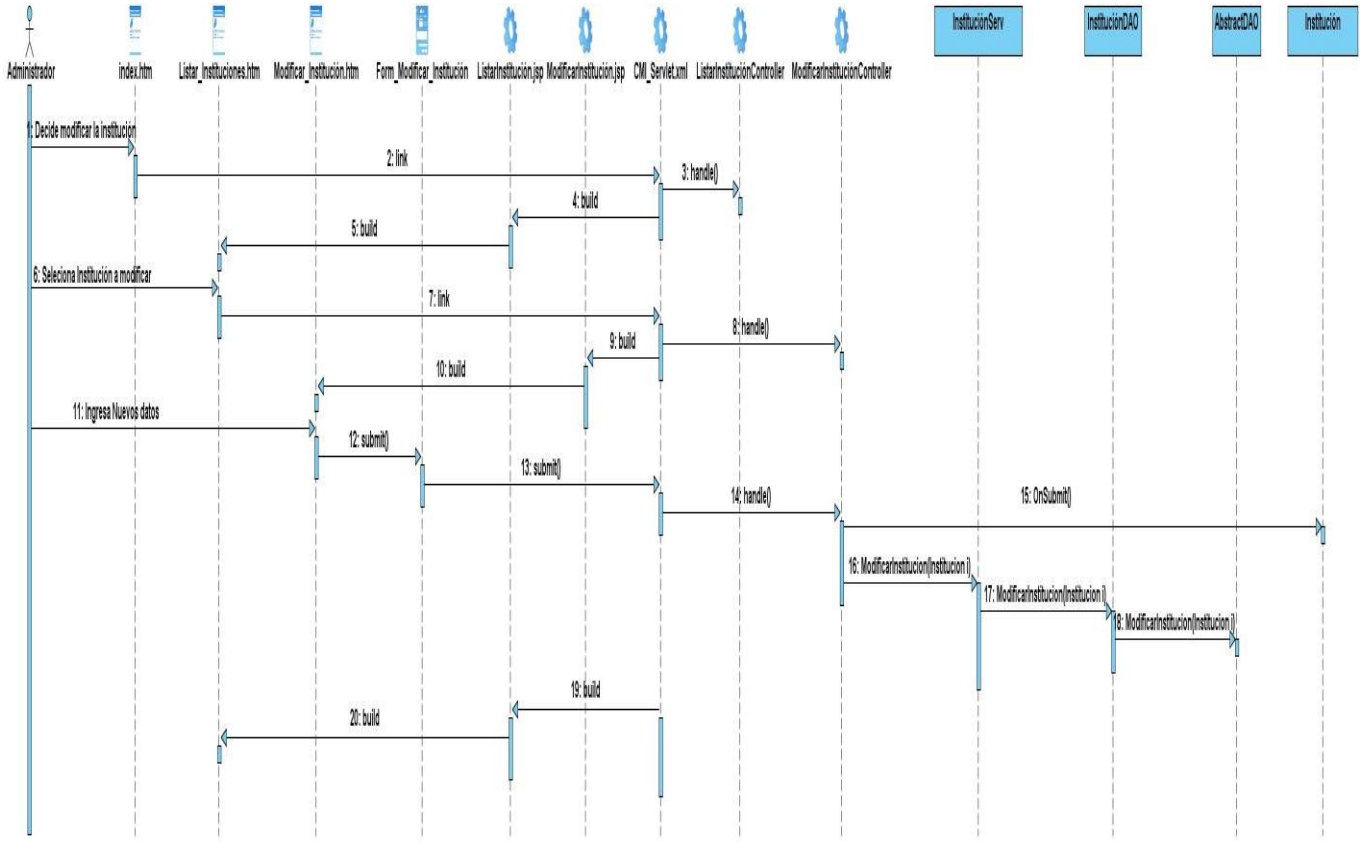


Figura 50 Diagrama de Secuencia Gestionar Institución. Escenario Modificar_ Institución

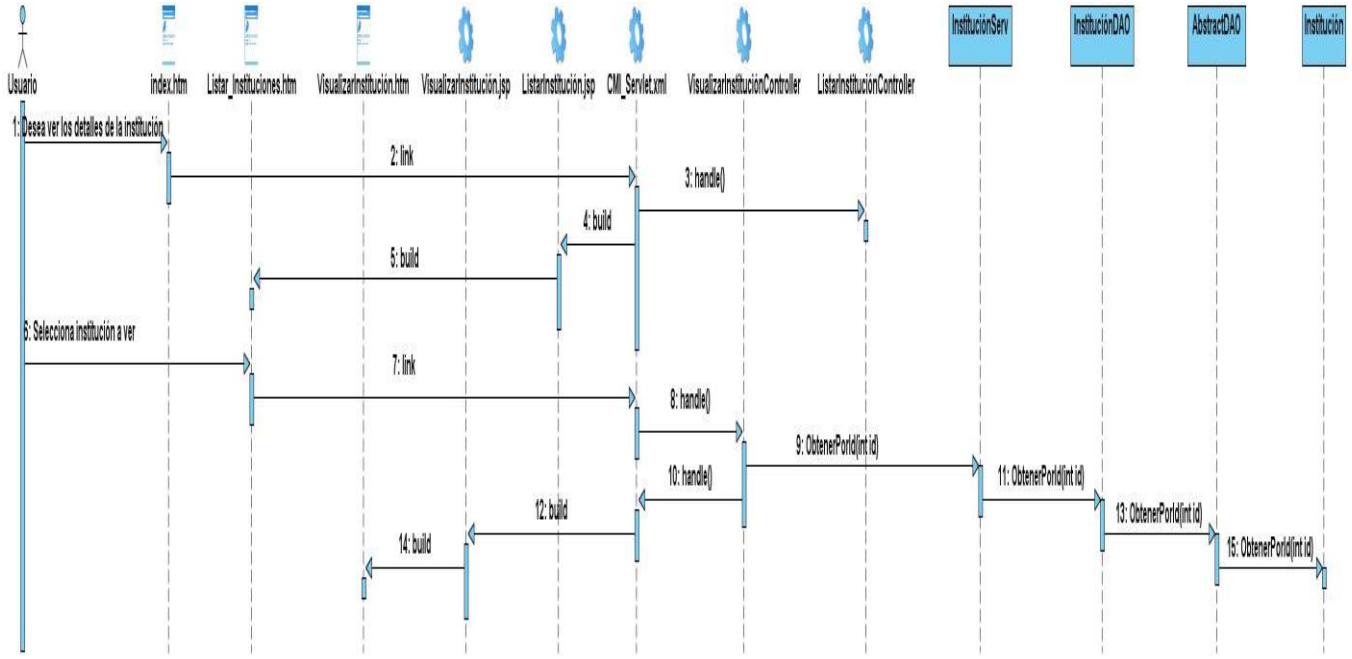


Figura 51 Diagrama de Secuencia Gestionar Institución. Escenario Visualizar_ Institución

