Universidad de las Ciencias Informáticas

Facultad 6

Título: “Catálogo de componentes para la generación de las Vistas de Implementación de la Arquitectura de SISalud”

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autores:

Armando Ramos Roche

Esteban Urbay Mora

Tutores:

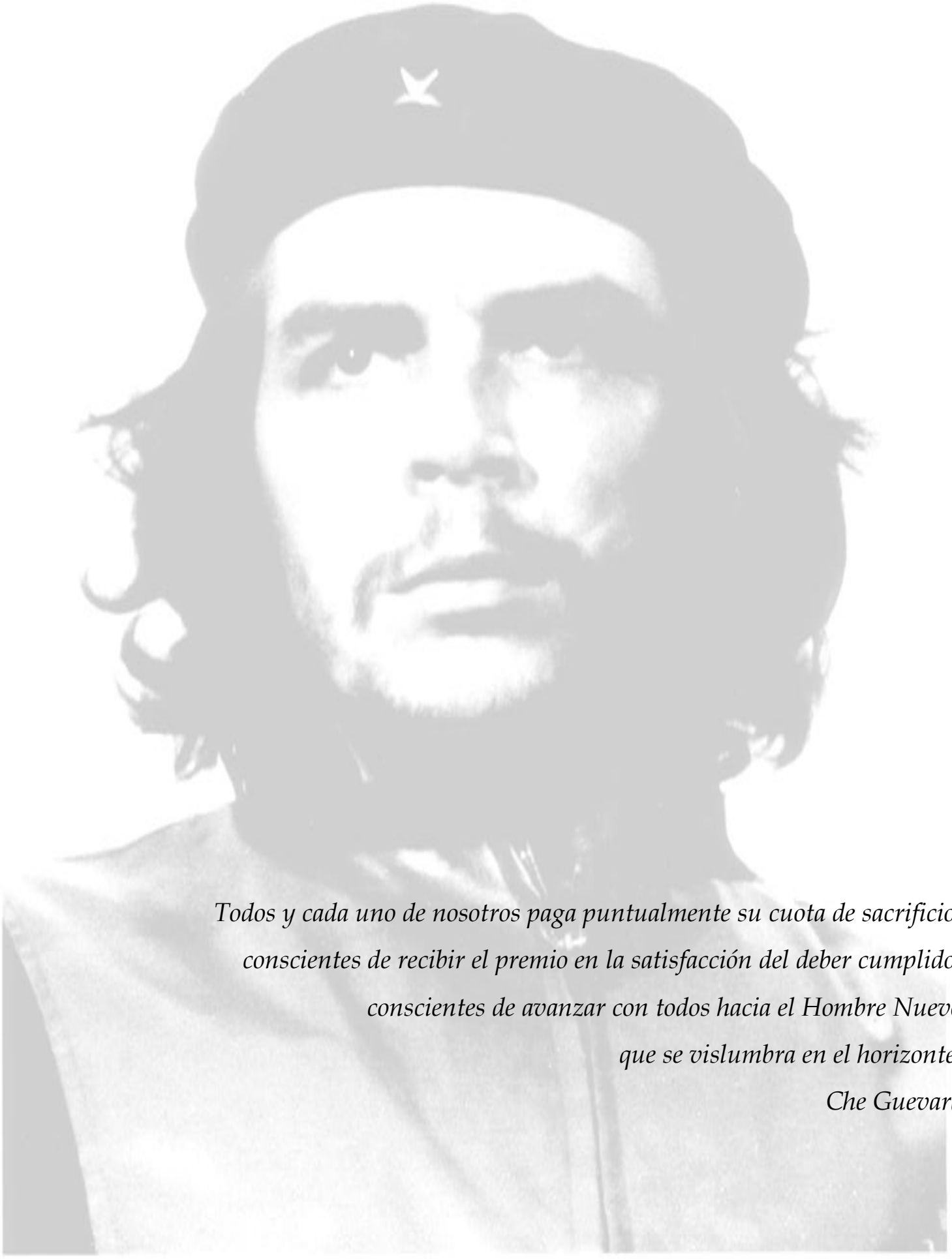
Ing. Alberto Acuña Sánchez

Ing. Otniel Barrera Palenzuela

Asesor:

Ing. Mirna Cabrera Hernández

Junio, 2009



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio,
conscientes de recibir el premio en la satisfacción del deber cumplido,
conscientes de avanzar con todos hacia el Hombre Nuevo
que se vislumbra en el horizonte.
Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Empresa SOFTEL a hacer uso del mismo en su beneficio.

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

Autores

Esteban Urbay Mora

_____

Armando Ramos Roche

_____

Tutores

Ing. Alberto Acuña Sánchez

_____

Ing. Otniel Barrera Palenzuela

_____

Asesor

Ing. Mirna Cabrera Hernández

_____

DATOS DE CONTACTO

Ing. Alberto Acuña Sánchez: Graduado de Ing. Informática en el año 1996. Posee categoría docente de Profesor Auxiliar y cursa la maestría de Gestión de Proyectos Informáticos. Cuenta con más de 12 años de experiencia en el desarrollo de aplicaciones informáticas y la gestión de proyectos. También ha impartido la asignatura Ingeniería y Gestión de Software desde el curso 2004-2005. Tiene publicaciones y ponencias en eventos científicos, así como tutorías de trabajos de diplomas de pregrado. Forma parte del Grupo de Integración de Soluciones como Arquitecto en la Empresa SOFTEL.

Ing. Otniel Barrera Palenzuela: Graduado de Ingeniero Informático en el año 1999 en el ISPJAE. Posee categoría docente de Profesor Asistente. Ha impartido las asignaturas Base de Datos y Gestión de Software en la Facultad 5 desde el año 2002. Fundador UCI. Ha presentado ponencias en eventos científicos nacionales e internacionales. Se desempeña como Arquitecto del Grupo de Integración de Soluciones en la Empresa SOFTEL.

Ing. Mirna Cabrera Hernández: Graduada de Ingeniería en Sistema Automatizado de Dirección Técnico-Económico (SAD) en el año 1986 en el ISPJAE. Posee categoría docente de Profesor Auxiliar y cursa la maestría de Gestión de Proyectos Informáticos. Ha impartido las asignaturas Gestión de Software, Perfil Salud y MIC desde el curso 2005-2006. Actualmente es profesora de Gestión de Software de la Facultad 6. Ha presentado ponencias en eventos científicos nacionales e internacionales. Se desempeña como Líder del Proyecto APS y Jefa del Grupo de Integración de Soluciones en la Empresa SOFTEL.

AGRADECIMIENTOS

A nuestros padres, tíos y abuelos nadie más que ellos hacen realidad el vernos hechos ingenieros.

A nuestro Comandante en Jefe Fidel Castro Ruz, por haber hecho realidad el sueño de ser un profesional.

A nuestros amigos y compañeros del grupo 6106, que han hecho más fácil la convivencia en la universidad.

A nuestros profesores, indudables creadores de lo que hemos podido llegar a ser.

A nuestros tutores Mirna, Otniel y Alberto.

A todos aquellos que no hemos mencionado, pero que de sobra se deben dar por aludidos por haber contribuido de un modo u otro en nuestra formación y a la consumación de este sueño.

A todos, Muchas Gracias.

DEDICATORIA

A mi mamá, que siempre me dio lo mejor del mundo y me supo guiar por el camino correcto.

A mi abuelita linda del alma por haberme ayudado toda mi vida a ser lo que soy.

A mi tía por haberme apoyado en cada parte de mi vida y de mi carrera.

A Enriquito por haber sido como mi padre durante mi vida de estudiante.

A Elizabeth por haber sido mi inspiración.

A mi Familia que siempre me dio apoyo para que continuara adelante.

A mis amigos y compañeros de aula por haber compartido tantos años de estudio con ellos.

A todos los que de una forma u otra me tendieron las manos y me ayudaron durante toda mi vida.

Esteban

A la persona que sus brazos siempre se abren cuando necesito un abrazo, que su corazón sabe comprender cuando necesito una amiga, que sus ojos sensibles se endurecen cuando necesito una lección, la que su fuerza y su amor me han dirigido por la vida, a la persona que le debo todo lo que soy, la mujer más bella que he conocido jamás, a ti mi mami te dedico este trabajo.

Armando

RESUMEN

El presente trabajo de diploma tiene como objetivo el desarrollo de la aplicación Web “Catálogo de Componentes para la generación de las Vistas de Implementación de la Arquitectura de SISalud”, sistema que aporta un conjunto de beneficios al proceso de desarrollo de las aplicaciones informáticas para la informatización del Sistema Nacional de Salud en Cuba, ya que permitirá al proceso de integración de estas aplicaciones, contar con las relaciones de dependencia a partir de la representación de las vistas de implementación de la Arquitectura del Sistema de Informatización para la Salud (SISalud) y con esto contribuir como resultado a la agilización en el trabajo de los diferentes equipos de desarrollo, que pueden obtener mediante el sistema, información relacionada con la arquitectura y un espacio para generar y transferir el conocimiento necesario para la continuidad del trabajo.

En el documento se expone el estudio de otros sistemas que muestren vistas de implementación como antecedentes, se fundamenta la selección de las metodologías, herramientas, tecnologías utilizadas y se presenta la propuesta de sistema a través de los resultados de la investigación realizada. Por último, se exponen los artefactos generados en el diseño e implementación. Para la elaboración de la documentación del sistema propuesto se tuvo en cuenta el proceso de desarrollo RUP, haciendo uso del lenguaje UML.

Palabras Clave: Sistema de Información para la Salud, Vistas, Implementación.

TABLA DE CONTENIDOS

Índice de Figuras

Índice de Tablas

Introducción

El mejoramiento de la infraestructura tecnológica y la profunda preparación del capital humano desde edades tempranas, son ejemplos de grandes esfuerzos del Estado Socialista Cubano por transitar aceleradamente hacia la informatización de la sociedad Cubana, como vía para aumentar la calidad de vida, la eficiencia y la competitividad del país.

Cuba enfrenta el reto de informatizar la sociedad con vistas de integrarse plenamente a la infraestructura global de la información, así como hacer un uso óptimo de las nuevas tecnologías, como parte de la Batalla de Ideas y del proceso de informatización de la sociedad. El Ministerio de Salud Pública (MINSAP) es el organismo rector del Sistema Nacional de Salud encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en cuanto a la Salud Pública, el desarrollo de las Ciencias Médicas y la Industria Medico Farmacéutica, desarrolla múltiples tareas orientadas por la dirección del país y uno de los programas que más expectativas y posibilidades abre en el campo del conocimiento, la información científica y la asistencia médica. (1)

Durante los últimos años el propio MINSAP ha definido la informatización del sector como fundamental, en busca de la optimización de los servicios de salud. En todos los casos el objetivo ha sido proveer al Sistema Nacional de Salud de información confiable, consistente y oportuna para la toma de decisiones y el mejoramiento de los procesos médicos asistenciales. Garantizar de esta manera el incremento en la calidad y seguridad de la atención médica a la población. (2)

A partir del año 2003 se traza una nueva estrategia para alcanzar la informatización del sector, alineada al proceso de informatización de la Sociedad Cubana, poniendo como centro de la misma al paciente y utilizando para su construcción las últimas tecnologías de la información y las comunicaciones para el desarrollo de aplicaciones, sobre una arquitectura basada en componentes y orientado a servicios.

Para llevar a cabo este plan fue necesario seleccionar una empresa capaz de cumplir con las exigencias y metas propuestas, por lo que se decide que la Empresa de Soluciones Informáticas SOFTEL, entidad del Ministerio de Informática y las Comunicaciones (MIC), que reorientara su trabajo hacia el desarrollo de productos y servicios informáticos, elevando la eficiencia del sistema de salud cubano y trabajando para ganar un espacio en el mercado internacional. Para el desarrollo de su labor, la empresa dispone de consultores especializados en diseño, implantación y gestión de soluciones,

cobertura nacional de servicios y soporte técnico, así como especialistas de gran profesionalidad encargados del desarrollo de sistemas informáticos para el sector de la salud. También se integran a esta empresa dos grupos que trabajan en la informatización de la salud en Cuba: la Universidad de las Ciencias Informáticas (UCI), y el Centro de Diseño y Desarrollo para la Salud Pública (CEDISAP).

Formando parte de la estructura de la Empresa SOFTEL, en el Grupo de Integración de Soluciones (GIS) se encuentra profesionales encargados guiar y avalar la integración del grupo de componentes que se desarrollan para el Sistema Nacional de Salud, además de brindar soporte tecnológico para garantizar la integración de las aplicaciones que conforman una solución informática.

En el desarrollo de las aplicaciones informáticas para la salud, las vistas de implementación de los componentes desarrollados forman parte de la documentación interna de cada grupo de desarrollo, por lo que el expediente de cada proyecto no es accesible para el resto de los desarrolladores de la comunidad que participan en el proceso de informatización de los procesos del Sistema Nacional de Salud, ni para el Grupo de Integración de Soluciones, responsabilizado con la evaluación y aceptación de la integración de los componentes del SISalud, y por el cumplimiento de la arquitectura propuesta y los estándares y políticas definidas por el MINSAP.

Por lo cual se define como **problema a resolver:** ¿Cómo generar las vistas de implementación de los componentes que forman la arquitectura de SISalud para que garantice la gestión de la información del grupo de integración de la Empresa SOFTEL?
En función del problema se define como **objeto de estudio:** el proceso de desarrollo de software y como **campo de acción:** se define las disciplinas de diseño e implementación del proceso de desarrollo de software de la Empresa SOFTEL.
Para dar respuesta al problema se propone como **objetivo general:** Diseñar e Implementar una aplicación web que garantice generación de las vistas de implementación de los componentes desarrollados para el Sistema Nacional de Salud.

Para darle cumplimiento a lo anterior se definen los siguientes **objetivos específicos:**

1. Estudiar la arquitectura del Sistema Nacional de Salud.
2. Identificar las funcionalidades que debe brindar el sistema.
3. Diseñar la aplicación.
4. Implementar la aplicación.

Para dar cumplimiento al objetivo general de la investigación se trazaron las siguientes **tareas de investigación:**

1. Estudio de la Arquitectura definida por el MINSAP para el desarrollo de sus aplicaciones.
2. Investigación del estado del arte sobre el tema de la generación de las vistas de implementación de la arquitectura de SISalud.
3. Realización de la Disciplina de Requerimientos.
4. Realización de la Disciplina de Diseño.
5. Realización de la Disciplina de Implementación.
6. Implementación de la aplicación web “Catálogo de Componentes para la generación de las vistas de Implementación de la Arquitectura de SISalud”.

Resultados esperados y aporte práctico.

Con el desarrollo de esta aplicación la Empresa SOFTEL tendrá la posibilidad de minimizar la falta de comunicación entre los diferentes grupos de desarrollo, ya que se pública información relacionada con las interacciones entre los diferentes módulos, ganando así en tiempo y eficacia. Permitiendo una mejor toma de decisión a la hora de implementar los diferentes módulos y reutilizar lo que ya existe.

El trabajo se estructura en 4 capítulos en los que daremos solución a la problemática planteada.

En el **Capítulo 1**: **Fundamentación Teórica**: El objetivo fundamental de este capítulo es abordar distintos aspectos que se utilizan como soporte teórico del sistema diseñado. Se exponen a través de una descripción los conceptos asociados al problema. Además se define el objeto de estudio, la situación problémica, los objetivos generales y específicos de la presente investigación.

En el **Capítulo 2**: **Características del Sistema:** En este capítulo se realiza la descripción de la propuesta de solución para el Catálogo de Componentes para la generación de las Vistas de Implementación de la Arquitectura de SISalud. Por otra parte, se enumeraron los requerimientos tanto como funcionales como no funcionales, agrupándose los primeros en Casos de Uso del Sistema.

En el **Capítulo 3: Diseño del Sistema:** En este capítulo se modela y adquiere forma el sistema para que soporte todos los requisitos funcionales o no funcionales e inclusive cualquier otro tipo de restricción, contribuyendo a obtener una arquitectura sólida y estable para la futura implementación del sistema de software. Entre los artefactos que serán mostrados en el presente capítulo se encuentran: Diagramas de Clases y Descripción de las Clases de Diseño.

En el **Capítulo 4: Implementación del Sistema:** El presente capítulo contiene las principales características de la implementación del Sistema. Además de los principales componentes y sus relaciones, mostrados de forma sencilla a través del Diagrama de Componentes.

Capítulo 1: Fundamentación Teórica.