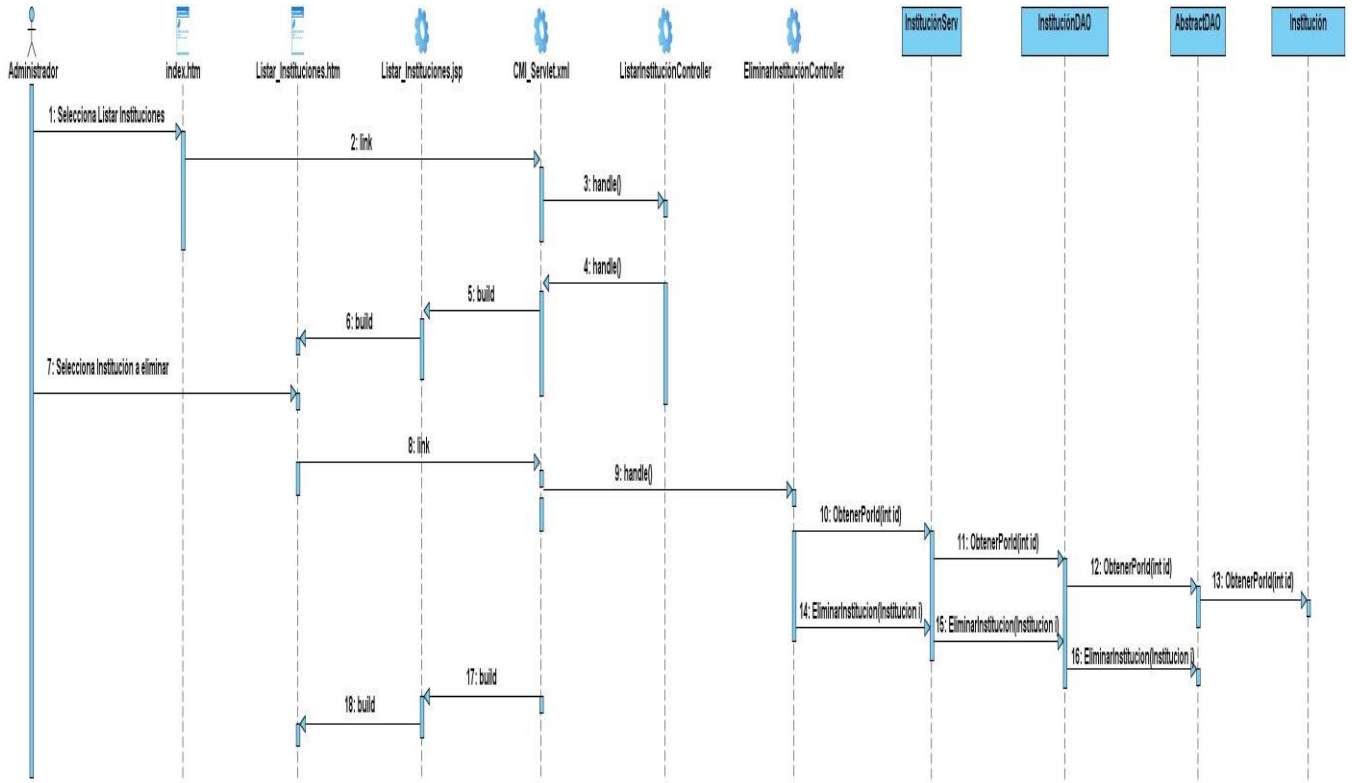


Figura 52 Diagrama de Secuencia Gestionar Institución. Escenario Eliminar_ Institución