El desarrollo de la Informática tiene un papel fundamental para el desarrollo de los diversos sectores en la sociedad. La Informatización del Sistema Nacional de la Salud no está excepto de ello, por lo que ha permitido una mayor eficiencia en el funcionamiento de este sector, unos de los logros de la Revolución Cubana.

El objetivo fundamental de este capítulo es abordar distintos aspectos que se utilizan como soporte teórico del sistema diseñado. Se exponen a través de una descripción los conceptos asociados al problema. Además se define el objeto de estudio, la situación problémica y los objetivos generales y específicos de la presente investigación.

1.1 Sistema Nacional de Salud.

La garantía de la atención médica gratuita a toda la población cubana se convirtió desde los primeros momentos del triunfo de la Revolución en uno de los paradigmas sociales fundamentales. Esto se corresponde con la esencia humanista y de justicia social que caracteriza a nuestro pueblo revolucionario. (3)

Desde el propio triunfo revolucionario se adoptaron medidas para transformar la salud pública en Cuba, una de las principales y novedosa fue la creación del Sistema Nacional de Salud (SNS), designándose al Ministerio de Salud Publica como su organismo rector. (4)

La estructura organizativa creada comenzó a realizar importantes reformas a partir de los años 60, como parte fundamental de las transformaciones del período revolucionario y en respuesta al respecto más absoluto de uno de los derechos humanos fundamentales de todo ciudadano. Surge el servicio de hospitales rurales llevando la atención medica a zonas apartadas de la geografía nacional, se dan los primeros pasos para el fortalecimiento de la atención primaria; surgen los policlínicos integrales como unidad asistencial creada para brindar servicios y resolver los principales problemas existentes en los primeros años de la revolución.

A partir de los años 60 y hasta la actualidad se han hecho varias reformas, en 1996, el SNS adoptó desde el punto de vista organizativo, estrategias fundamentales y priorizó cuatro programas básicos para continuar perfeccionándose: Programa de Atención Materno Infantil (PAMI), el de control de enfermedades trasmisibles, el de control de enfermedades crónicas no trasmisibles, y el de atención al

adulto mayor, todos los que han sido monitorizados, controlados y evaluados de acuerdo a la metodología establecida. (5)

El Programa del Médico y la Enfermera de la Familia, se ratifica como eje del actual desarrollo estratégico del MINSAP. (6)

Este modelo de atención es la mayor fortaleza y potencia que tiene el SNS. Por su existencia, filosofía, bases teóricas y lo que ha podido proporcionarle al sistema de salud, se ha logrado mantener los indicadores de salud y satisfacer las necesidades de la población constituyendo un pilar básico de la Salud Pública Cubana. (7)

Con más de 20 años de experiencia en este programa se comienza a experimentar cambios en la atención primaria, servicios que antes eran excluidos de hospitales son abiertos en instituciones de la atención primaria, surgiendo así hace 5 años aproximadamente, el nuevo modelo de policlínico con nuevas funciones, acercando más los servicios a la población, para hacer realidad las palabras de nuestro Comandante en Jefe cuando dijo: **"…** ***una profunda revolución en los servicios de salud tendrá lugar en nuestra Patria…*****".** (8)

1.1.1 Sistema de Información para la Salud (SISalud).

El Sistema de Información para la Salud (SISalud), forma parte de la Informatización del Sistema Nacional de Salud. Este sistema propone que todas las aplicaciones informáticas para la salud estén integradas; abarca: Administración del Sistema, Registros Básicos y Codificadores, Sistemas de Atención Médica para los niveles de Atención Primaria, Secundaria y Terciaria, Ayuda a la decisión y Otros Sistemas, mostrados en el Anexo 2.

Como Registros Básicos y Codificadores se incluye el Registro Informatizado de Salud (RIS) que contiene los nomencladores nacionales que brindarán información para el funcionamiento del resto de las aplicaciones. Se puede citar para gestionar los recursos: Registro de Unidades de Salud (RUS), Registro Personal de la Salud (RPS), Registro del Ciudadano (RC), Registro de Equipos Médicos y No Médicos (REQ) y en los que gestionan los servicios: Registro de Servicios Médicos (RSM).

Con estos registros desarrollados e implementados se garantiza por primera vez la gestión en tiempo real y con alcance nacional de la información referida a los principales recursos y servicios de la salud, así como la obtención de un grupo de estadísticas para ser analizadas a cualquier instancia de dirección del SNS para la toma de decisiones. Son capaces de interactuar entre ellos y de esta forma reutilizar la información gestionada por cada uno, y por el resto de las aplicaciones que conforman SISalud. (9)

1.1.2 Empresa de Soluciones Informáticas (SOFTEL).

La Empresa SOFTEL, entidad del Ministerio de Informática y las Comunicaciones (MIC) que reoriento su trabajo al desarrollo de productos y servicios informáticos, elevando la calidad del sistema de salud de nuestro país y trabajando para alcanzar un espacio en el mercado internacional. Dispone en ella consultores especializados en diseño, implementación y gestión de soluciones, cobertura nacional de servicios y soporte técnico, así como especialistas de gran profesionalidad encargados del desarrollo de sistemas informáticos para el sector de la salud.

1.1.3 Grupo de Integración de Soluciones Informáticas (GIS).

El GIS está conformado por un grupo de profesionales de la Empresa SOFTEL que se encargan de brindar soporte tecnológico para garantizar la integración de las aplicaciones que conforman una solución informática. Teniendo como objetivos fundamentales:

- ✓ Garantizar que los nuevos desarrollos y mantenimientos cumplan con las estrategias definidas para lograr la integración de las aplicaciones en el Marco Regulatorio de la DI_ MINSAP.
- ✓ Validar el cumplimiento de la arquitectura definida en las aplicaciones que conforman una solución informática.
- ✓ Validar en un ambiente de pruebas, la integración de las aplicaciones que conforman una solución informática.
- ✓ Validar y gestionar los cambios que ocurren en los componentes ya implementados.
- ✓ Llevar el control de los productos y mantenimientos que no cumplan con las estrategias definidas para la integración de las aplicaciones que conforman una solución informática. (10)

1.2 Situación Problémica y problema a resolver.

El Sistema de Información para la Salud (SISalud) surge como la solución informática integral para la Salud pública, acorde con los objetivos de informatización de la sociedad cubana, constituyendo la plataforma de administración, el procesamiento y la trasmisión de la información para el Sistema Nacional de Salud y está formado por un conjunto de aplicaciones independientes (Módulos del Sistema) que se interconectan según las necesidades del flujo de información. Entre ellos se encuentran los módulos que integran el Registro Informatizado de Salud, los módulos que responden a las funcionalidades de los diferentes niveles de Atención Medica (Atención Primaria, Atención Hospitalaria y Especializada) y los módulos que responden al resto de las funcionalidades administrativas del Sistema Nacional de Salud.

De lo anterior se deriva la alta complejidad estructural del flujo informativo que se modela para dar cobertura a esta solución informática, además del creciente desarrollo en paralelo de equipos multidisciplinarios para la implementación de soluciones para un mismo cliente.

En la actualidad las vistas de implementación de los componentes desarrollados forma parte de la documentación interna de cada grupo de desarrollo, por lo que no esta accesible al resto de los desarrolladores de la comunidad que informatizan los procesos del Sistema Nacional de Salud y del grupo de arquitectura MINSAP – MIC responsabilizado con la integración y definición de esta arquitectura.

Por otra parte el actual desarrollo parcial del portal contiene un conjunto de componentes donde no están documentadas sus interrelaciones, y durante el proceso de mantenimiento no se actualiza la documentación haciéndose demasiado engorroso el proceso de revisión de la documentación necesaria para extraer la información de las relaciones entre componentes.

Para manejar la complejidad del sistema y propiciar los desarrollos paralelos efectivos se hace necesario contar con un Catálogo para la generación de las vistas de implementación de cada uno de los componentes que integran la arquitectura definida por el MINSAP para el desarrollo de las aplicaciones.

1.3 Sistemas de Generación de Vistas de Implementación.

Debido a la importancia que han adquirido los sistemas de gestión de información, por el gran valor que aportan a las entidades, se han hecho imprescindibles para las mismas. Actualmente en el mundo se pueden encontrar aplicaciones que realizan generación de Vistas de Implementación como por ejemplo:

1.3.1 Sistemas de Generación de Vistas de Implementación en el mundo.

Rational Rose: Es una herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Visual Paradigm: Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y

a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Enterprise Architect: Herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener.

Estos son algunos ejemplos de herramientas, que al igual que nuestra solución planteada al problema a resolver generan vistas de implementación pero no se corresponden del todo con la necesidad total de este problema, de manera que no pueden llegar a mostrar las vistas de implementación de los componentes que integran la arquitectura del Sistema de Informatización para la Salud.

1.3.2 Sistemas de Generación de Vistas de Implementación en Cuba y la UCI.

En nuestro país y en la UCI no existen sistemas que generen de Vistas de Implementación por lo que nuestro sistema será el primero en desarrollarse para el apoyo de los grupos de desarrollos que desarrollan para el MINSAP con el objetivo de facilitarle la documentación de las relaciones entre los componentes que existen.

1.4 Roles que participan en el Desarrollo de la Aplicación.

Para el desarrollo de este trabajo se decidió seleccionar varios roles, los cuales aportarán para el desarrollo de la aplicación entre los cuales se encuentran:

Analista de Sistema: Dirige y coordina la adquisición de requisitos esquematizando la funcionalidad del sistema y delimitándolo.

Entre las principales tareas que este rol desempeña se encuentran esencialmente:

- ✓ Encontrar actores y casos de uso.
- ✓ Estructurar el modelo de caso de uso.
- ✓ Priorizar los casos de uso.
- ✓ Definir el contexto de sistema.

Del desarrollo de esas tareas este rol crea entonces los artefactos siguientes:

- ✓ Glosario.
- ✓ Modelo de casos de uso.
- ✓ Modelo de Análisis.

Especificador de Requisitos: Específica y mantiene los requisitos del sistema detallados.

- ✓ Entre las principales tareas que realiza se encuentran:

- ✓ Detallar los requisitos de sistema.
- ✓ Detallar los casos de uso.

Arquitecto de Software: Dirige el desarrollo de la arquitectura de software del sistema, que incluye la promoción y la creación de soporte para las decisiones técnicas clave que restringen el diseño global y la implementación para el proyecto.

Entre las principales tareas que realiza se encuentran:

- ✓ Priorizar los casos de uso del sistema.
- ✓ Identificar mecanismos de diseño.
- ✓ Identificar elementos de diseño.
- ✓ Análisis de la arquitectura.
- ✓ Describir la distribución.
- ✓ Estructurar el modelo de implementación.

Las tareas que realiza tienen como resultado los siguientes artefactos:

- ✓ Modelo de despliegue.
- ✓ Modelo de diseño.
- ✓ Modelo de implementación.

Diseñador: El Diseñador dirige el diseño de una parte del sistema, dentro de las restricciones de los requisitos, arquitectura y proceso de desarrollo para el proyecto.

Realiza las siguientes tareas:

- ✓ Diseño de caso de uso.
- ✓ Diseño de clase.
- ✓ Diseño del subsistema.

Y genera los siguientes artefactos:

- ✓ Realización de casos de uso.
- ✓ Clase de diseño.
- ✓ Modelo de análisis.
- ✓ Modelo de diseño.
- ✓ Paquete de diseño.
- ✓ Subsistema de diseño.

Implementador: Desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes, de acuerdo con los estándares adoptados de proyecto. Sus tareas esenciales son:

- ✓ Implementar elementos de diseño.

Los artefactos que genera son:

- ✓ Subsistema de implementación.
- ✓ Elemento de implementación.

1.5 Herramientas y Tecnologías.

Existen creencias de que un grupo de desarrollo debería organizarse en torno a las habilidades de los individuos altamente calificados, que saben cómo hacer el trabajo y lo hacen bien, y que raramente necesitan dirección. Esto constituye un error en la mayoría de los casos, y una grave equivocación en el caso de desarrollo de software. Por lo tanto, es necesario un proceso que esté ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentran implicados. Todo esto unido a una correcta selección de las herramientas, métodos, técnicas y procedimientos que ayuden a obtener un producto de elevada calidad.

1.5.1 Metodología de Desarrollo de Software.

Con el propósito de solucionar unos de los principales retos que afrontaban los desarrolladores de software, se crearon las diferentes metodologías de desarrollo las cuales son un conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software con calidad y eficiencia. Obteniendo cada año un aumento en el mejoramiento y productividad de estas metodologías.

Elegir la metodología adecuada para realizar este ciclo de desarrollo es vital para lograr un sistema de alta calidad en un tiempo razonablemente corto. Existen varias metodologías de desarrollo de software, dentro de las cuales están Microsoft Solution Framework (MSF) (11), eXtreme Programming (XP) (12)y Proceso Unificado de Desarrollo (RUP)(13).

Esta ultima definida por ser constituir la metodología estándar más utilizada para el Análisis, Implementación y Documentación de sistemas orientados a objetos. También por ser adaptables al contexto y necesidades de cada organización. En su modelación define como sus principales elementos:

Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo o grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Flujo de Actividades ("cuándo"): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

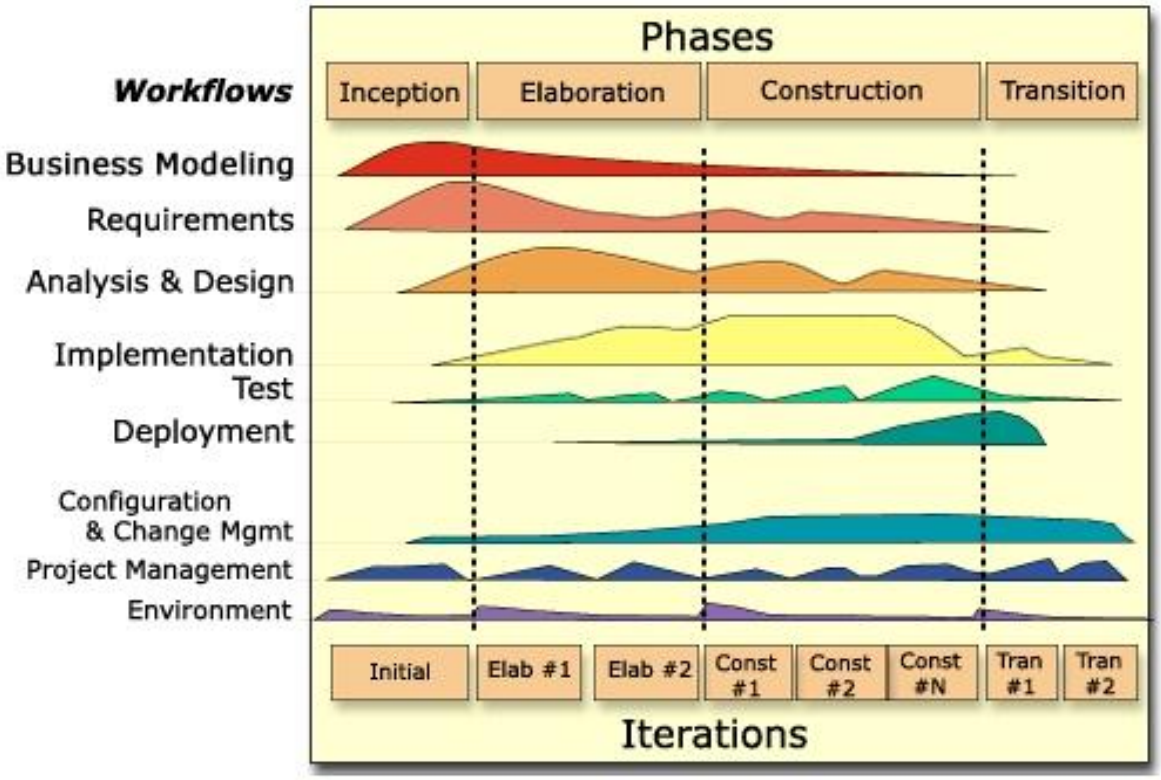


Figura 1: Proceso Unificado de Desarrollo de Software.

Flujos de trabajo:

- ✓ **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- ✓ **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- ✓ **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- ✓ **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- ✓ Prueba (Testeo): Busca los defectos a lo largo del ciclo de vida.

- ✓ **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- ✓ **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- ✓ **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- ✓ **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Describe en su desarrollo cuatro fases:

- ✓ **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los CU del sistema.
- ✓ **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los CU que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- ✓ **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.
- ✓ **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores. (14)

1.5.2 Lenguaje de modelado

Para dar solución al problema planteado, usaremos como lenguaje de modelado UML (15) (16). El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos

semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes.
La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

1.5.3 Herramienta CASE.

Las herramientas CASE son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información, completamente o en algunas fases.
Como ejemplo de herramientas CASE tenemos MagicDraw (17), Umbrello (18), Rational Rose (19), y Visual Paradigm for UML. Por decisión se decidió a utilizar Visual Paradigm for UML por ser una herramienta CASE que soporta la última versión del Lenguaje de Modelado Unificado (UML), la Notación del Proceso de Modelado de Negocio (BPMN), y genera código para un gran número de lenguajes de programación.
La herramienta fue desarrollada para una amplia gama de usuarios incluyendo Ingenieros de Software, Analistas de Sistemas, Analistas del Negocio y Arquitectos de Sistemas. Permite la integración con varias herramientas; brinda además una gran interoperabilidad con otras herramientas CASE como Rational Rose.

1.5.4 Lenguaje de Programación y Entorno de Desarrollo Integrado (IDE).

1.5.4.1 PHP, lenguaje de programación web del lado del servidor.

Se seleccionó como lenguaje de programación del lado del servidor a PHP por ser un lenguaje de programación web interpretado. Es diseñado para la creación de páginas web dinámicas. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Entre sus ventajas, tiene capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad. Destaca su capacidad de expandir su potencial utilizando la enorme cantidad de módulos. Permite las técnicas de Programación Orientada a Objetos y una biblioteca nativa de funciones sumamente amplia.

1.5.4.2 JavaScript, lenguaje de programación web de lado del cliente.

Como lenguaje de programación se decidió JavaScript por ser utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. También fueron utilizadas framework como Ext JS y JQuery.

- ✓ **Ext JS**

  Ext JS es una librería Javascript ligera y de alto rendimiento, compatible con la mayoría de navegadores para crear páginas web dinámicas. Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).

- ✓ **jQuery**

  Es un framework Javascript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (Document Object Model), manejar eventos, desarrollar animaciones y agregar interacción con la tecnología Asynchronous JavaScript And XML (AJAX) a páginas web.

1.5.4.3 Zend Studio.

Este IDE fue seleccionado por ser un completo entorno integrado de desarrollo para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Además fue elegido por no requerir previa instalación de PHP, por su soporte de PHP 4 y PHP 5, también autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase. Aporta al desarrollador inserción automática de paréntesis y corchetes de cierre, sangrado automático y otras ayudas de formato de código y Detección de errores de sintaxis en tiempo real.

1.5.5 MySQL, Sistema Gestor de Bases de Datos

En la selección de gestores de Bases de Datos se decidió hacer uso de MySQL por ser un sistema de gestión de base de datos relacional, multihilo y multiusuario. También por ser muy utilizado en aplicaciones web, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python). MySQL es una base de datos muy rápida en la lectura, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Conclusiones del Capítulo

En este capítulo se realizó un estudio de la situación problemica que nos concierne y de esta forma centrando como objetivo principal, el estudio del arte de sistemas similares en Cuba y el Mundo que podrían brindar solución a la problemática abordada. Se definió la metodología a usar así como los lenguajes y herramientas para el desarrollo de la aplicación. También se hizo una selección estricta de los patrones a utilizar tanto en la arquitectura como en el diseño del futuro sistema**.**

Capítulo 2: Características del Sistema.

En este capítulo se realiza la descripción de la propuesta de solución para el Catálogo de Componentes para la generación de las Vistas de Implementación de la Arquitectura de SISalud. Debido a que los procesos visibles no estaban bien definidos se determina desarrollar un Modelo de Dominio, donde se expone un marco conceptual y las relaciones entre estas definiciones. Por otra parte, se identificaron los requerimientos tanto funcionales como no funcionales, agrupándose los primeros en Casos de Uso del Sistema.

2.1 Modelo de Dominio

El Modelo de Dominio es un artefacto de la disciplina de análisis, durante la fase de concepción, teniendo como objetivo principal la comprensión y la descripción de las clases más importantes dentro del contexto donde se desarrolla el sistema. En la tarea construcción del Modelo de Dominio, es representado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física.

Los Modelos de Dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Es también utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir.

El diagrama del modelo de dominio permite visualizar las distintas asociaciones que tienen las entidades.

Figura 2: Diagrama del Modelo del Dominio

2.2 Conceptos Fundamentales.

Paquete de Componentes: Esta concebido en 3 capas (Presentación, Negocio, Acceso a datos) y cada una están compuestos por componentes con diferentes relaciones de dependencia entre sí ya sea entre componentes de diferentes capas o de diferentes paquetes de componentes.

Componente: Es una unidad básica que puede ser, un fichero HTML, un servicio web, un fichero PHP, una entidad.

2.3 Especificación de los Requerimientos de Software.

2.3.1 Requerimientos Funcionales.

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

- R1. Gestionar Paquete de Componente.
  - R1.1. Insertar Paquete.
  - R1.2. Modificar Paquete.
  - R1.3. Eliminar Paquete.
- R2. Gestionar Relaciones entre Paquetes.
  - R2.1. Insertar Relación.
  - R2.2. Modificar Relación.
  - R2.3. Eliminar Relación.
- R3. Gestionar Componentes.
  - R3.1. Insertar Componente.
  - R3.2. Modificar Componente.
  - R3.3. Eliminar Componente.
  - R3.4. Filtrar Componente
- R4. Generar Diagramas.
  - R4.1. Generar la Matriz de Impacto.
  - R4.2. Generar el Diagrama de Componentes.
    - R4.2.1. Generar el Diagrama de Global del Sistema.
    - R4.2.2. Generar el Diagrama por Paquete de Componentes específico.
- R5. Gestionar Usuarios.
  - R5.1. Insertar Usuario.
  - R5.2. Modificar Usuario.
  - R5.3. Eliminar Usuario.

R5.4. Filtrar Usuario.

R6. Autenticar Usuarios.

R6.1. Cambiar Contraseña.

2.3.2 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 al producto atractivo, usable, rápido o confiable.

Software

- ✓ Sistema Operativo GNU/Linux, Windows 98 o superior
- ✓ Las computadoras clientes de los usuarios accederán al sistema utilizando los navegadores: Mozilla Firefox, Opera, Google Chrome.

Hardware

- ✓ PC Cliente: Pentium 4, Microprocesador 1,7 GHz o superior, 256 MB mínimo de memoria RAM y una tarjeta de red.
- ✓ PC Servidora: Pentium 4, Microprocesador superior a 1,7 GHz, con un espacio libre como mínimo en el disco duro de 300 MB y una tarjeta de red.

Apariencia o interfaz externa

- ✓ El sistema tendrá una apariencia amigable propia de una interfaz web con colores de bajos tonos.
- ✓ Las páginas de la aplicación web no se cargaran con mucha información, y contendrán solo las imágenes necesarias.

Usabilidad

- ✓ La aplicación tendrá un ambiente sencillo y será fácil de manejar por los usuarios, incluso aquellos que no han tenido mucha experiencia en el trabajo con computadoras o con sistemas informáticos.

Rendimiento

- ✓ Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

Soporte

- ✓ Se impartirá una preparación a los usuarios finales con la explicación de cómo realizar el trabajo con el software.

Portabilidad

- ✓ El sistema podrá ser usado sobre los sistemas operativos Windows y Linux.

Seguridad

- ✓ Las contraseñas deberán tener más de 7 y menos de 20 caracteres de longitud.
- ✓ El sistema no será utilizado en el caso de que el usuario no se encuentre autenticado. autenticado.

2.4 Modelo de Casos de Uso del Sistema.

El modelado de casos de uso es una técnica efectiva y a la vez simple para modelar los requerimientos del sistema desde la perspectiva del usuario. Presenta el sistema desde la perspectiva de su uso y esquematiza cómo proporcionará valor a sus usuarios. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores para limitar las funciones con que dispondrá el sistema luego de ser implementado, además proporciona la entrada fundamental para el análisis, el diseño, la implementación y las pruebas.

2.4.1 Definición de Actores.

Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.
A continuación se presenta los actores del sistema con su descripción.

Actor	Descripción
Administrador	El administrador una especialización del actor Usuario de Sistema el cual es el encargado de administrar el sistema, gestionando los usuarios y los paquetes de componentes que serán utilizados en la aplicación.
Ingeniero en Componente	Es una especialización del actor Usuario de Sistema con determinados privilegios que es el encargado de realizar algunas

	funciones como Gestionar Relaciones, Generar Diagramas y Gestionar Componentes.
Usuario del Sistema	Es una generalización del Administrador y del Ingeniero de Componentes los cuales van a realizar una misma acción: Autentificarse.

Tabla 1: Tabla de Actores

2.4.2 Diagrama de Casos de Uso.

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

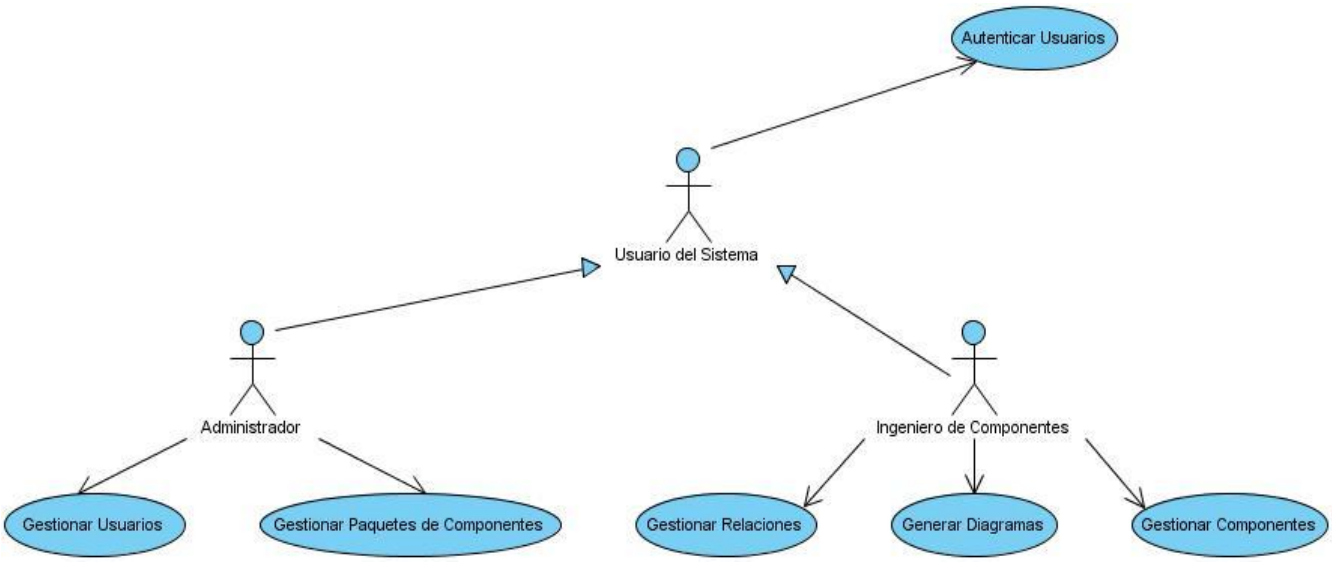


Figura 3: Diagrama de Casos de Uso

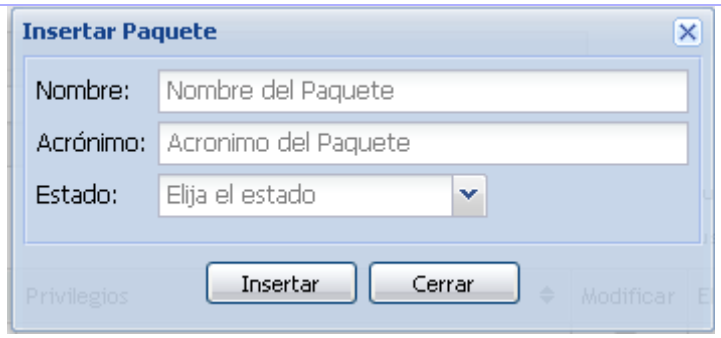
2.4.3 Descripción Textual de los Casos de Uso.

2.4.3.1 Descripción Textual de los "Casos de Uso Gestionar Paquetes de Componentes".

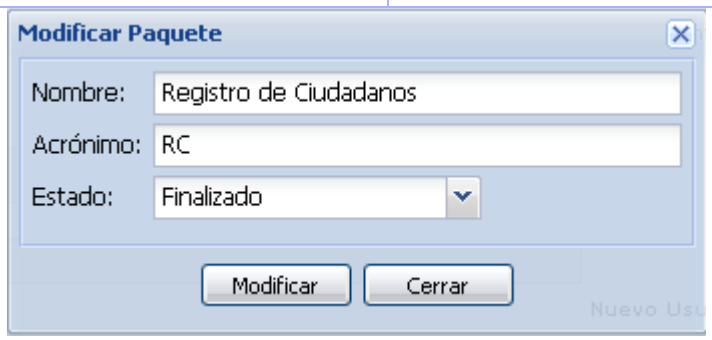
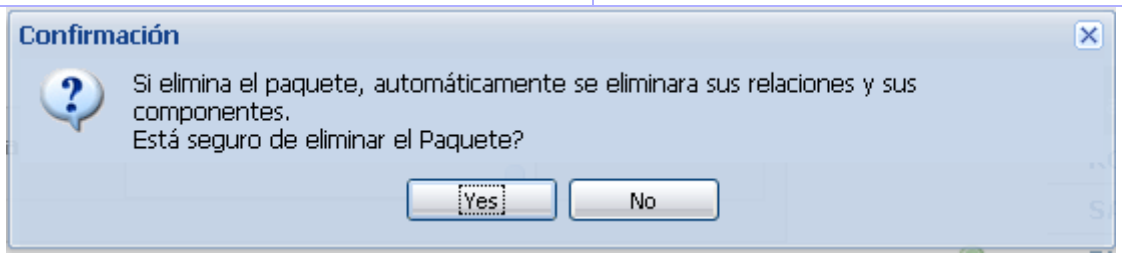
Caso de Uso:	**Gestionar Paquete de Componentes**
Actores:	**Administrador**

Resumen:	**El CU se inicia cuando el Administrador decide Insertar, Modificar o Eliminar un Paquete de Componentes en el sistema.**
Precondiciones:	**Existencia real del Paquete en SISalud.** **Estar autenticado el Administrador que es el encargado de inicializar el caso de uso Gestionar Paquetes de Componentes.**
Referencias	**R1, R1.1, R1.2, R1.3**
Prioridad	

Flujo Normal de Eventos	
1. El usuario selecciona el menú Paquetes	**2. El sistema muestra un menú con las opciones.**
3. El usuario escoge una de las siguientes opciones: **a) Insertar Paquete.** **b) Modificar Paquete.** **c) Eliminar Paquete.**	
Sección "Insertar Paquete de Componentes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita Insertar Paquete en el signo de "+" del menú Paquetes.	**2. Muestra una ventana con el formulario correspondiente.**
3. El usuario inserta en el formulario los datos del Paquete (nombre, acrónimo, estado) y presiona el botón Insertar.	**4. El sistema valida los datos de entrada.**
	5. El sistema verifica que el paquete no exista ya en el sistema.
	6. El Sistema muestra una mensaje "Se ha insertado el paquete correctamente"
7. El usuario presiona Cerrar y termina la acción.	



Flujos Alternos Sección "Insertar Paquete de Componentes"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si hay errores en la validación de los datos muestra el mensaje "Llene bien los campos del formulario", y pasa a la acción #2 de la sección Insertar Paquete de Componentes.
	5.1 Si ya existe el Paquete el sistema muestra el mensaje "No se ha insertado el paquete" y regresa a la acción #2 de la sección Insertar Paquete de Componentes.
Sección "Modificar Paquete de Componentes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita el Paquete de Componente en el menú Paquete; selecciona el paquete que desea y presiona en Modificar.	**2. El sistema muestra el formulario con los datos actuales de ese Paquete**
3. El usuario modifica los datos (nombre, acrónimo, estado).	**4. El sistema verifica los cambios realizados.**
	5. El sistema valida los datos de entrada.
	6. El sistema muestra un mensaje "Se ha modificado el paquete correctamente".