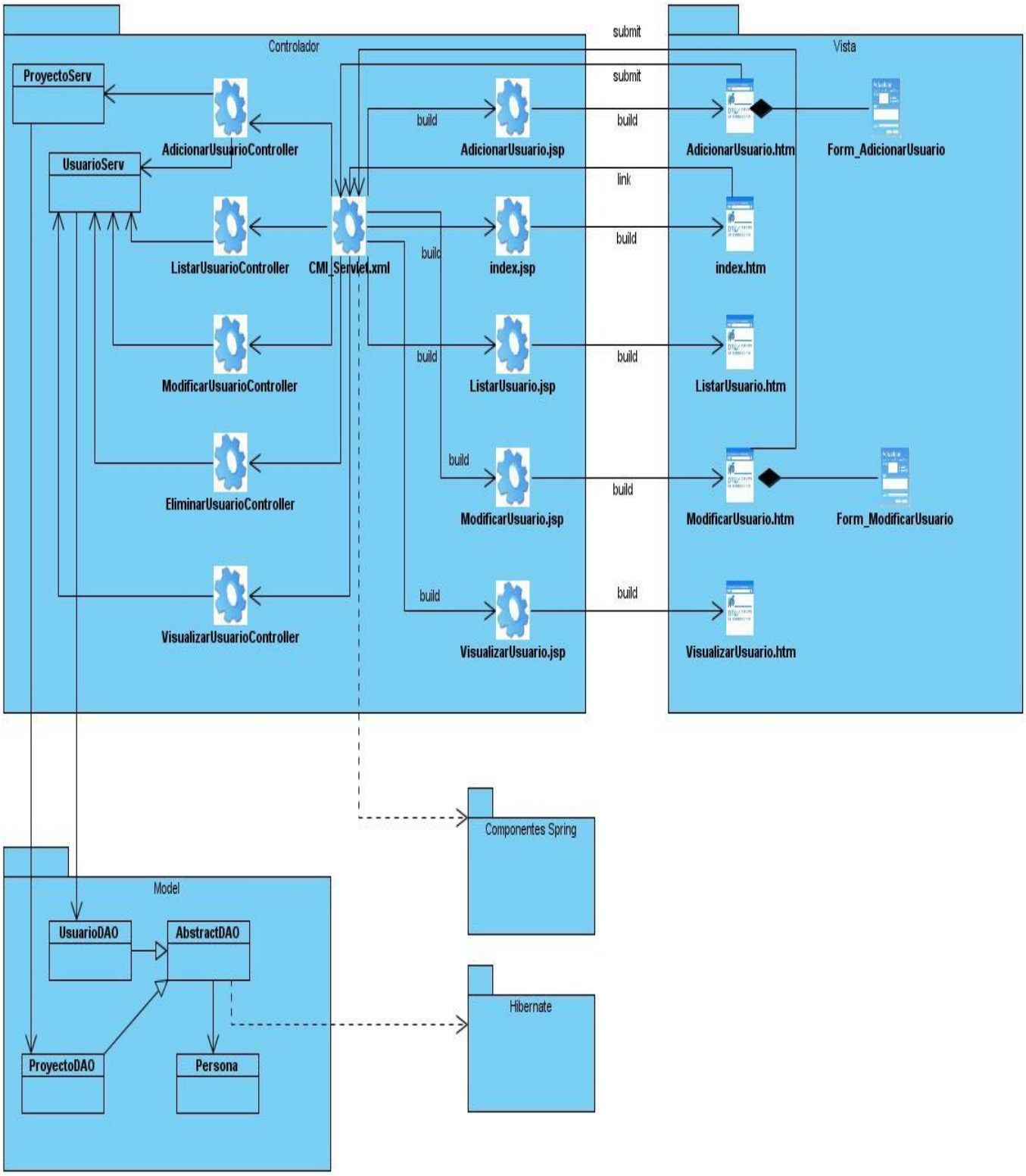


Figura 53 Diagrama de Clases del Diseño Gestionar Usuario

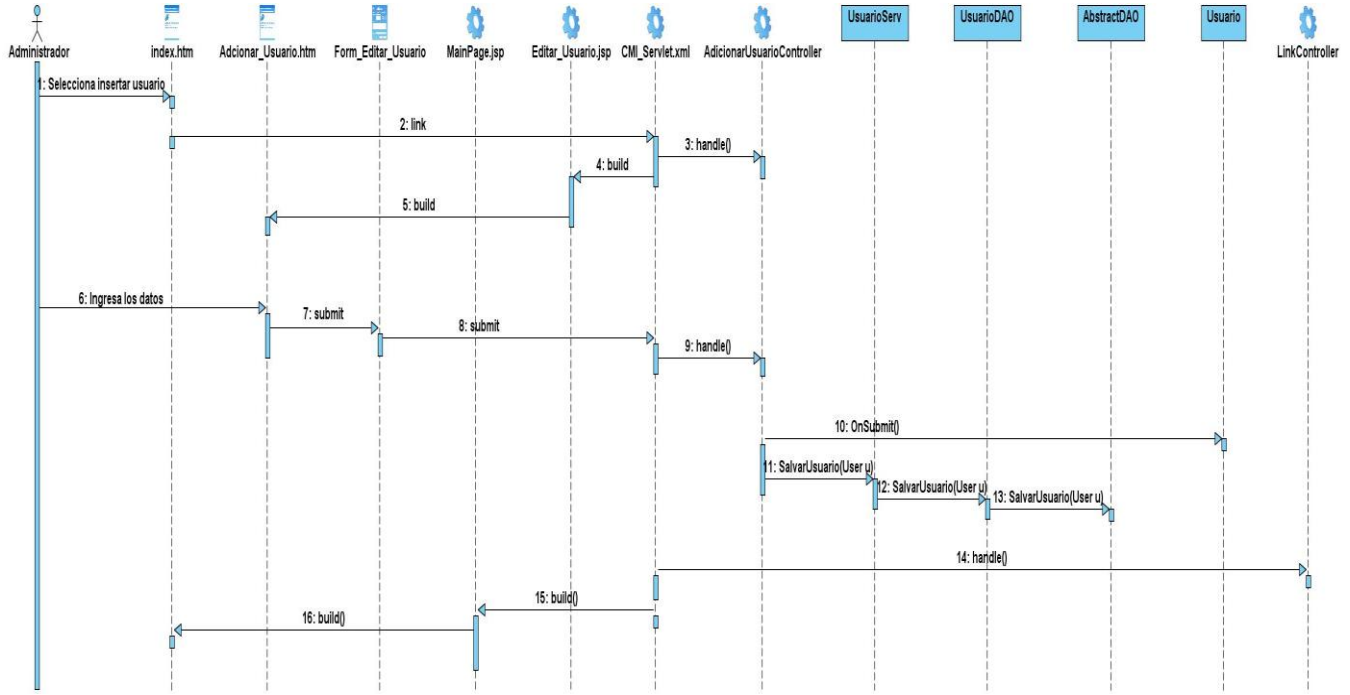


Figura 54 Diagrama de Secuencia Gestionar Usuario. Escenario Insertar_Usuario

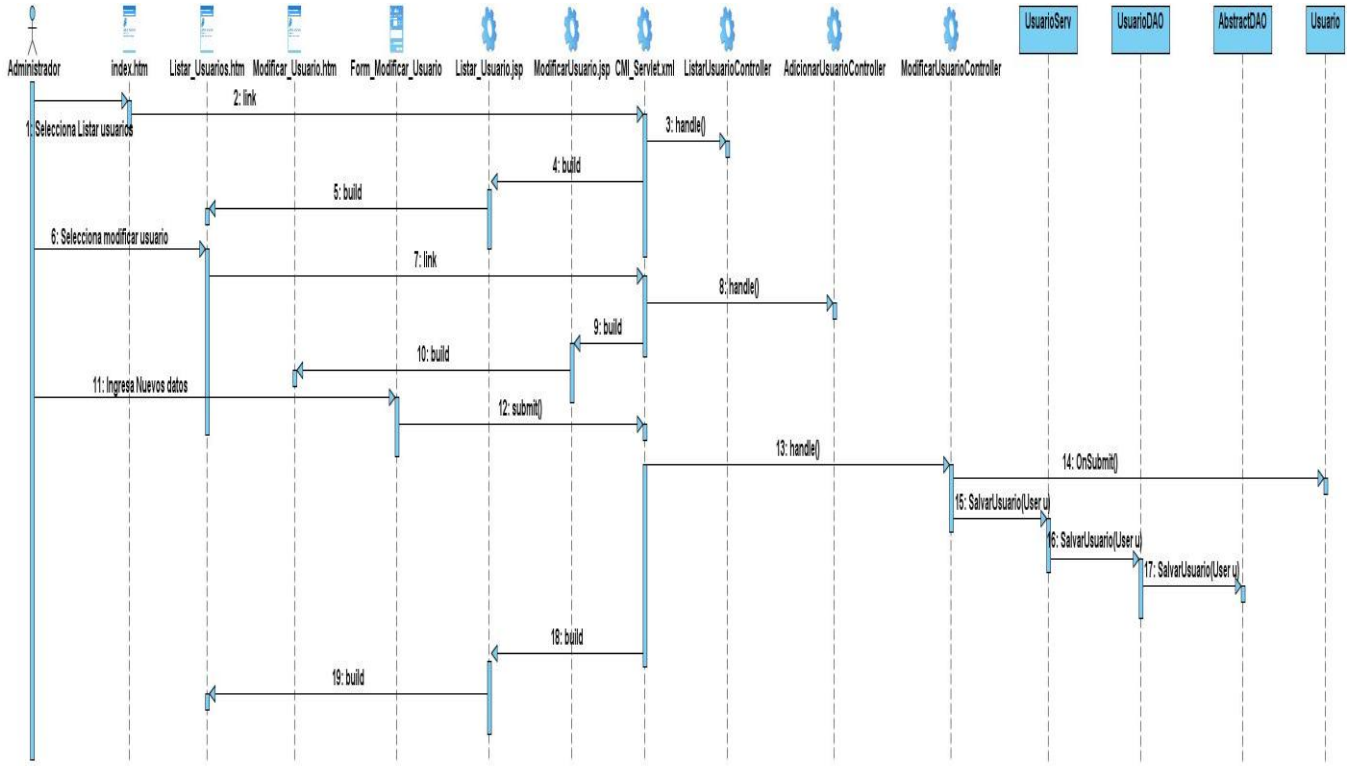


Figura 55 Diagrama de Secuencia Gestionar Usuario. Escenario Modificar_ Usuario

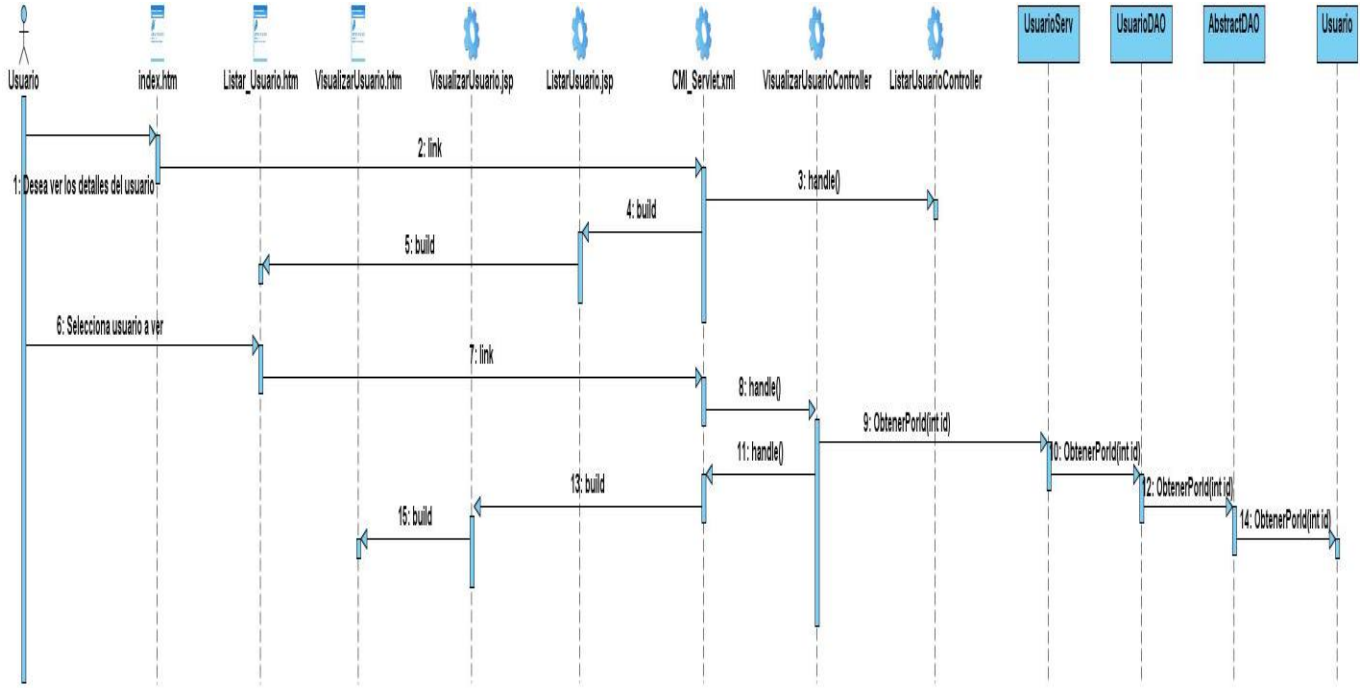