7. El usuario presiona Cerrar y termina la acción.	
	
Flujos Alternos Sección "Modificar Paquete de Componentes"	
Acción del Actor	**Respuesta del Sistema**
	5.1 Si hay errores en la validación de los datos muestra el mensaje "Error en los datos de entrada", y pasa a la acción #2 de la sección Modificar Paquete de Componentes.
Sección "Eliminar Paquete de Componentes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita el Paquete de Componente en el menú Paquete; selecciona el paquete que desea y presiona en Eliminar.	**2. El sistema elimina el Paquete del sistema.**
	3. El sistema muestra un mensaje de confirmación
4. El usuario presiona Yes.	**5. El sistema muestra el mensaje "Se ha eliminado el paquete correctamente"**
	
Flujos Alternos Sección "Eliminar Paquete de Componentes"	

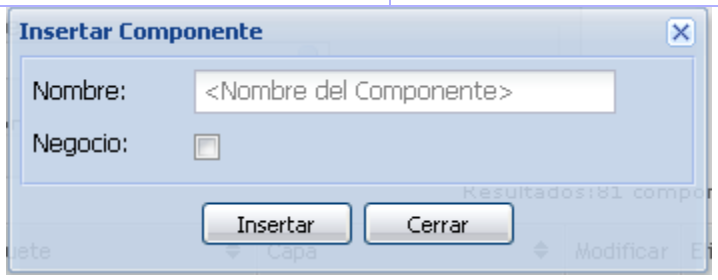
Acción del Actor	Respuesta del Sistema
1.1 El usuario decide no eliminar el Paquete.	
Poscondiciones	

Tabla 2: Descripción Textual de los "Casos de Uso Gestionar Paquetes de Componentes".

2.4.3.2 Descripción Textual de los "Casos de Uso Gestionar Componentes".

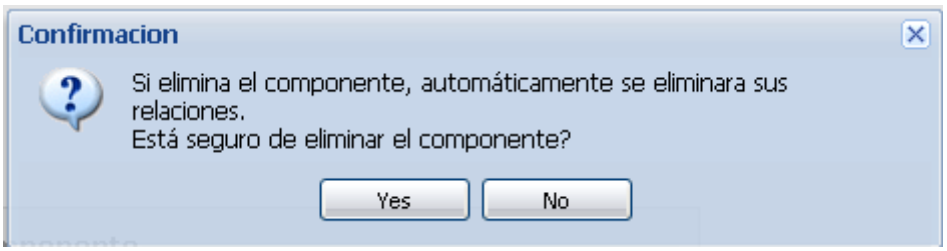
Caso de Uso:	**Gestionar Componentes**
Actores:	**Ingeniero en Componentes**
Resumen:	**El CU se inicia cuando el Ingeniero en Componentes decide Insertar, Modificar o Eliminar un componente en un paquete determinado.**
Precondiciones:	**Existencia real del componente en SISalud.** **Existencia del Paquete en el sistema.** **Actor que inicializa autenticado.**
Referencias	**R3, R3.1, R3.2,R3.3, R3.4**
Prioridad	
Flujo Normal de Eventos	
1. El usuario decide, Gestionar Componente.	**2. El sistema presenta un menú Paquete donde se escogerá el paquete a insertar un nuevo componente desplegándose el menú del paquete para realizar la acción, en el caso de modificar y eliminar se gestionara en la tabla de los componentes, al igual que cuando decida filtrar que tendrá un sección de filtrado.**
3. El usuario escoge una de las siguientes opciones: **a) Insertar Componente.** **b) Modificar Componente.** **c) Eliminar Componente.** **d) Filtrar Componente**	
Sección "Insertar Componente"	

Acción del Actor	Respuesta del Sistema
1. El usuario selecciona el paquete en el Menú Paquete y selecciona el paquete al cual va a insertarle el componente y presiona en Insertar Componente.	2. Muestra el formulario correspondiente.
3. El usuario inserta en el formulario los datos (nombre, negocio) y presiona el botón Insertar.	4. El sistema valida los datos de entrada.
	5. El sistema verifica que el Componente no exista ya en ese Paquete.
	6. El sistema muestra un mensaje "Se ha insertado el componente correctamente"
7. El usuario presiona Cerrar y termina la acción.	



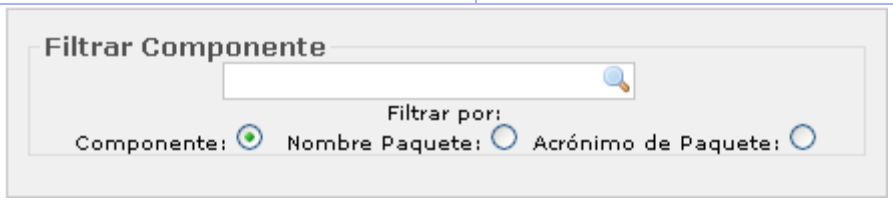
Flujos Alternos Sección "Insertar Componente"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si hay errores en la validación de los datos muestra el mensaje "Llene los campos correctamente", y pasa a la acción #2 de la sección Insertar Componente.
	5.1 Si ya existe el componente el sistema muestra el mensaje "Ya existe el Componente en ese Paquete" y regresa a la acción #2 de la sección Insertar Componente.

Sección "Modificar Componente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita Modificar Componente.	**2. Muestra un formulario de búsqueda.**
3. El usuario presiona "Modificar" en el componente a modificar.	**4. El sistema muestra un formulario lleno con los datos actuales del componente seleccionado.**
5. El usuario modifica los datos (nombre, negocio) y selecciona la opción modificar.	**6. El sistema valida los datos de entrada.**
	7. El sistema muestra un mensaje de "Se ha actualizado el componente correctamente".
8. El usuario presiona Cerrar y termina la acción.	
	
Flujos Alternos Sección "Modificar Componente"	
Acción del Actor	**Respuesta del Sistema**
	6.1 Si hay errores en la validación de los datos muestra el mensaje "Llene los campos correctamente", y pasa a la acción #2 de la sección Modificar componente.
Sección "Eliminar Componente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita Eliminar Componente.	**2. El sistema muestra un mensaje de confirmación.**
3. El usuario selecciona el botón "Yes"	**4. El sistema elimina el componente del sistema y muestra un mensaje "Se ha eliminado el paquete correctamente".**



Flujos Alternos Sección "Eliminar Componente"	
Acción del Actor	**Respuesta del Sistema**
1.1 El usuario decide no eliminar el componente.	
Poscondiciones	

Sección "Filtrar Componente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide Filtrar Componente.	**2. El sistema muestra un campo de texto donde se llena con el nombre del componente que se desea filtrar.**
3. El usuario selecciona como quiere filtrar el componente presiona el botón "Filtrar" del componente que decida filtrar.	**4. El sistema muestra los componentes filtrados.**



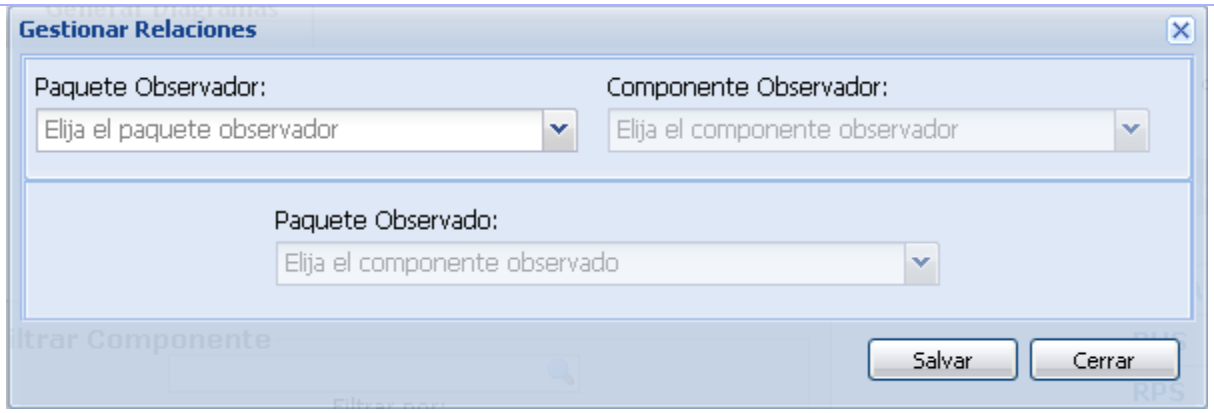
Flujos Alternos Sección "Filtrar Componente"	
Acción del Actor	**Respuesta del Sistema**
1.1 El usuario decide no filtrar componentes.	
Poscondiciones	

Tabla 3: Descripción Textual de los "Casos de Uso Gestionar Componentes".

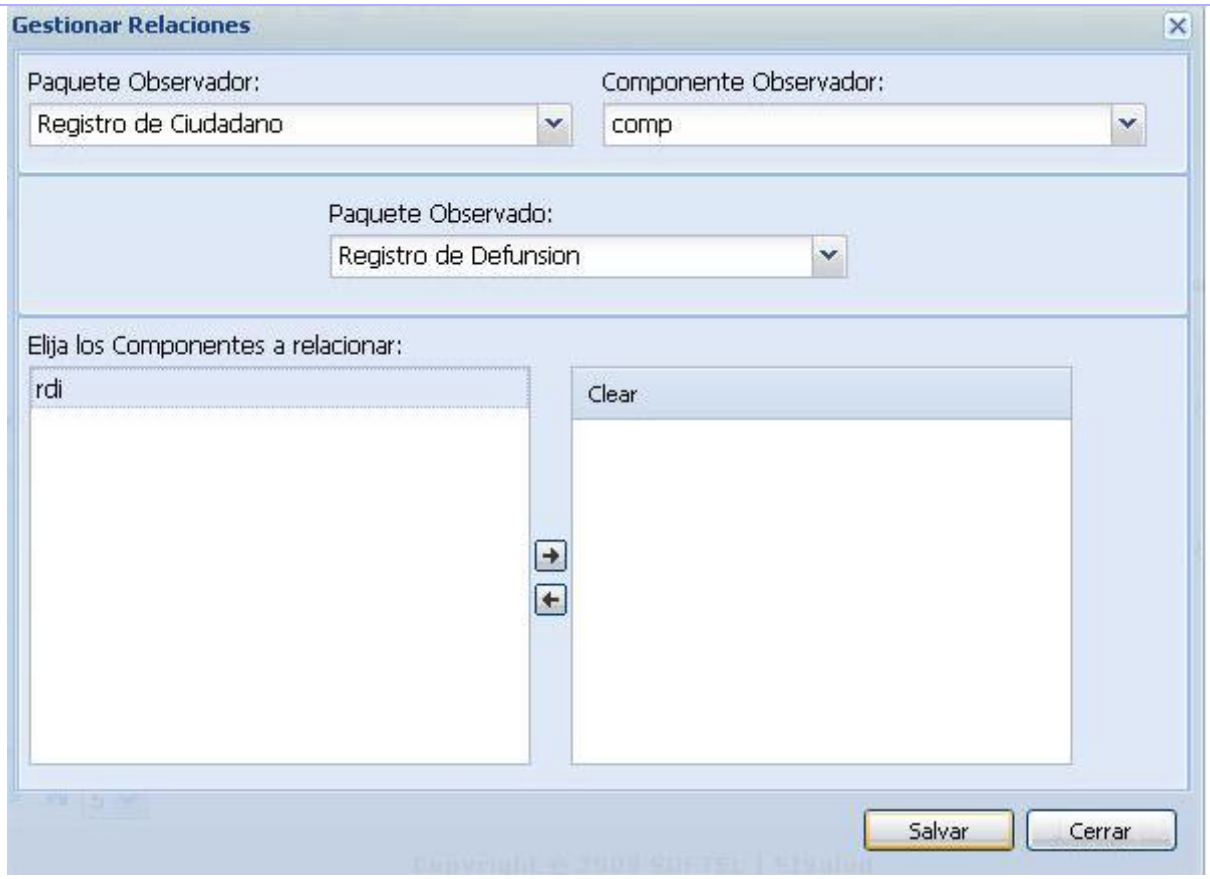
2.4.3.3 Descripción Textual de los "Casos de Uso Gestionar Relaciones".

Caso de Uso:	**Gestionar Relaciones**
Actores:	**Ingeniero en Componente**
Resumen:	**El CU se inicia cuando el Ingeniero en Componente decide Insertar, Modificar o Eliminar una relación entre Paquetes de Componentes del sistema.**
Precondiciones:	**Existencia real de los paquetes relacionados así como de los componentes que los componen en SISalud.** **El actor debe estar autenticado.**
Referencias	**R2, R2.1, R2.2, R2.3**
Prioridad	

Flujo Normal de Eventos	
1. El usuario selecciona, Gestionar Relaciones.	**2. El sistema muestra un formulario**
3. El usuario decide una de las siguientes opciones: **e) Insertar Relación.** **f) Modificar Relación.** **g) Eliminar Relación.**	
Sección "Insertar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide Insertar Relaciones entre Paquetes.	**2. Muestra el formulario correspondiente, con los datos de los paquetes y los componentes.**
3. El usuario selecciona el paquete observador, el componente observador y el paquete observado.	**4. El sistema brinda la opción de Salvar y Cerrar.**
5. El usuario presiona el botón "Salvar"	**6. El sistema muestra el mensaje "Se ha gestionado la relación Correctamente"**
7. El usuario presiona el botón Cerrar para salir de la acción.	

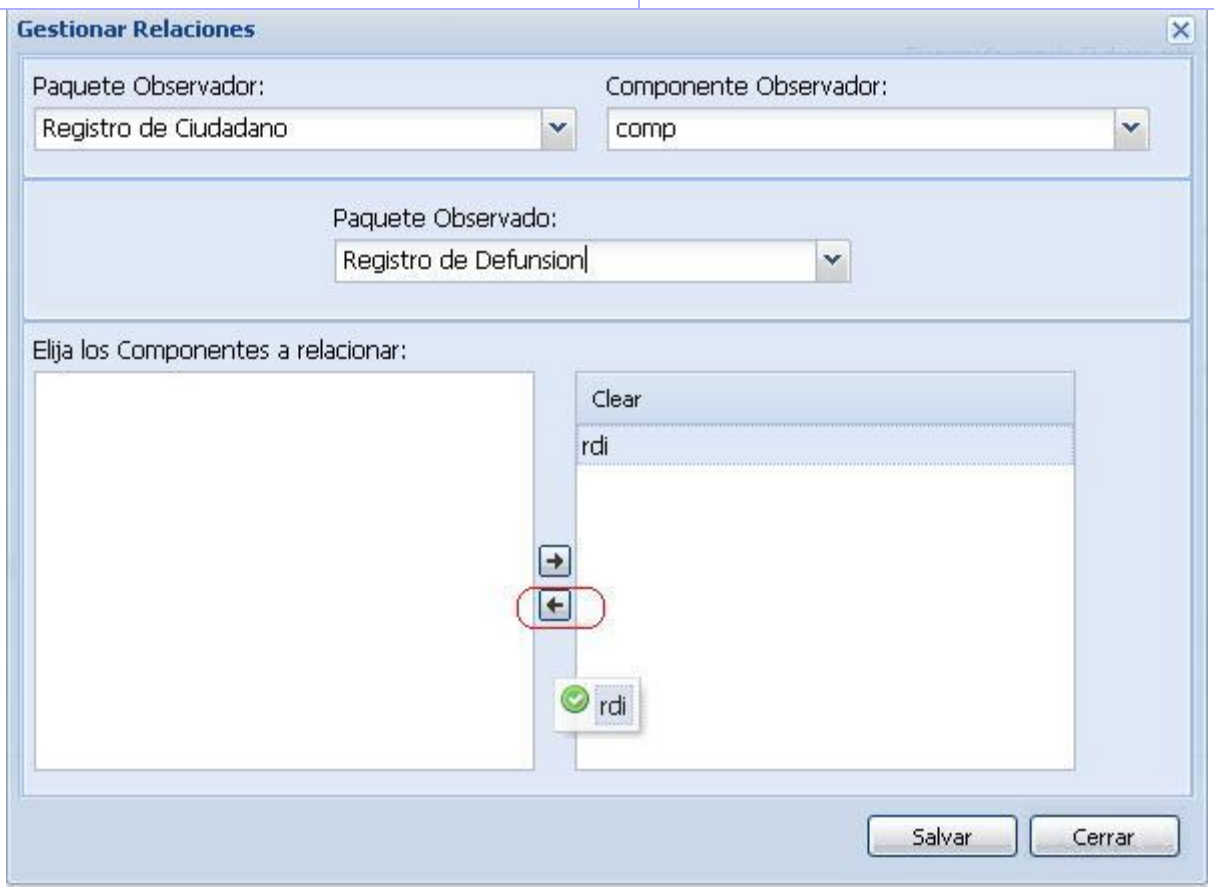


Flujos Alternos Sección "Insertar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
1.1. El usuario no decide establecer relaciones	
Sección "Modificar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide Modificar Relación entre Paquetes.	**2. El sistema muestra un formulario con todos los paquetes relacionados y sus compontes además de las opciones:** **a) Salvar** **b) Cancelar**
3. El usuario selecciona los paquetes en los que se decide modificar sus relaciones.	**4. El sistema muestra un formulario con las relaciones asociadas al paquete seleccionado.**
5. El usuario modifica las relaciones y presiona el botón Salvar.	**6. El sistema muestra un mensaje de "Se ha gestionado la relación correctamente".**
7. El usuario presiona cerrar para terminar la acción.	