Figura 56 Diagrama de Secuencia Gestionar Usuario. Escenario Visualizar_ Usuario

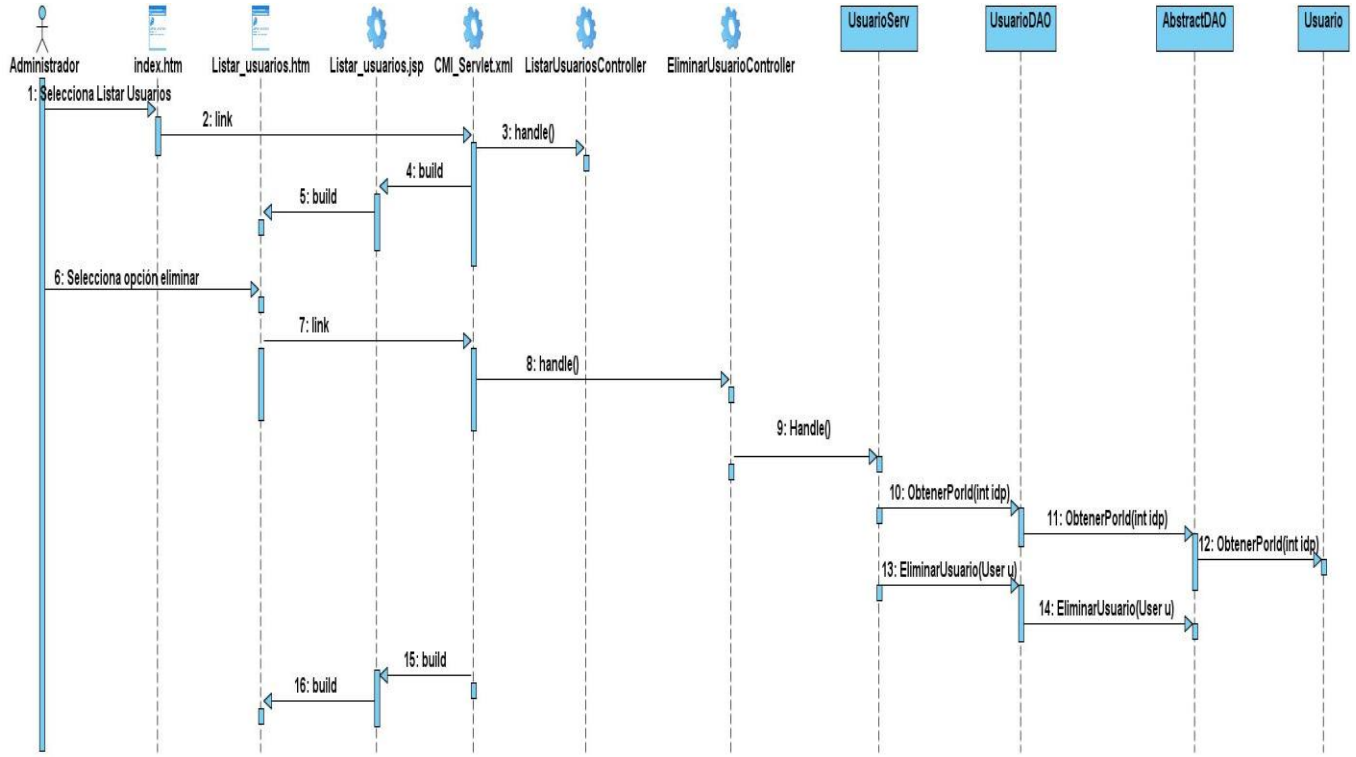


Figura 57 Diagrama de Secuencia Gestionar Usuario. Escenario Eliminar_ Usuario

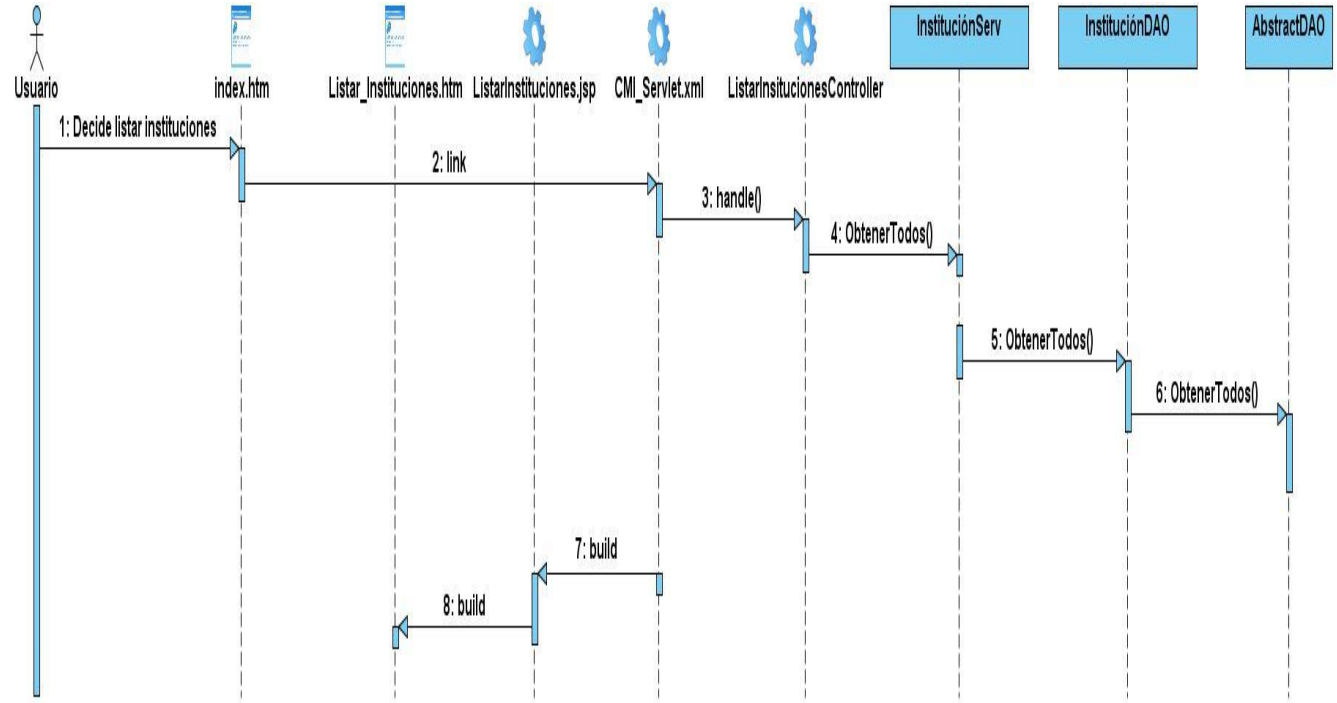


Figura 58 Diagrama de Secuencia de Listar_Instituciones

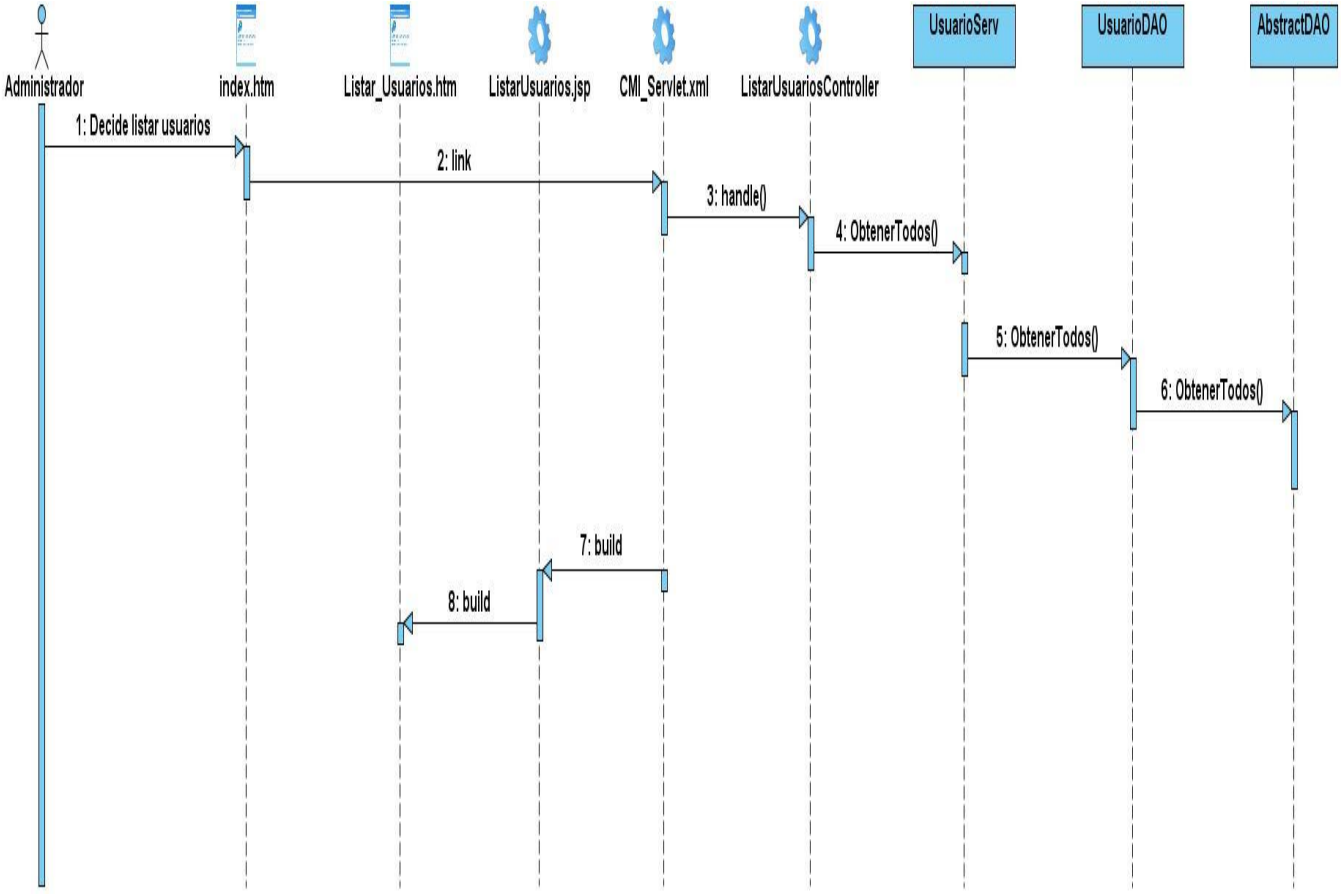


Figura 59 Diagrama de Secuencia de Listar_ Usuario.

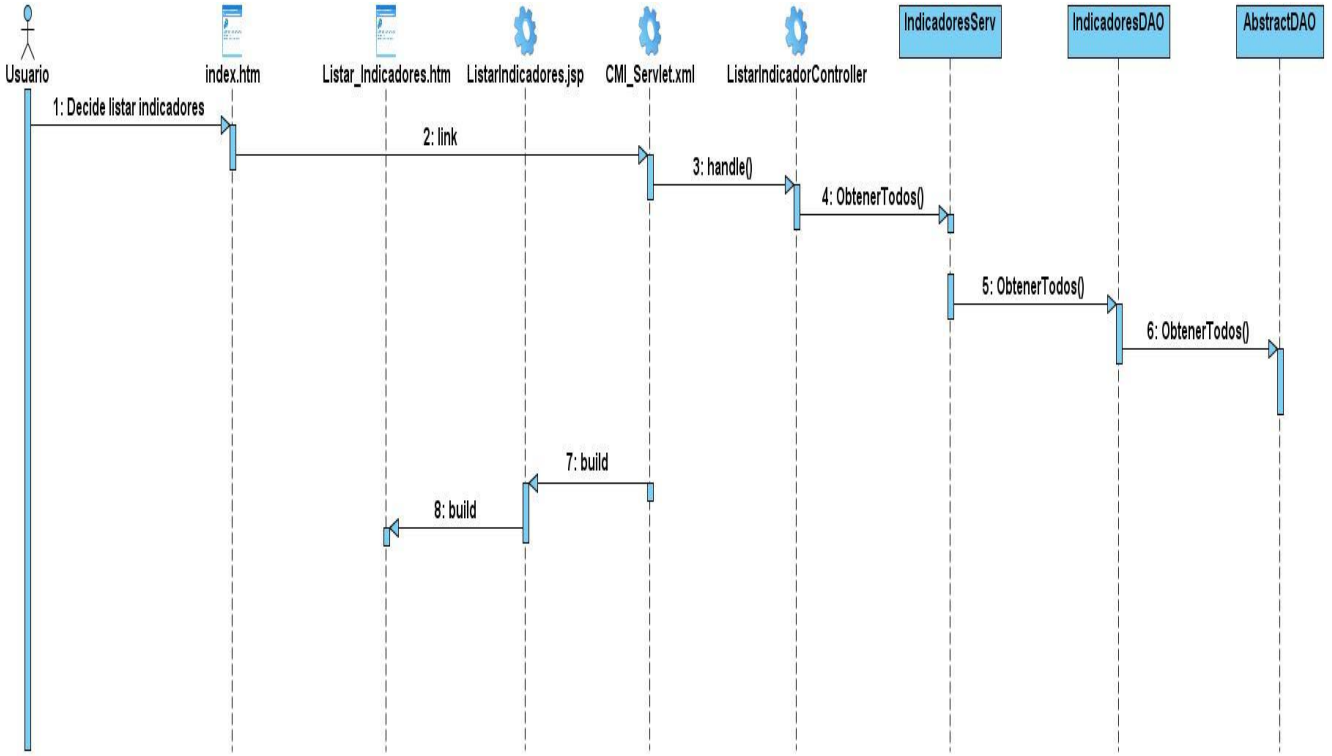


Figura 60 Diagrama de Secuencia de Listar_Indicadores.

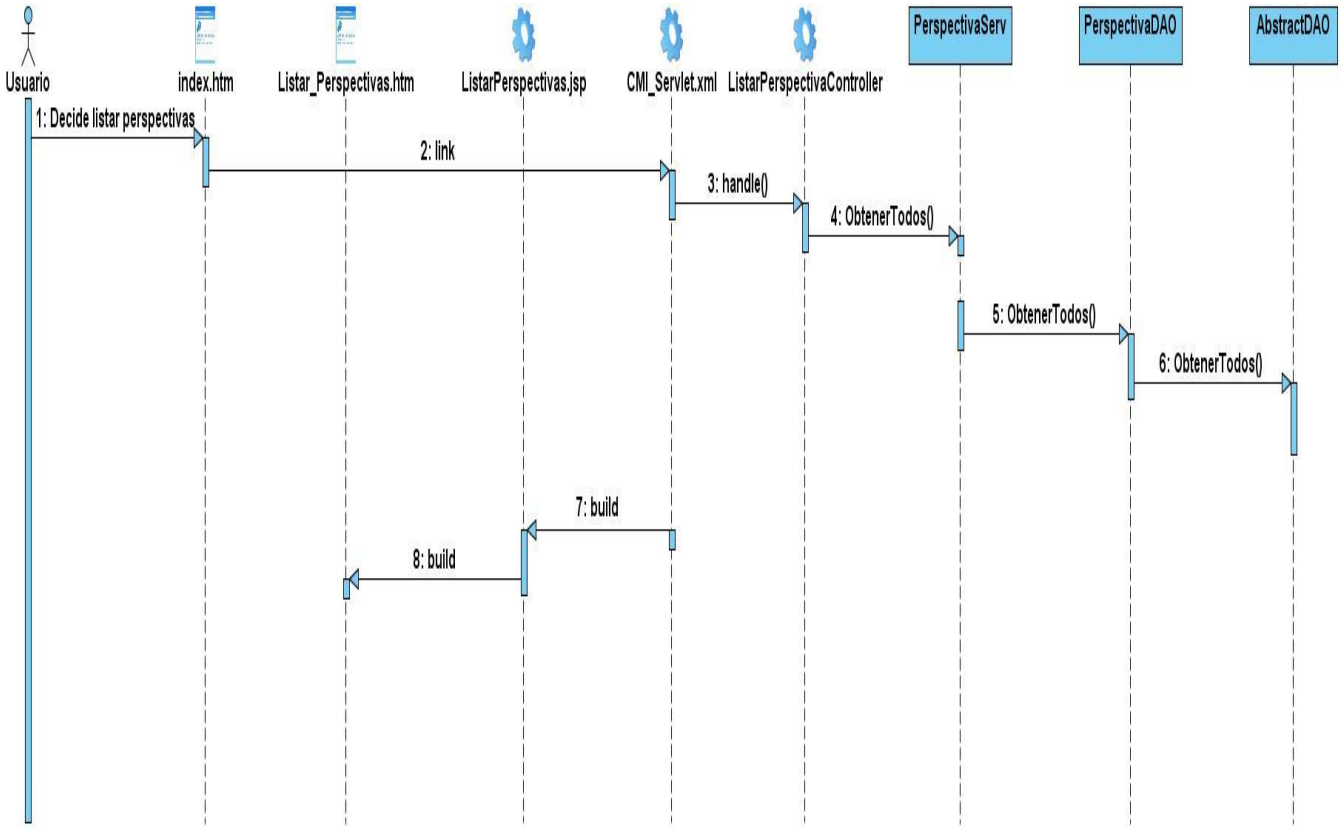


Figura 61 Diagrama de Secuencia de Listar_Perspectivas.