Flujos Alternos Sección "Modificar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
1.1 El usuario presiona en Cerrar para terminar la acción.	
Sección "Eliminar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide Eliminar Relaciones entre Paquetes.	**2. El sistema muestra un formulario con todos los paquetes relacionados y sus compontes además de las opciones:** **a) Salvar** **b) Cancelar**
3. El usuario selecciona el paquete al cual desea eliminarle la relación.	**4. El sistema muestra un listado con las relaciones existentes.**
5. El usuario selecciona el componente y lo	**6. El sistema permite eliminar otra**

arrastra hasta donde están los componentes sin relacionar.	**Relación.**
7. El usuario presiona en Cerrar para terminar la acción.	



Flujos Alternos Sección "Eliminar Relaciones entre Paquetes"	
Acción del Actor	**Respuesta del Sistema**
3.1 El usuario decide no eliminar la Relación.	
Poscondiciones	

Tabla 4: Descripción Textual de los "Casos de Uso Gestionar Relaciones".

2.4.3.4 Descripción Textual de los "Casos de Uso Gestionar Usuarios".

Caso de Uso:	**Gestionar Usuarios**
Actores:	**Administrador**
Resumen:	**El CU se inicia cuando el Administrador decide Insertar, Modificar, Eliminar o Filtrar un usuario en el sistema.**

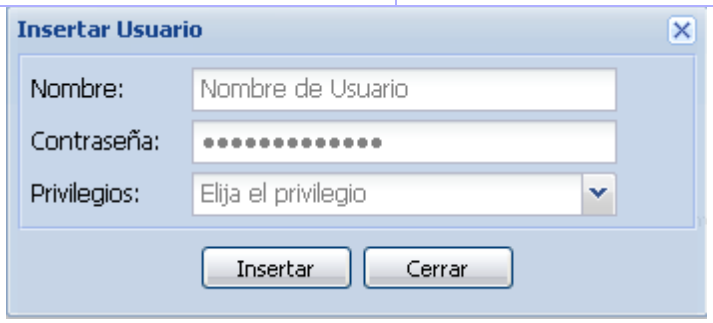
Precondiciones:	**Administrador debe estar autenticado.**
Referencias	**R5, R5.1, R5.2,R5.3, R5.4**
Prioridad	

Flujo Normal de Eventos

1. El usuario escoge una de las siguientes opciones: a) Insertar Usuario. b) Modificar Usuario. c) Eliminar Usuario. d) Filtrar Usuario	

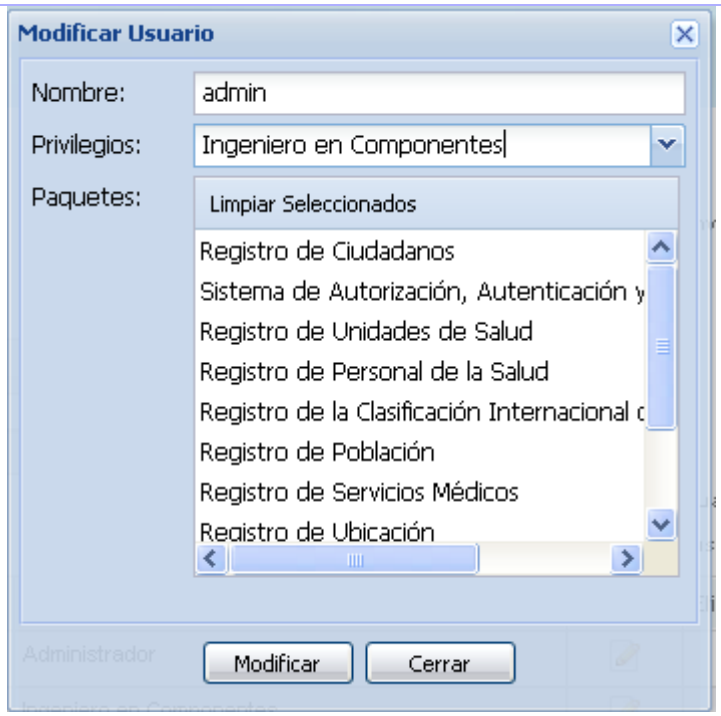
Sección "Insertar Usuario"

Acción del Actor	Respuesta del Sistema
1. El Administrador solicita Insertar Nuevo Usuario.	2. Muestra el formulario correspondiente.
3. El Administrador llena el formulario con los datos (nombre de usuario, contraseña, privilegios) y presiona el botón Insertar.	4. El sistema valida los datos de entrada.
	5. El sistema verifica que el Usuario no exista ya en el sistema.
	6. El sistema muestra un mensaje "Se ha insertado el usuario correctamente".
7. El usuario presiona en Cerrar para salir de la acción.	



Flujos Alternos Sección "Insertar Usuario"

Acción del Actor	Respuesta del Sistema
	4.1 Si hay errores en la validación de los datos muestra el mensaje "Error en los datos de entrada", y pasa a la acción #2 de la sección Insertar Usuario.
	5.1 Si ya existe el usuario el sistema muestra el mensaje "Ya existe el Usuario" y regresa a la acción #2 de la sección Insertar Usuario.
Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador solicita Modificar Usuario.	**2. Muestra el formulario correspondiente.**
3. El Administrador modifica los datos del formulario (nombre y privilegios) y presiona el botón Modificar.	**4. El sistema valida los datos de entrada.**
	5. El sistema muestra un mensaje de "Se ha modificado el usuario correctamente".
6. El usuario presiona en Cerrar para terminar la acción.	



Flujos Alternos Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si hay errores en la validación de los datos muestra el mensaje "Error en los datos de entrada", y pasa a la acción #2 de la sección Modificar Usuario.
Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador solicita Eliminar Usuario.	**2. El sistema elimina el usuario.**
	3. El sistema muestra un mensaje "Se ha eliminado el usuario correctamente"

Nombre Usuario	Privilegios	Modificar	Eliminar
admin	Administrador		
usuario	Ingeniero en Componentes		
teby	Administrador		
Esteban	Ingeniero en Componentes		
Nombre 1/1 5	Privilegios	Modificar	Eliminar

Flujos Alternos Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
4.1 El Administrador decide no eliminar otro Método.	
Poscondiciones	
Sección "Filtrar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide filtrar usuario.	**2. El sistema muestra un campo de texto donde se llene con el nombre del usuario que se desea filtrar.**
3. El usuario presiona el botón "Filtrar" del usuario que decida filtrar.	**4. El sistema muestra el usuario filtrado.**

Flujos Alternos Sección "Filtrar Usuario"

Acción del Actor	Respuesta del Sistema
1.1 El usuario decide no filtrar el usuario	
Poscondiciones	

Tabla 5: Descripción Textual de los "Casos de Uso Gestionar Usuarios".

2.4.3.5 Descripción Textual de los "Casos de Uso Generar Diagramas".

Caso de Uso:	Generar Diagramas
Actores:	Ingeniero en Componente
Resumen:	El CU se inicia cuando el Ingeniero en Componente decide generar los diagramas de componentes.
Precondiciones:	Administrador y/o Desarrollador deben estar autenticados.
Referencias	R4, R4.1, R4.2
Prioridad	

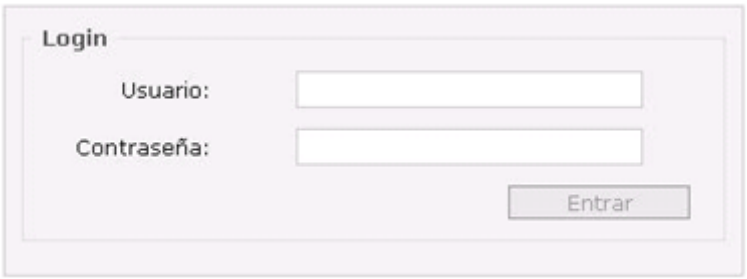
Flujo Normal de Eventos	
1. El Administrador selecciona, Generar Diagramas.	2. El sistema muestra el menú con las opciones
3. El usuario escoge: Diagrama Global de Componentes.	4. El sistema muestra el Diagrama Global de Componentes.
Sección "Matriz de Impacto"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona, Generar Diagramas.	2. El sistema muestra el menú con las opciones
3. El usuario escoge: Matriz de Impacto.	4. El sistema muestra opción deseada.

Tabla 6: Descripción Textual de los "Casos de Uso Generar Diagramas".

2.4.3.6 Descripción Textual de los "Casos de Uso Autenticar Usuarios".

Caso de Uso:	**Autenticar Usuarios**
Actores:	**Usuario del sistema**
Resumen:	**El CU se inicia cuando el usuario desea entrar al sistema.**
Precondiciones:	
Referencias	**R6, R6.1**
Prioridad	

Flujo Normal de Eventos	
	1. El sistema muestra la pantalla de autenticación.
2. El usuario entra sus datos y presiona el botón Entrar.	**3. El sistema valida los datos de entrada.**
	4. El sistema comprueba si el usuario y la contraseña son correctos.
	5. El sistema visualiza la página de acuerdo a los privilegios del usuario.



Flujos Alternos Sección "Cambiar Contraseña"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona en "Cambiar Contraseña".	**2. El sistema muestra un formulario donde deberá llenar los campos de la contraseña y la nueva contraseña que desea cambiar.**
3. El sistema llena los campos del formulario	**4. El sistema comprueba los datos de entrada**
5. El usuario presiona el botón "Cambiar"	**6. El sistema muestra un mensaje "Se**

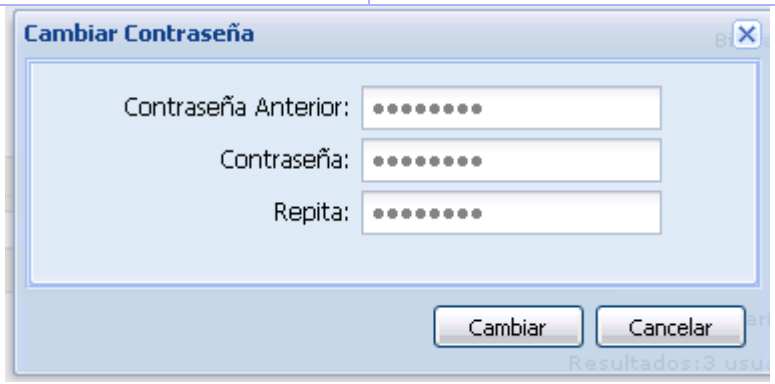
	ha cambiado la contraseña correctamente"
	
Poscondiciones	**La contraseña del usuario queda cambiada**
Flujos Alternos Sección "Cambiar Contraseña"	
Acción del Actor	**Respuesta del Sistema**
1.1 El usuario decide no cambiar la contraseña	**2.1 El sistema va a la acción #2 de la sección Eliminar componente.**
Poscondiciones	

Tabla 7: Descripción Textual de los "Casos de Uso Autenticar Usuarios".

Conclusiones del Capítulo.

En este capítulo se definieron las principales funcionalidades que debe tener el sistema a desarrollar, estructurándose en casos de uso. Se elaboró el diagrama de casos de uso del sistema y se describieron textualmente cada uno de los casos de uso identificados.

Todo lo anterior, provee de una visión general de lo que el sistema debe hacer, por lo que se está en condiciones de pasar al siguiente capítulo, que expondrá cómo se realizarán las operaciones que aquí se describieron.

Capítulo 3: Diseño del Sistema.

En este capítulo se modela y adquiere forma el sistema para que soporte todos los requisitos funcionales o no funcionales e inclusive cualquier otro tipo de restricción, contribuyendo a obtener una arquitectura sólida y estable para la futura implementación del sistema de software. Entre los artefactos que serán mostrados en el presente capítulo se encuentran: el Modelo de Diseño, Diagrama de Clases y Descripción de las Clases de Diseño.

3.1 Estilo de Arquitectura

Un estilo de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. El estilo de arquitectura seleccionado es dentro de los de llamada y retorno, el Modelo Vista Controlador (MVC). Este estilo o patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario. Despliega la información contenida en el modelo.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

3.2 Patrones de Casos de Uso.

Para el desarrollo de esta investigación se usarán patrones de casos de usos, los cuales son utilizados como herramientas o técnicas, obtenidas de la experiencia que poseen los desarrolladores de software, facilitando la solución a los problemas que se presentan en la modelación de sistemas, lo que hace que se puedan obtener modelos con mayor calidad y de forma más rápida.

Los patrones de casos de usos a utilizar son:

- ✓ **Patrón CRUD (Creating, Reading, Updating and Deleting):** Propone identificar un CU, llamado “Información CRUD” o “Administrar Información”, que modela todas las operaciones que se pueden realizar sobre una parte de la información de cierto tipo (o sea en una misma entidad), tal como crearla, leerla, actualizarla y eliminarla. Dicho patrón se pone de manifiesto en los requisitos funcionales de Gestionar Usuarios, Gestionar Paquetes de Componentes, Gestionar Componentes y Relaciones.

✓ **Múltiples actores**: Consiste en que dos o más actores juegan el mismo papel hacia un caso de uso determinado. Este rol es representado por otro actor, heredado por los actores que comparten este rol. En este caso este patrón de caso de uso se pone de manifiesto con el Administrador de Sistema y El Ingeniero de Componentes los cuales cumplen funciones diferentes en nuestra aplicación pero en un momento determinado realizan una misma función la cual es a la hora de autenticarse para la entrada al sistema.

3.3 Patrones de Diseño

Los patrones de diseño contribuyen un diseño más sólido del sistema y a lograr un bajo acoplamiento, característica vital que deben de tener los sistemas distribuidos.

La arquitectura MVC implementa internamente los patrones de asignación de responsabilidades GRASP.

”Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.”

Entre ellos se utilizaron:

✓ **Bajo acoplamiento:** Se basa en la idea de tener las clases lo menos ligadas entre sí posible. De forma tal que en caso de producirse una modificación en alguna de ellas, la repercusión en el resto de clases sea mínima, potenciando así la reutilización, y disminuyendo la dependencia entre estas.Este es usado para desacoplar las vistas de los modelos y estos de la forma en que se muestran e ingresan los datos, además las funcionalidades estan encapsuladas en las clases que heredan de Command, de manera que es muy sencillo adicionar, eliminar o modificar dichas funcionalidades.

✓ **Experto:** Es el patrón más usado pues el asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la “intuición” de que los objetos hacen cosas relacionadas con la información que poseen. Experto es empleado en la clase Helper que es la responsable de crear el comando apropiado o la cadena de filtros necesaria, en dependencia de la peticion que se hace a la clase controladora.

✓ **Controlador:** Sirve de intermediario entre una interfaz y el algoritmo que la implementa, siendo el que recibe los datos del usuario y los envía a las distintas clases según el método llamado. (20)

Se pone de manifiesto en la clase MainController que recibe las peticiones de las diferentes vistas y delega en las clases que encapsulan las diferentes funcionalidades la operación que se solicita.

Los patrones del grupo GoF se clasifican en: creacionales, estructurales y de comportamiento se hará uso en nuestro trabajo de:

✓ **Intercepting Filter:** Recibe diferentes tipos de solicitudes, que requieren diversos tipos de procesamiento. Algunas solicitudes son transmitidas a las manejadoras, mientras que las demás solicitudes deben ser modificadas, controladas, antes de ser procesada.
Es aplicada en las clases que heredan de Filter además de la clase Filter Chain que en conjunto verifican que la peticion que se hace de las vistas sea la correcta.

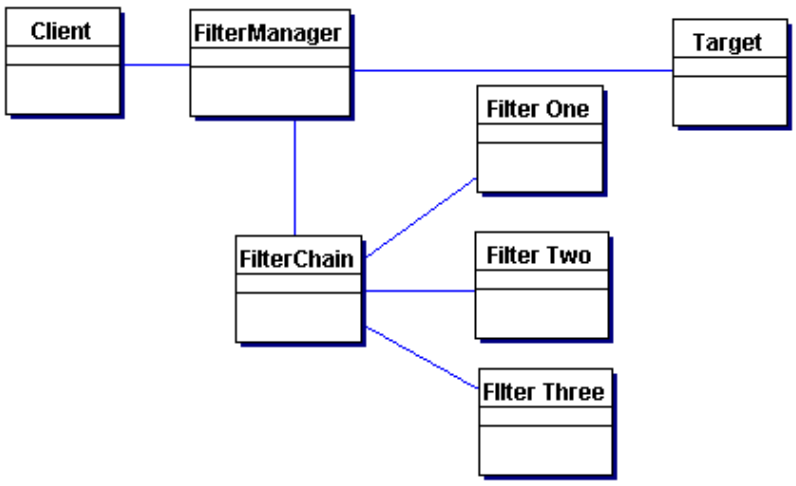


Figura 4: Patrón Intercepting Filter.

✓ **Command (Acción):** Es el que permite que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación. Este patrón GoF es aplicado en cada clase command la cual es una funcionalidad especifica del sistema.

✓ **Front Controller:** Es la encargada de controlar y coordinar la tramitación de cada usuario a través de múltiples peticiones. Tales mecanismos de control pueden ser gestionados tanto en forma centralizada o descentralizada. El patrón Front Controller en la clase MainController, es el unico punto al cual las vistas enviaran las peticiones y este se encargara en conjunto de otras clases de procesarlas.

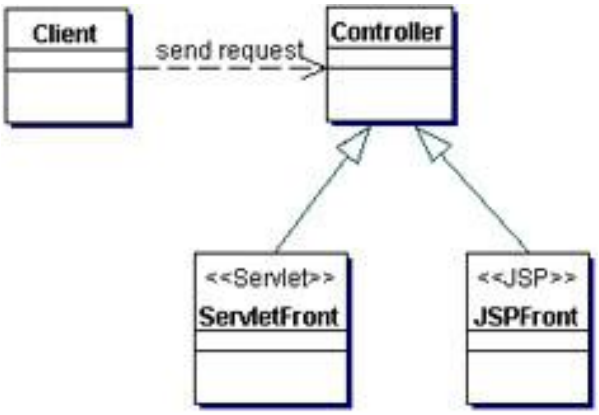


Figura 5: Patrón Front Controller.

✓ **Data Access Object (DAO):** El acceso a los datos varía dependiendo de la fuente de los datos. El acceso a la persistencia de almacenamiento, como a una base de datos varía mucho, dependiendo del tipo de almacenamiento (bases de datos relacionales, bases de datos orientadas a objetos, archivos planos) y el proveedor de la aplicación. Se aplica en cada unas de las clases entidades, donde para cada una de ellas se crea una clase DAO que es la responsable de interactuar con la fuente de datos y una clase Manager que es la responsable de encapsular todas las funcionalidades correspondientes a las operaciones que se quieran hacer con esa entidad.

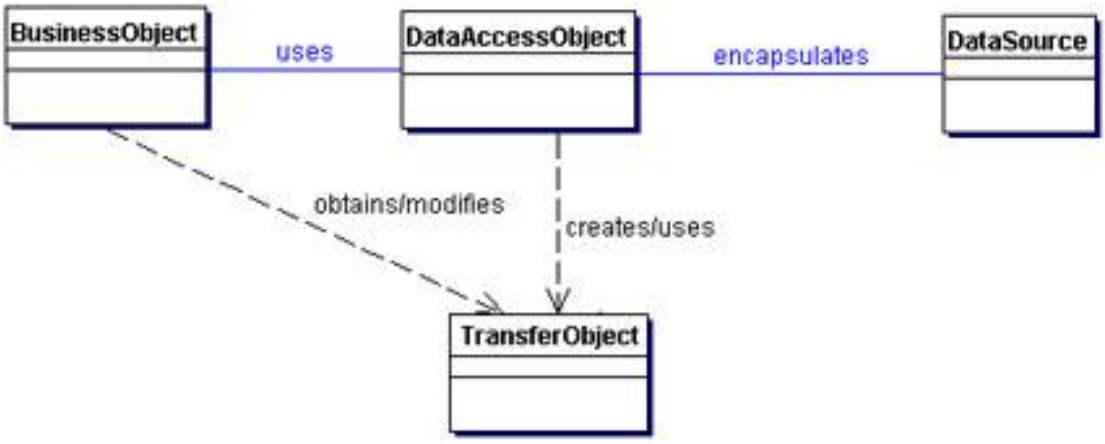


Figura 6: Patrón Data Access Object.

3.4 Diagrama de Clases

Un Diagrama de Clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases (nombre, atributos y métodos), interfaces, colaboraciones y las relaciones entre ellas.

3.4.1 Diagrama de Clases de Diseño.

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. (21) Para cada CU del sistema se realizó un diagrama de clases de diseño. Con el objetivo de lograr una mayor claridad y comprensión.

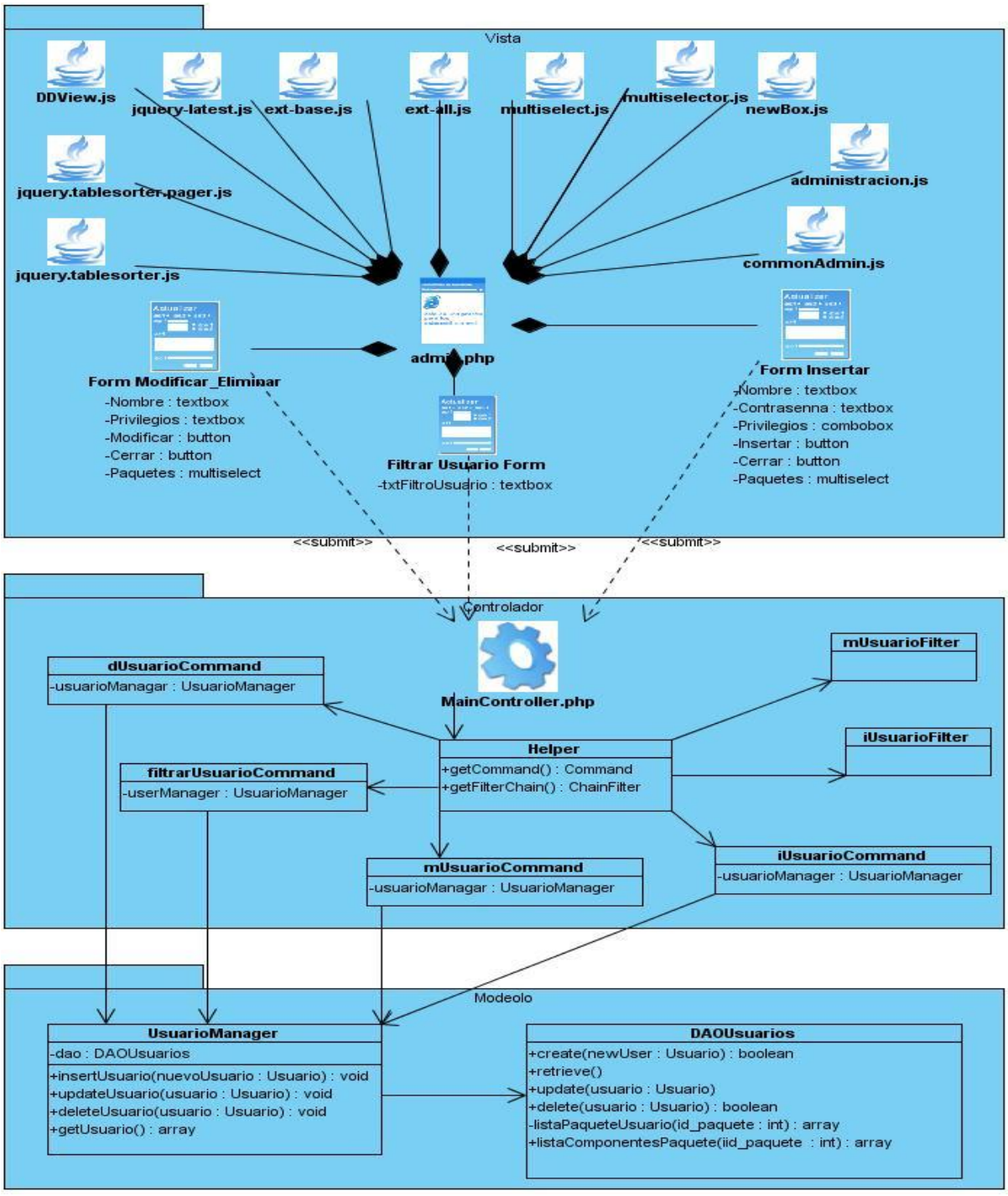


Figura 7: Diagrama de Clases de Diseño "CU Gestionar Usuario".

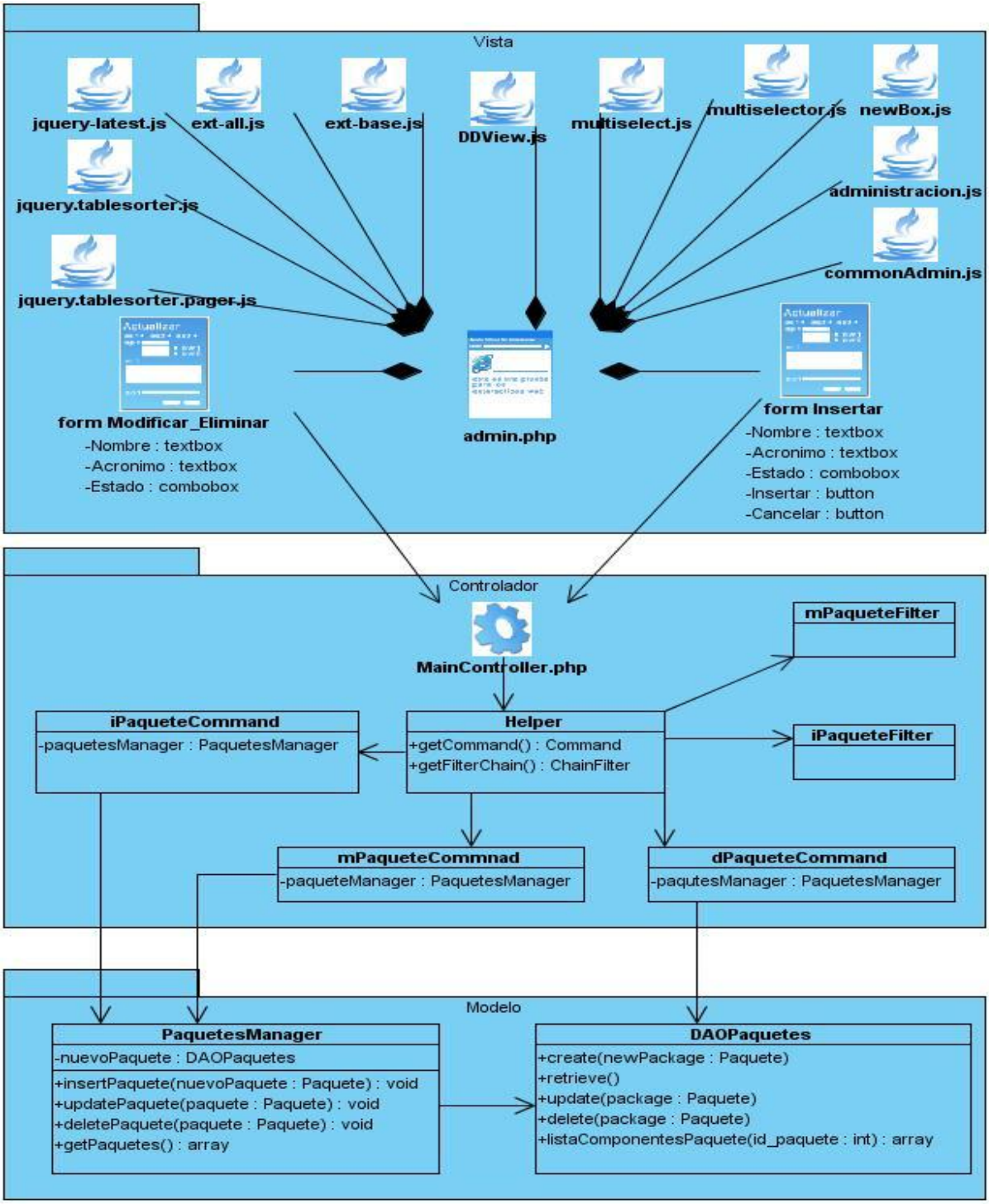


Figura 8: Diagrama de Clases de Diseño "CU Gestionar Paquetes de Componentes"

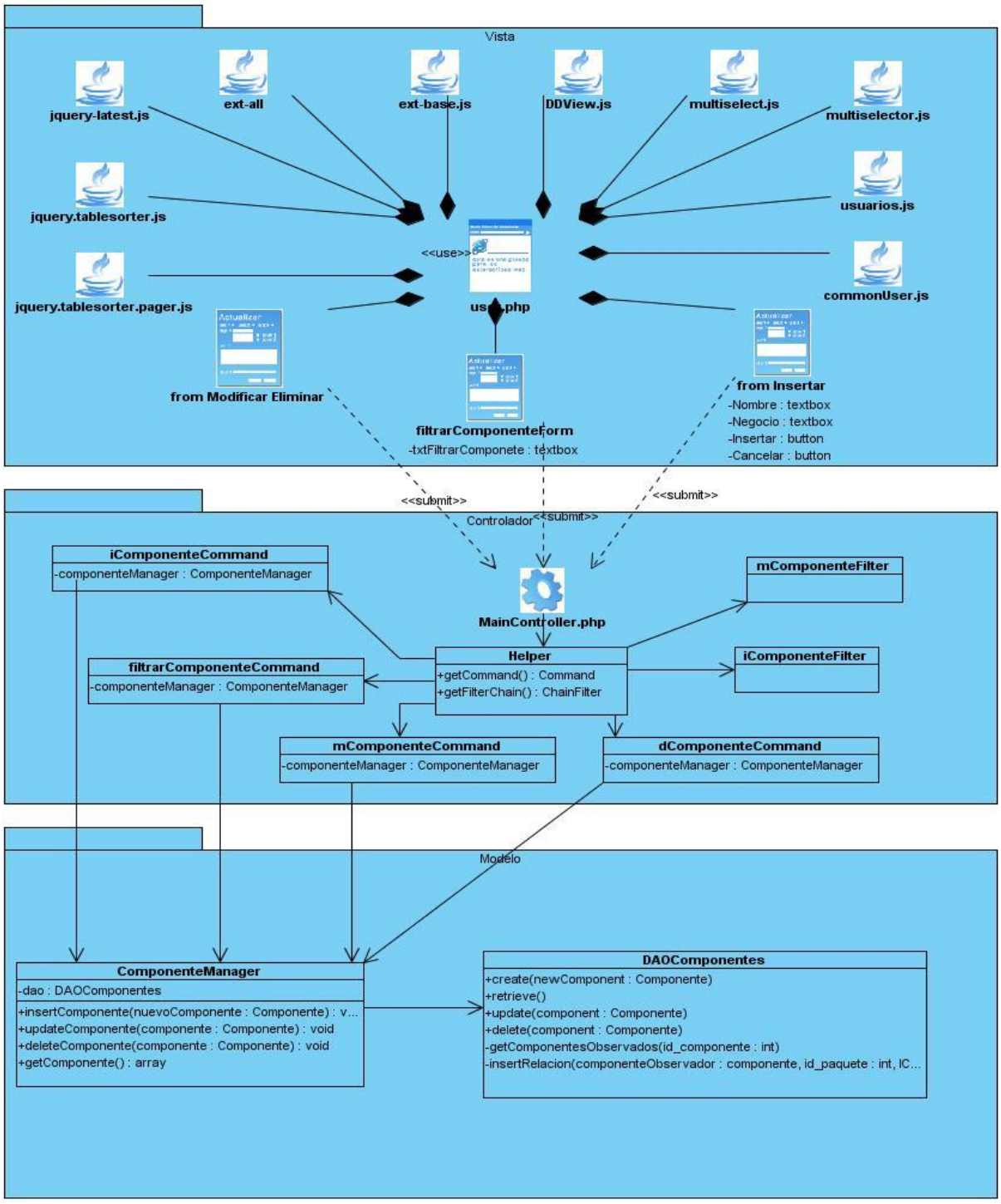


Figura 9: Diagrama de Clases de Diseño "CU Gestionar Componentes"