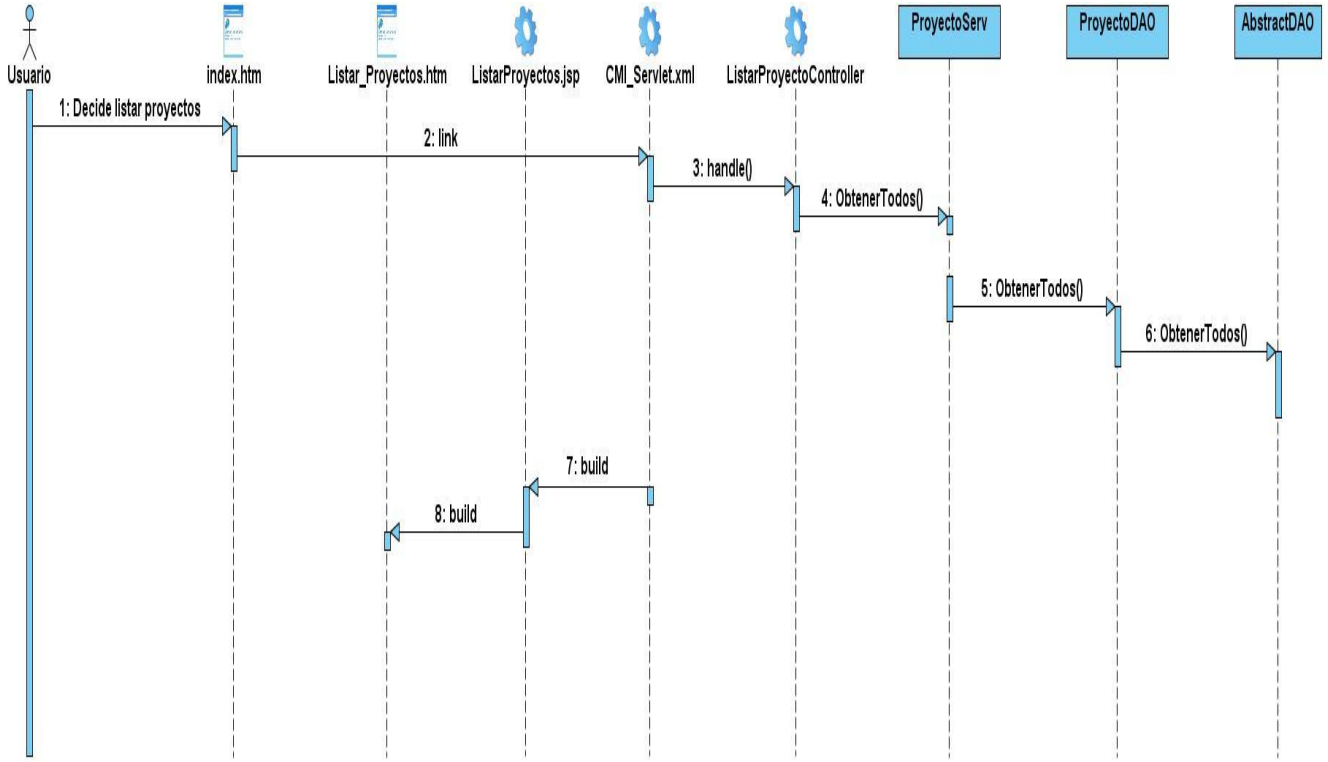


Figura 62 Diagrama de Secuencia de Listar_Proyectos.