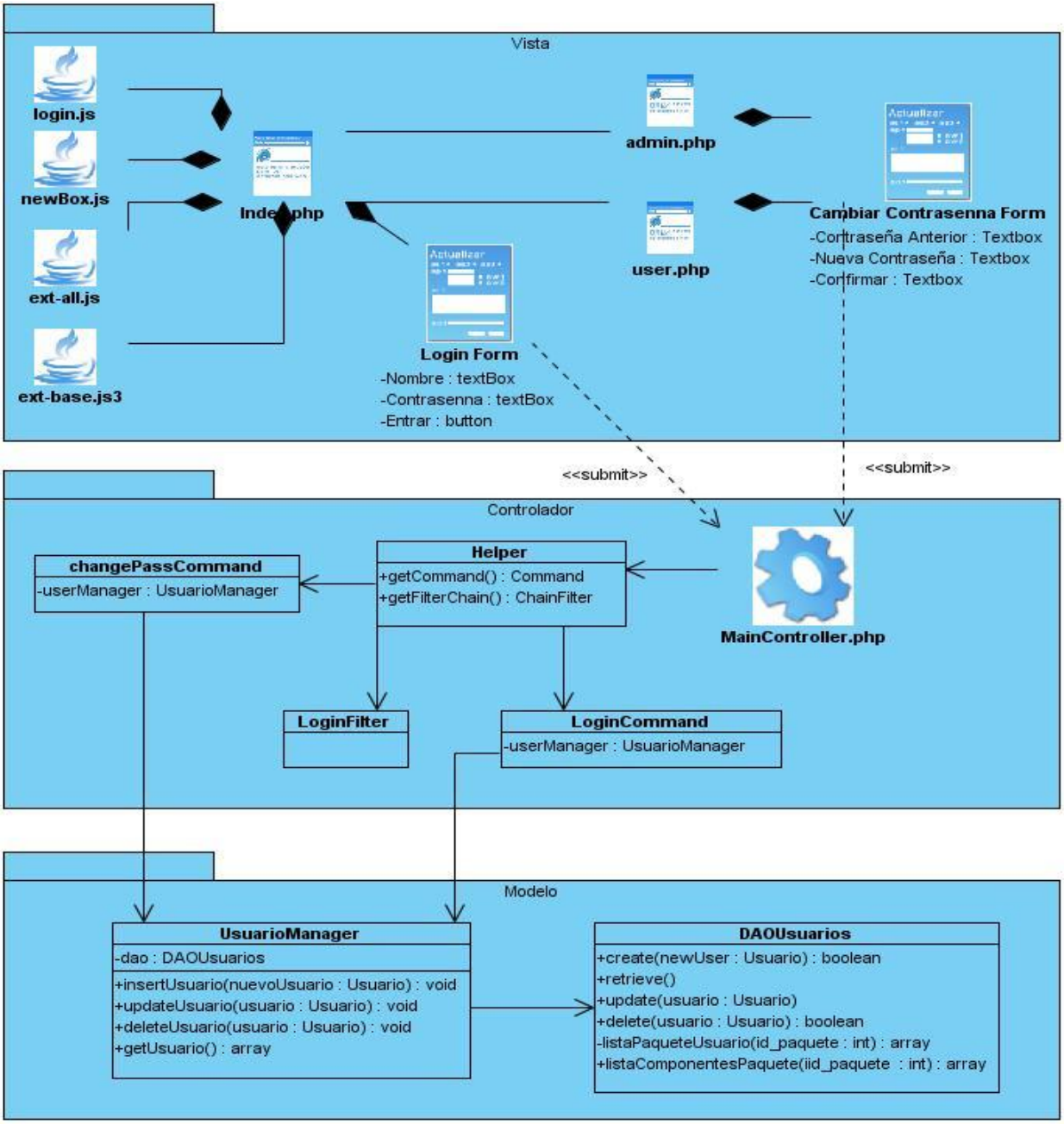


Figura 10: Diagrama de Clases de Diseño "CU Autenticar Usuarios"

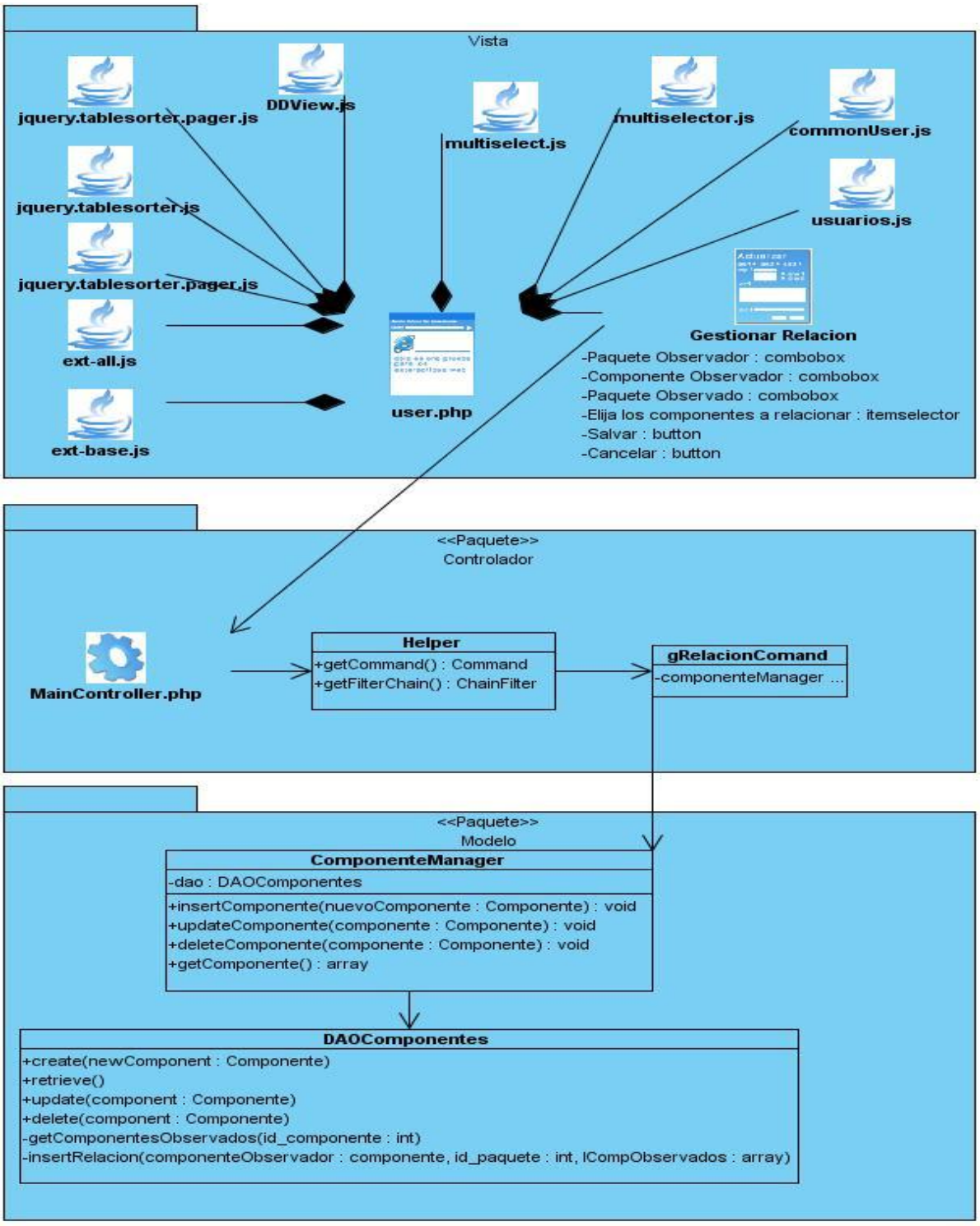


Figura 11: Diagrama de Clases de Diseño "CU Gestionar Relaciones"

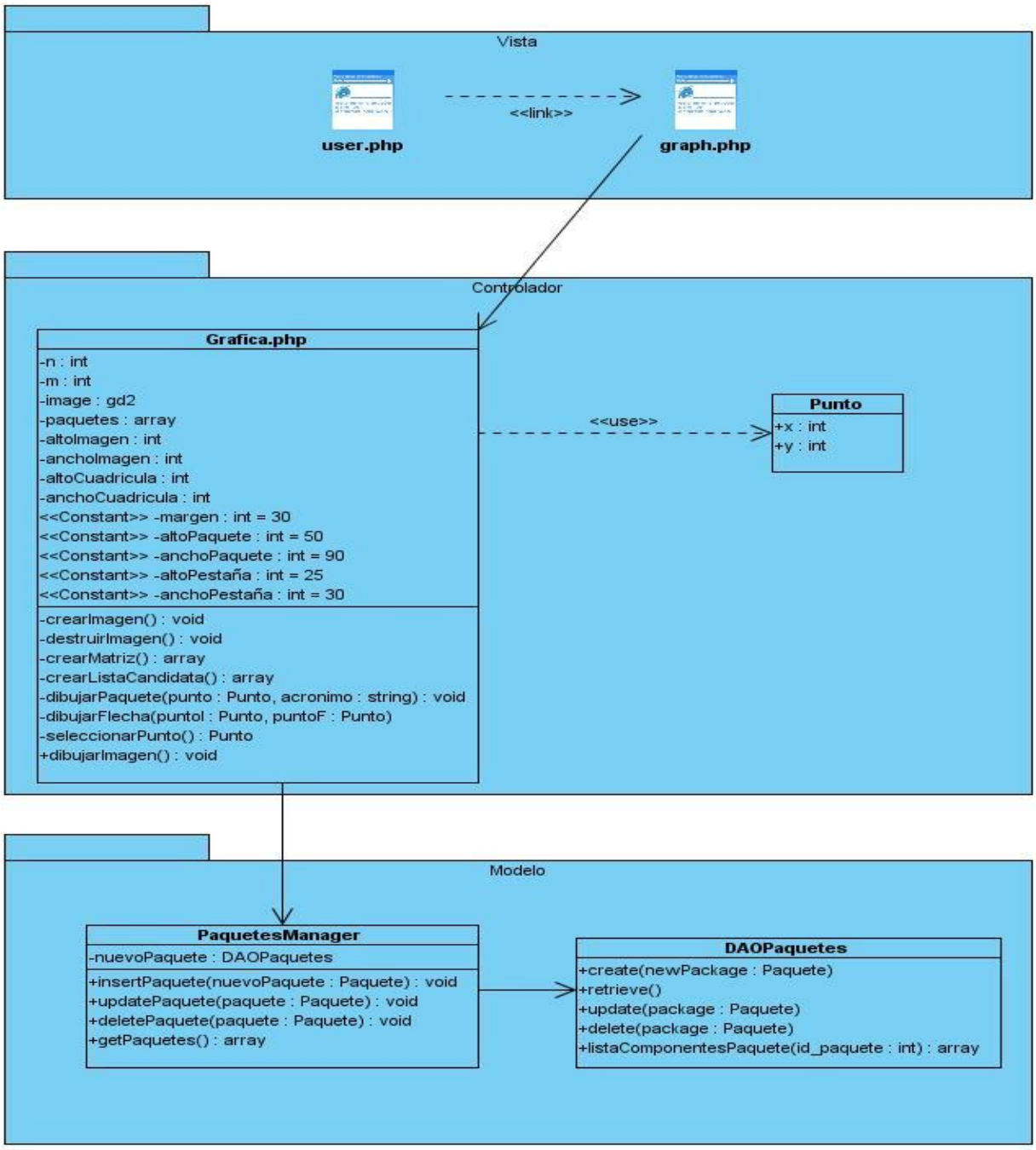


Figura 12: Diagrama de Clases de Diseño "CU Generar Diagramas"

3.5 Diagrama de Interacción.

Se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases describen la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, la funcionalidad de dichos casos de uso se recoge como un flujo de eventos utilizando para ello interacciones entre sociedades de objetos. (22)

3.5.1 Diagrama de Secuencias.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes, diferenciándose del diagrama de colaboración, en el que se destaca la organización estructural de los objetos que envían y reciben mensajes.

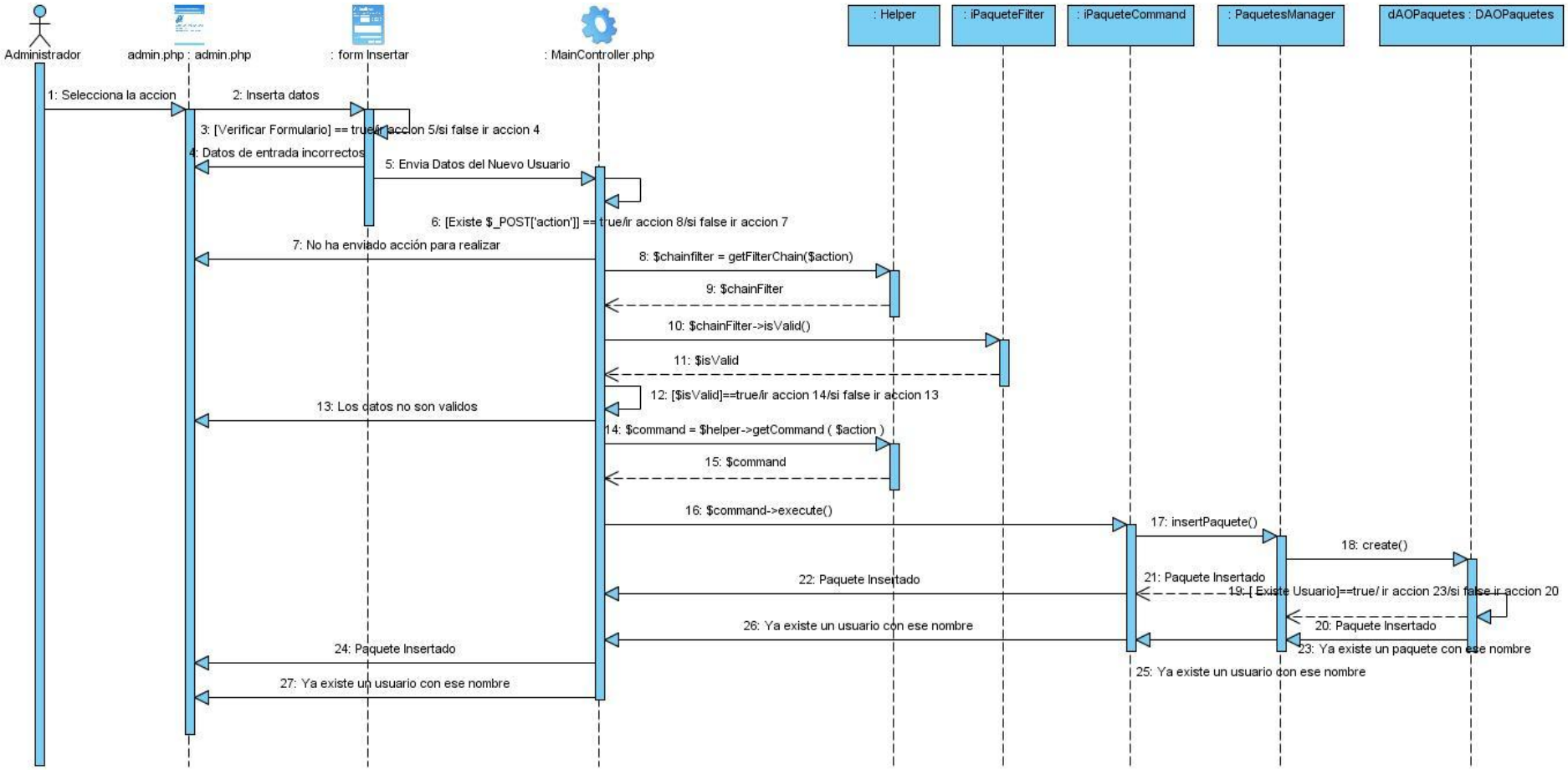


Figura 13: Diagrama de Secuencia "CU Insertar Paquetes de Componente"