ANEXOS CAPITULO IV

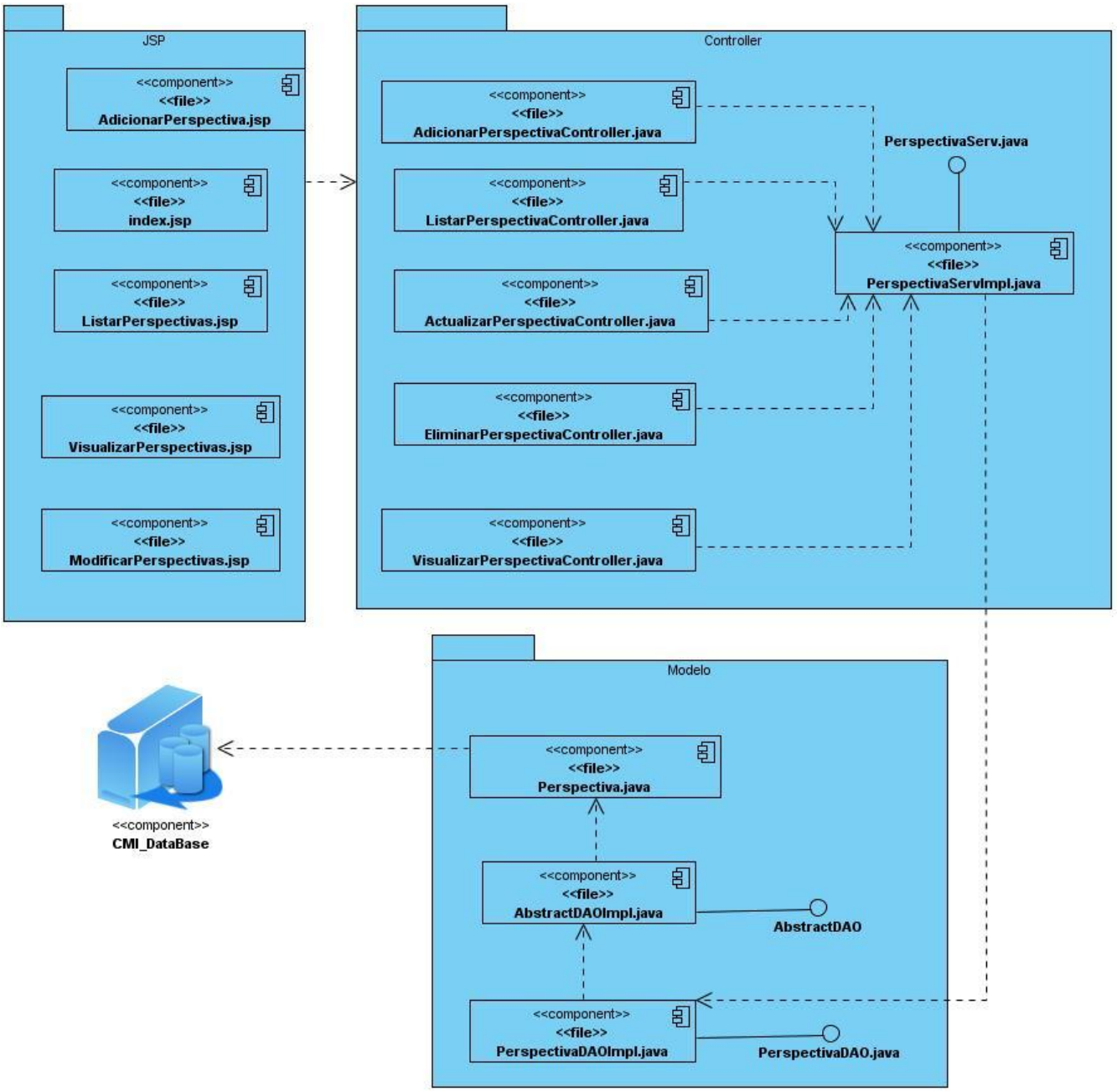


Figura 63 Diagrama de Componentes Gestionar_Perspectiva

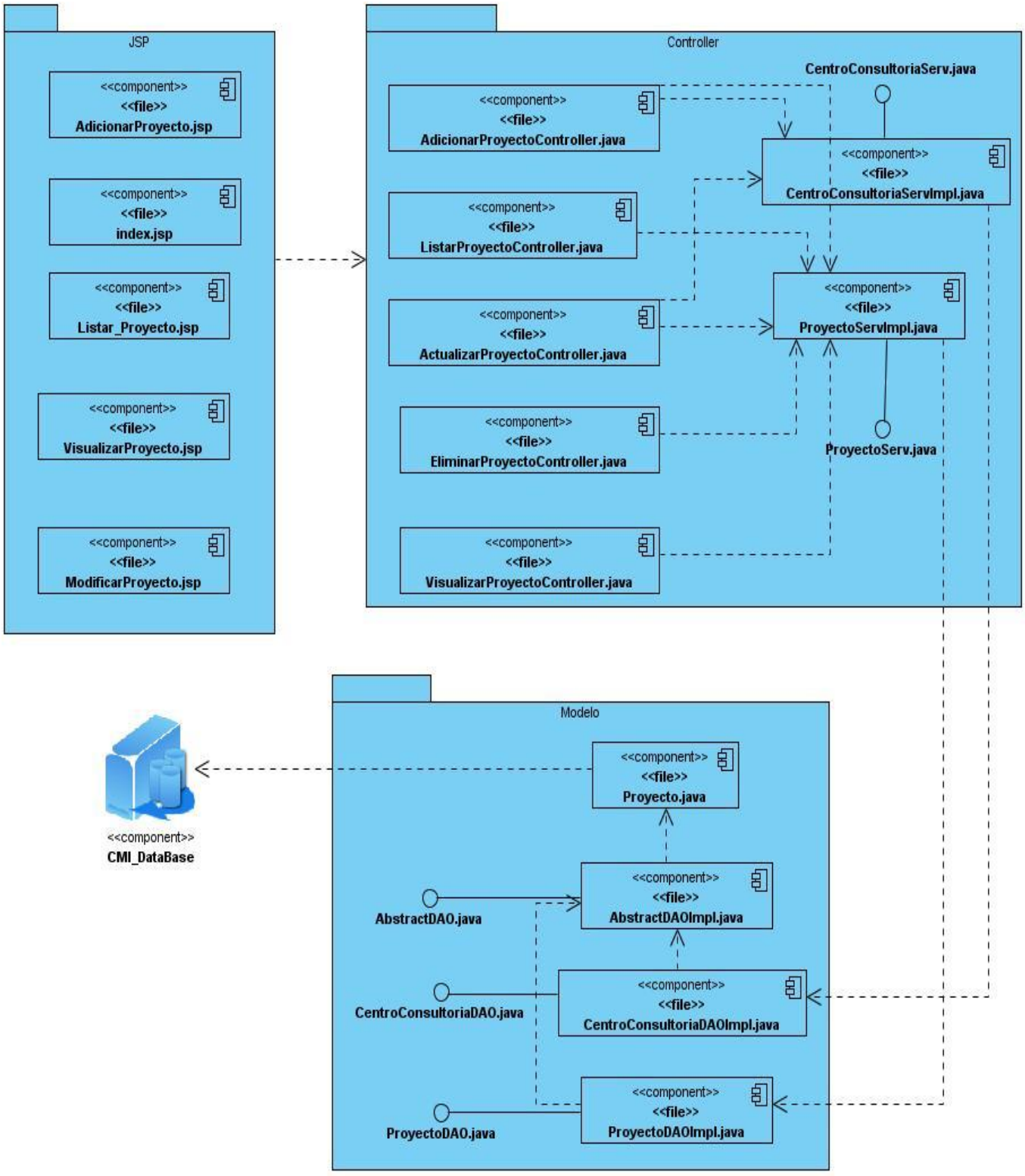


Figura 64 Diagrama de Componentes Gestionar_Proyecto

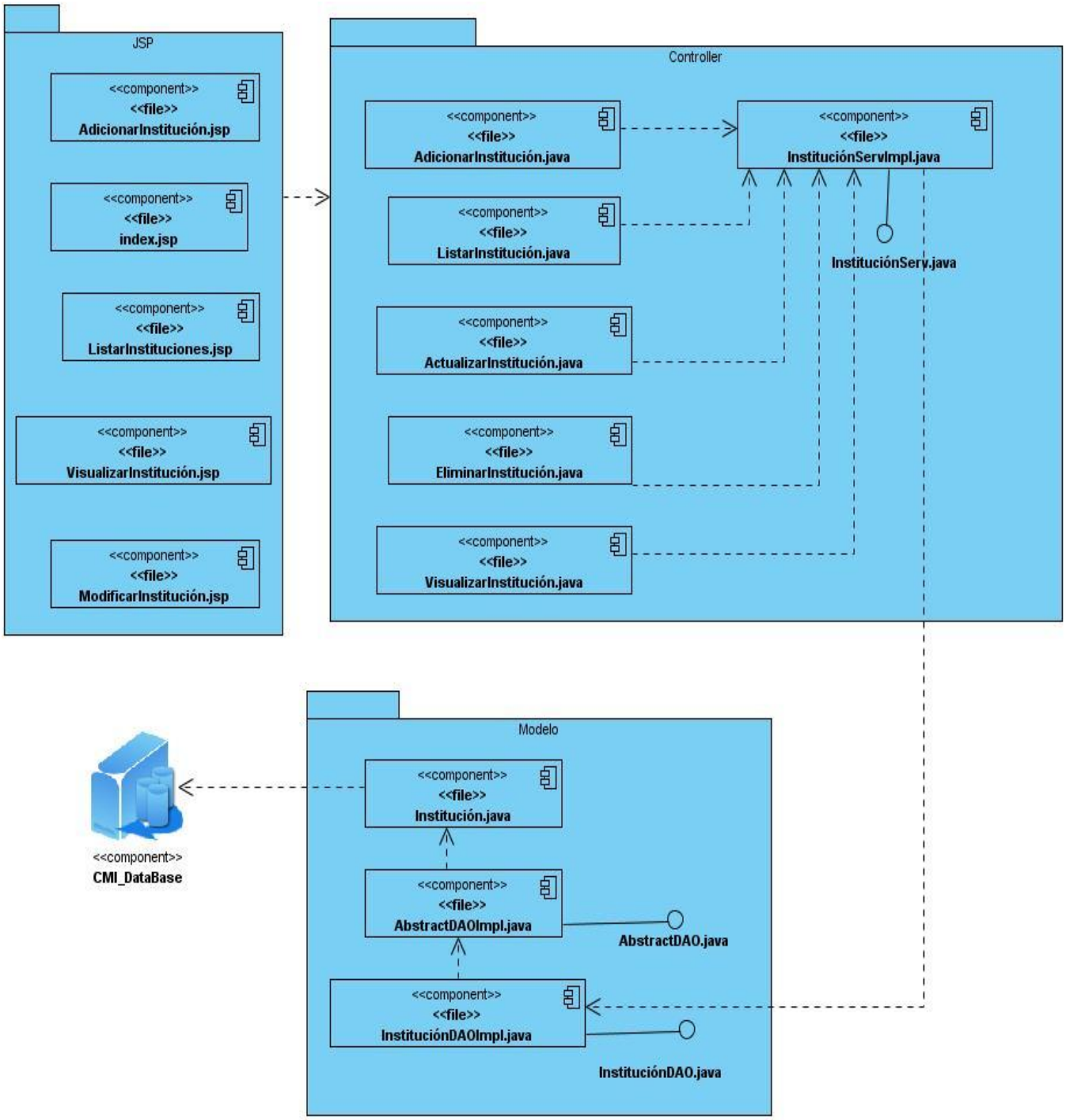


Figura 65 Diagrama de Componentes Gestionar_Institución

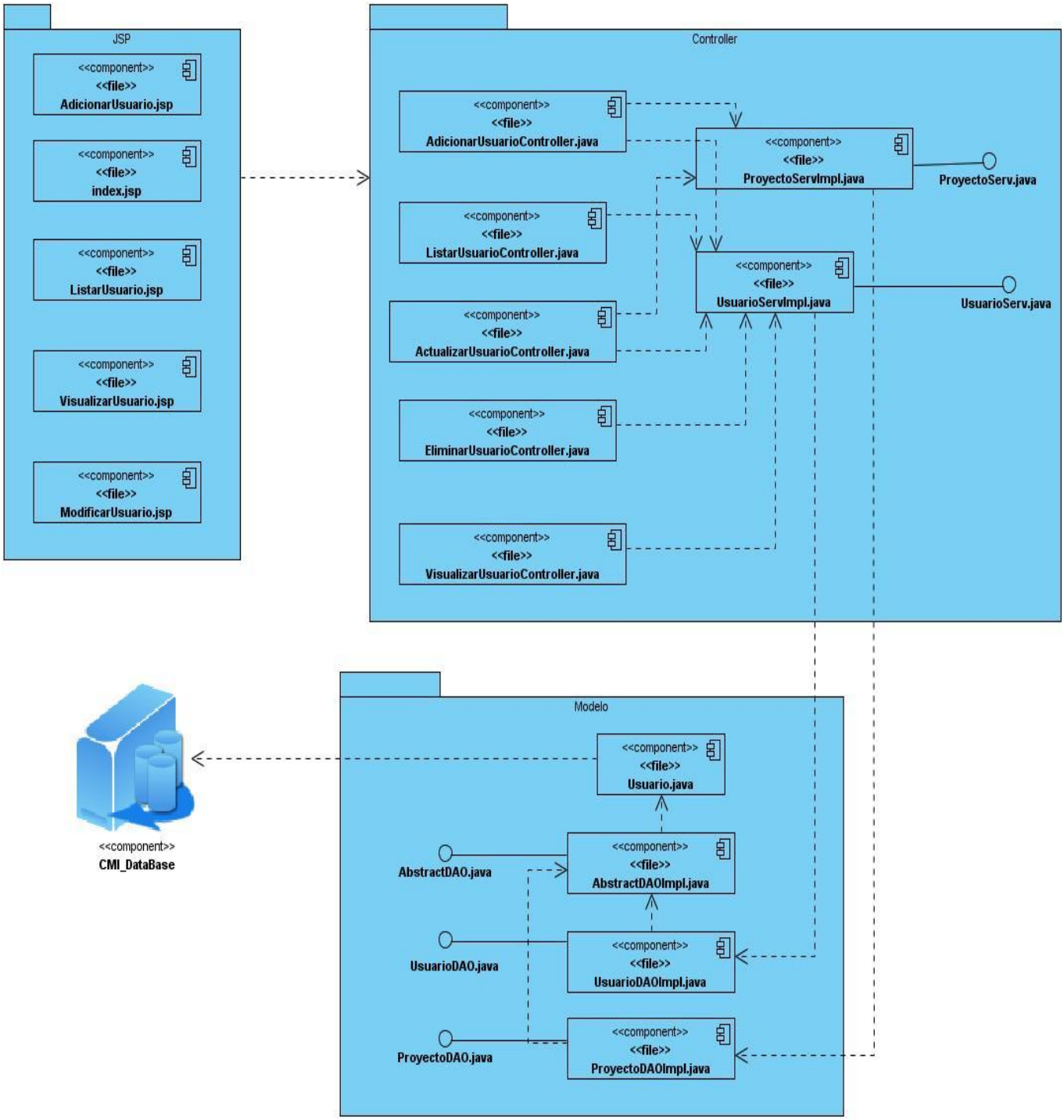


Figura 66 Diagrama de Componentes Gestionar_ Usuario

Glosario de Términos

Indicadores: son una representación sintética de la realidad, no son la realidad, a la cual intentan representar.

Metodologías: imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente.

Perspectivas: representan un grupo equilibrado de objetivos estratégicos y son utilizadas para organizar los modelos de negocio así como para estructurar los indicadores y la información.

Indicador clave: En la terminología empresarial, un indicador clave es una medida cuantificable para valorar los éxitos empresariales.

Indicadores financieros: son índices estadísticos que muestran la evolución de las principales magnitudes de las empresas financieras, comerciales e industriales a través del tiempo.

Estrategia: es el programa, general de una organización para definir y alcanzar sus objetivos, así pues, la estrategia es un proceso interactivo entre la empresa y su entorno que implica la formulación de la misión y los objetivos para el horizonte temporal, que abarca el sistema de decisión, persigue mejorar y defender la competitividad de la empresa, y requiere los establecimientos de políticas y objetivos operativos.

Estrategia Institucional: puede definirse como la determinación del propósito (o misión) y de los objetivos básicos a largo plazo de una empresa, así como la adopción de los cursos de acción y de la asignación de recursos necesarios para cumplirlas.

Artefacto: Es un término general para cualquier tipo de información creada, producida, cambiada o utilizada por los trabajadores en el desarrollo del sistema. Algunos ejemplos de artefactos son los diagramas UML y su texto asociado (Jacobson, et al., 2000).

Rol: papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso.

Herramientas CASE: son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Integridad Referencial: es un sistema de **reglas** que utilizan la mayoría de las bases de datos (BD) relacionales para *asegurarse que los registros de tablas relacionadas son válidos* y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad. Básicamente hace que el SGBD se asegure de que no haya en las claves foráneas valores que no estén en la tabla principal.

La **minería de datos** (**DM**, *Data Mining*): consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. En otras palabras, la minería de datos **prepara**, **sondea** y **explora** los datos para sacar la información oculta en ellos.

Base de Datos: Conjunto de datos organizados de tal manera, que pueda extraerse información y que pueda ser compartida. Entre los usuarios debe existir un consenso en lo que representan los datos (la información a obtener).

Base de datos relacional: Base de datos o sistema de administración de bases de datos que almacena información en tablas como filas y columnas de datos y realiza búsquedas mediante los datos de columnas especificadas de una tabla para buscar datos adicionales en otra tabla.

Datos: Es la representación de un mensaje (debe ser objetivo).

Información: Son los datos procesados (debe ser subjetivo).

Modelo: Representación gráfica de la realidad que son clarificados a través de texto explicativo. Ejemplo: Una representación a escala de lo que será en el futuro el diseño de una clase.

Modelo de Datos: Estructura de datos y reglas de negocio que representan los requerimientos de un sistema.

Entidad: Una persona, lugar, evento o concepto acerca del cual el negocio necesita guardar datos.

Atributo: Propiedad de una entidad que almacenará datos.

Llave Primaria (PK): Un atributo (llave simple) o conjunto de atributos (llave compuesta) que identifican únicamente una instancia (fila o registro) de una entidad.

Llave alterna: Un atributo (llave simple) o conjunto de atributos (llave compuesta) que identifican únicamente una instancia (fila o registro) de una entidad, pero que NO ES ESCOGIDA como lave primaria.

Llave Foránea (FK): Llave primaria de una entidad padre (fuerte) que es agregada a la entidad hijo (débil) a través de su relación.

Relación: Un enlace lógico entre dos entidades que representa una regla de negocio o una restricción.

Relación Identificada: La llave primaria de la entidad padre es migrada a través de la RELACION para FORMAR parte de la llave primaria de la entidad hijo.

Relación de MUCHOS a MUCHOS: La llave primaria de la entidad padre no es migrada como llave foránea.

Herramienta CASE (*Computer **A**ided Assisted Automated Software Systems Engineering*): conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todas las fases del Ciclo de Vida de desarrollo de un Software.

Caso de Uso: Son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

Java Database Connectivity (JDBC): es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Los **Servlets** son objetos que corren dentro del contexto de un contenedor de Servlets (ejemplo Tomcat) y extienden su funcionalidad. También podrían correr dentro de un servidor de aplicaciones. Un Servlet es un programa que se ejecuta en un servidor.

Applet: pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web.

Servidor: en informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos.

Random Access Memory (RAM): es donde el computador guarda los datos que está utilizando en el momento presente. El almacenamiento es considerado temporal por que los datos y programas permanecen en ella mientras que la computadora este encendida o no sea reiniciada.

Framework (plataforma, entorno, marco de trabajo) desde el punto de vista del desarrollo de software, un **framework** es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

Plataforma: en informática una plataforma es un determinado software y/o hardware con el cual una aplicación es compatible y permite ejecutarla.

Tecnología: es un concepto amplio que abarca un conjunto de técnicas, conocimientos y procesos, que sirven para el diseño y construcción de objetos para satisfacer necesidades humanas.

Software: en el mundo informático todo programa o aplicación, programado para realizar tareas específicas se denomina software.

Hardware: en computación, término inglés que hace referencia a cualquier componente físico tecnológico, que trabaja o interactúa de algún modo con la computadora. Es el substrato físico en el cual existe el software.

Java Server Page (JSP): es una tecnología que nos permite mezclar código HTML estático con código HTML generado dinámicamente.

Arquitectura: es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales.

Clase: descripción de un conjunto de objeto que comparten los mismos atributos, operaciones, métodos, relaciones y semántica.

Operación: servicio que puede solicitarse a un objeto para que realice un comportamiento.

Método: es la implementación de una operación que especifica el algoritmo o procedimiento de esta última.

Interfaz: conjunto de operaciones visibles en el exterior.

Portlets: son componentes modulares de interfaz de usuario gestionado y visualizado en un portal web. Los *portlets* producen fragmentos de código de marcado que se agregan en una página de un portal. Típicamente, siguiendo la metáfora de escritorio, una página de un portal se visualiza como una colección de ventanas de *portlet* que no se solapan, donde cada una de estas muestra un *portlet*. Por lo tanto un *portlet* (o colección de *portlets*) se asemeja a una aplicación web que está hospedada en un portal. Como por ejemplo, un portlet de aplicación puede ser para el correo, el parte meteorológico, un foro, noticias entre otros.

Se pretende que los estándares de los *portlets* permitan al desarrollador de software crear portlets que puedan ser utilizados en cualquier portal que soporte estos estándares.

Los portlets son similares a los Servlets en que:

- Los portlets son manejados por un contenedor especializado.
- Los portlets generan contenido dinámicamente.
- El ciclo de vida de los portlets es controlado por el contenedor.
- Los portlets interactúan con el cliente web mediante el uso del paradigma request/response.

Los portlets son diferentes a los Servlets en que:

- Los portlets son únicamente generados como fragmento de etiquetado y no como documentos completos.

- Los portlets no están asociados directamente a una URL.
- Los portlets no pueden generar contenido arbitrario, ya que el contenido de los portlets va a estar incluido en la página del portal. Si un servidor de un portal está solicitando text/html, entonces todos los portlets deben ser generados en text/html. Por otro lado si el servidor del portal está solicitando por WML, entonces cada portlet deberá ser generado en contenido WML.