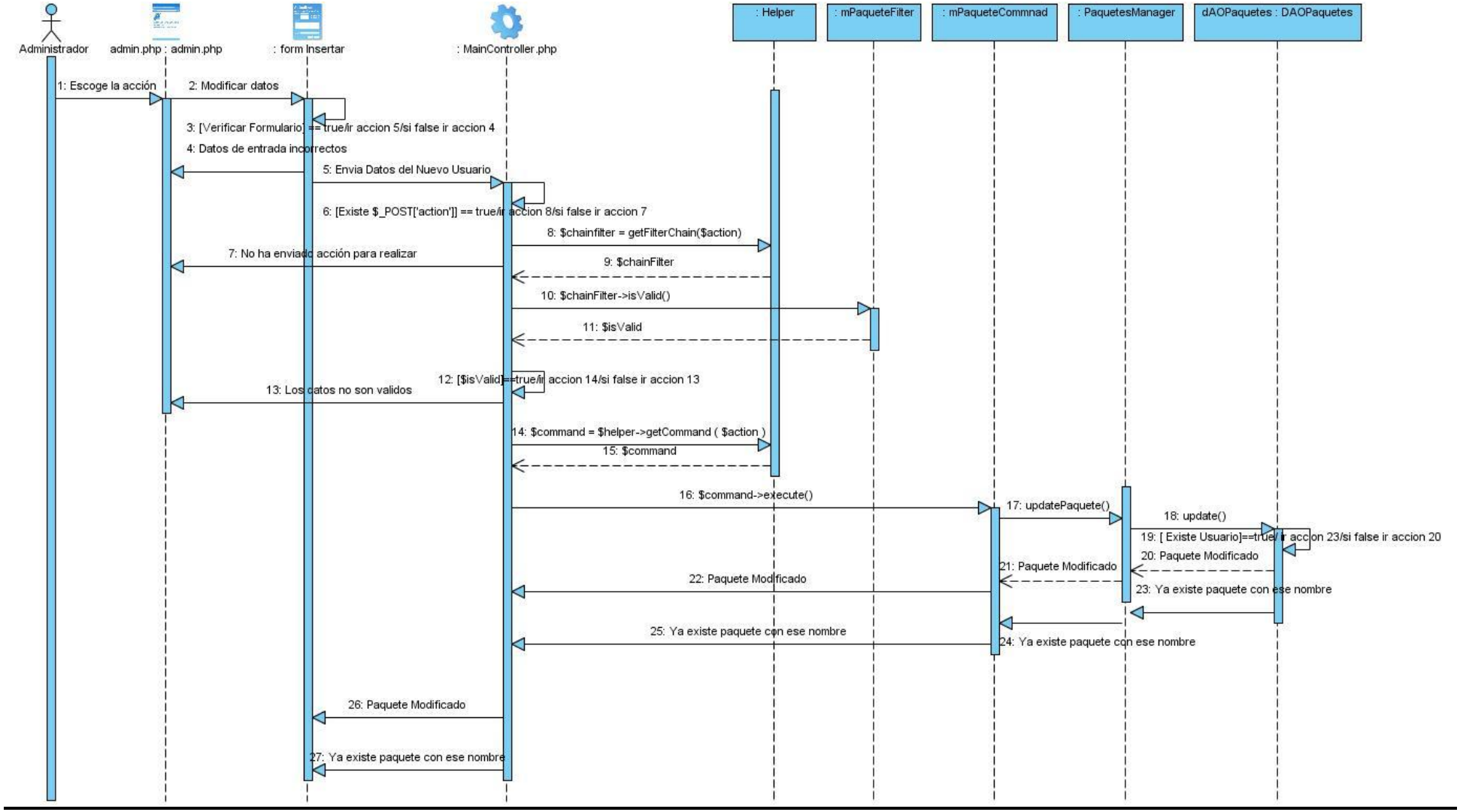


Figura 14: Diagrama de Secuencia "CU Modificar Paquetes de Componente"

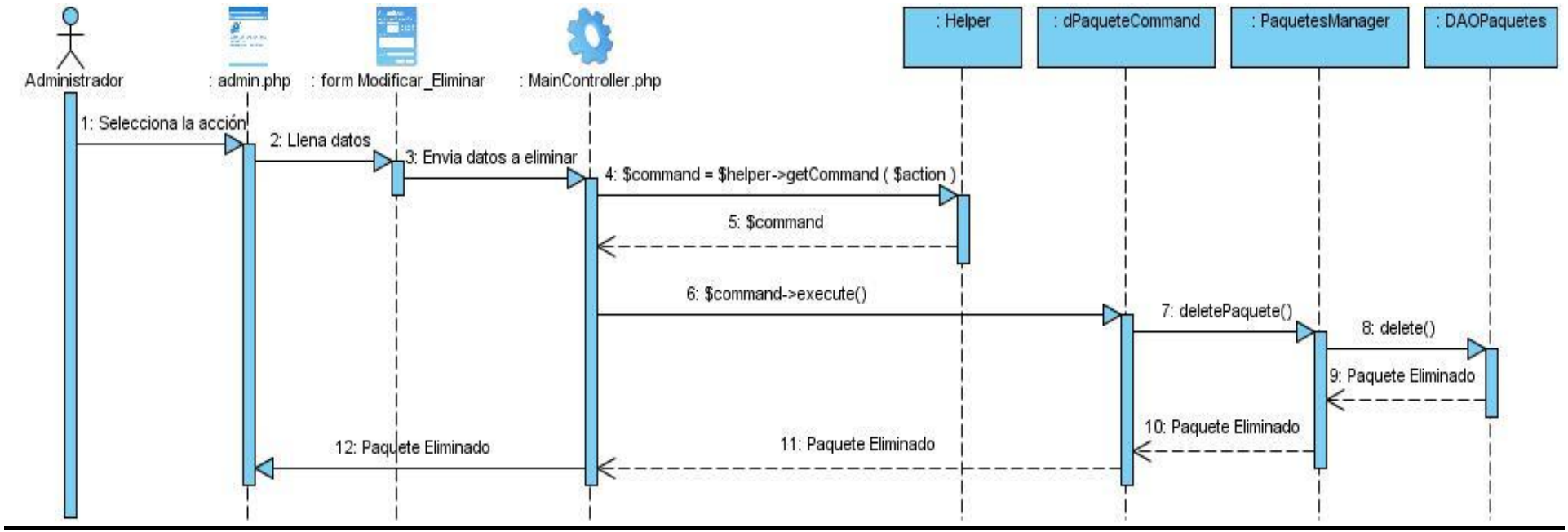


Figura 15: Diagrama de Secuencia "CU Eliminar Paquetes de Componente"

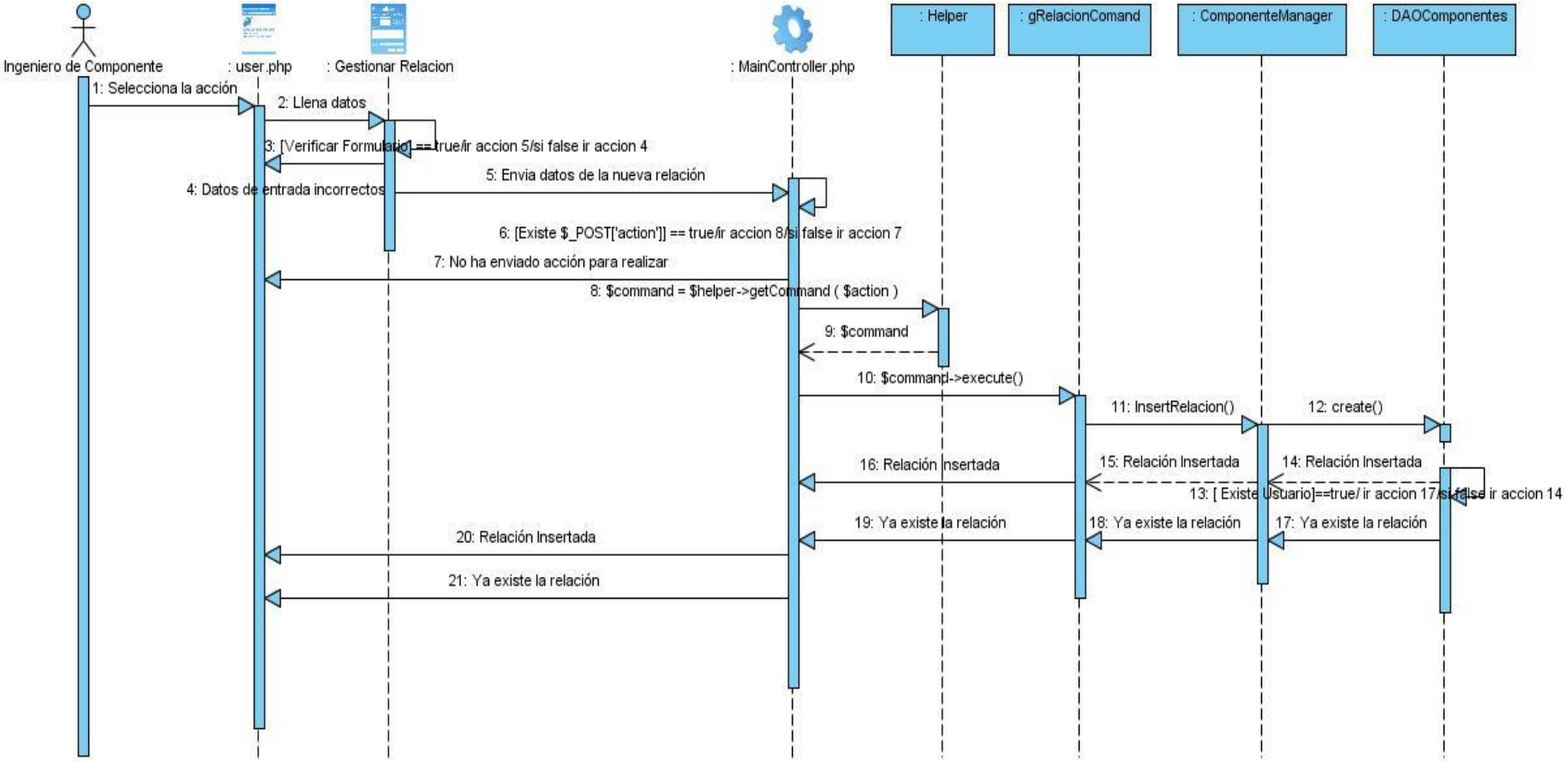


Figura 16: Diagrama de Secuencia "CU Insertar Relación"

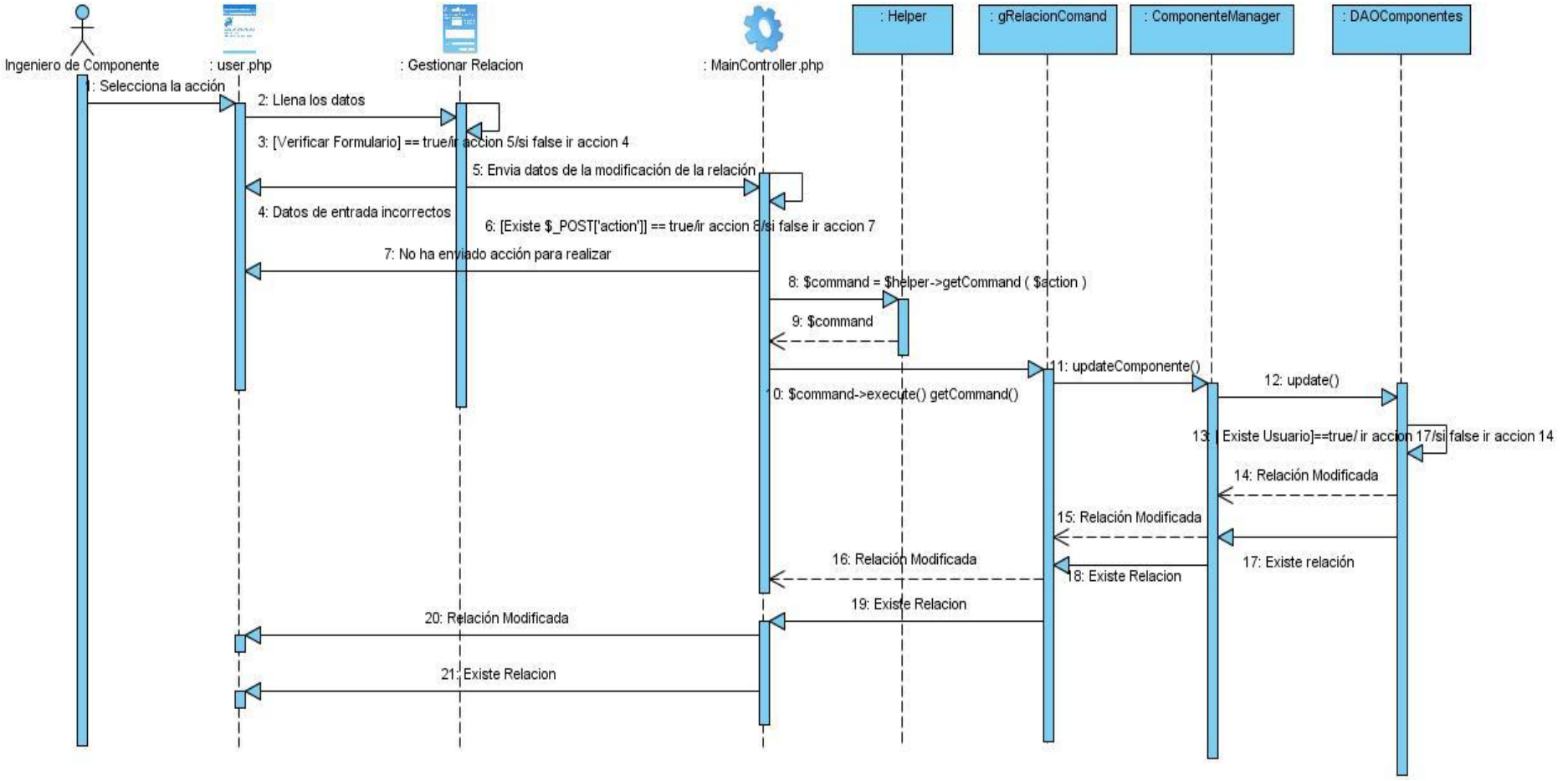


Figura 17: Diagrama de Secuencia "CU Modificar Relación"

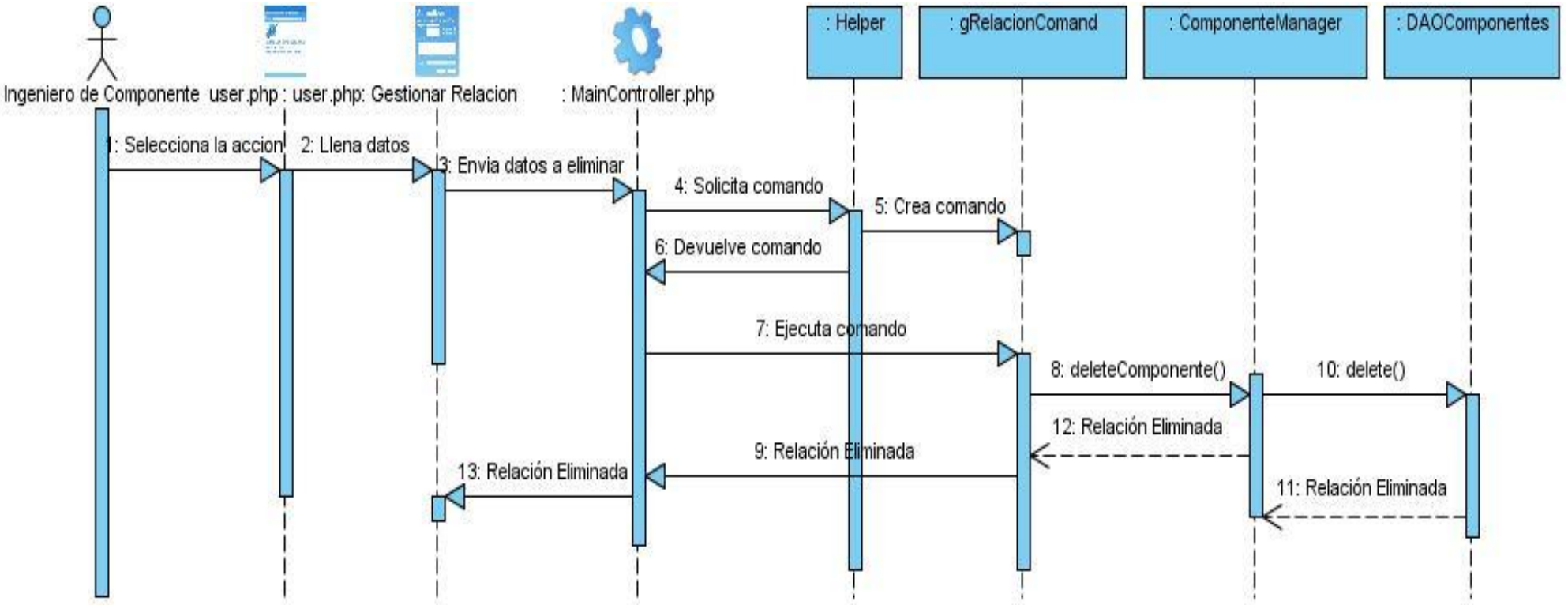


Figura 18: Diagrama de Secuencia "CU Eliminar Relación"

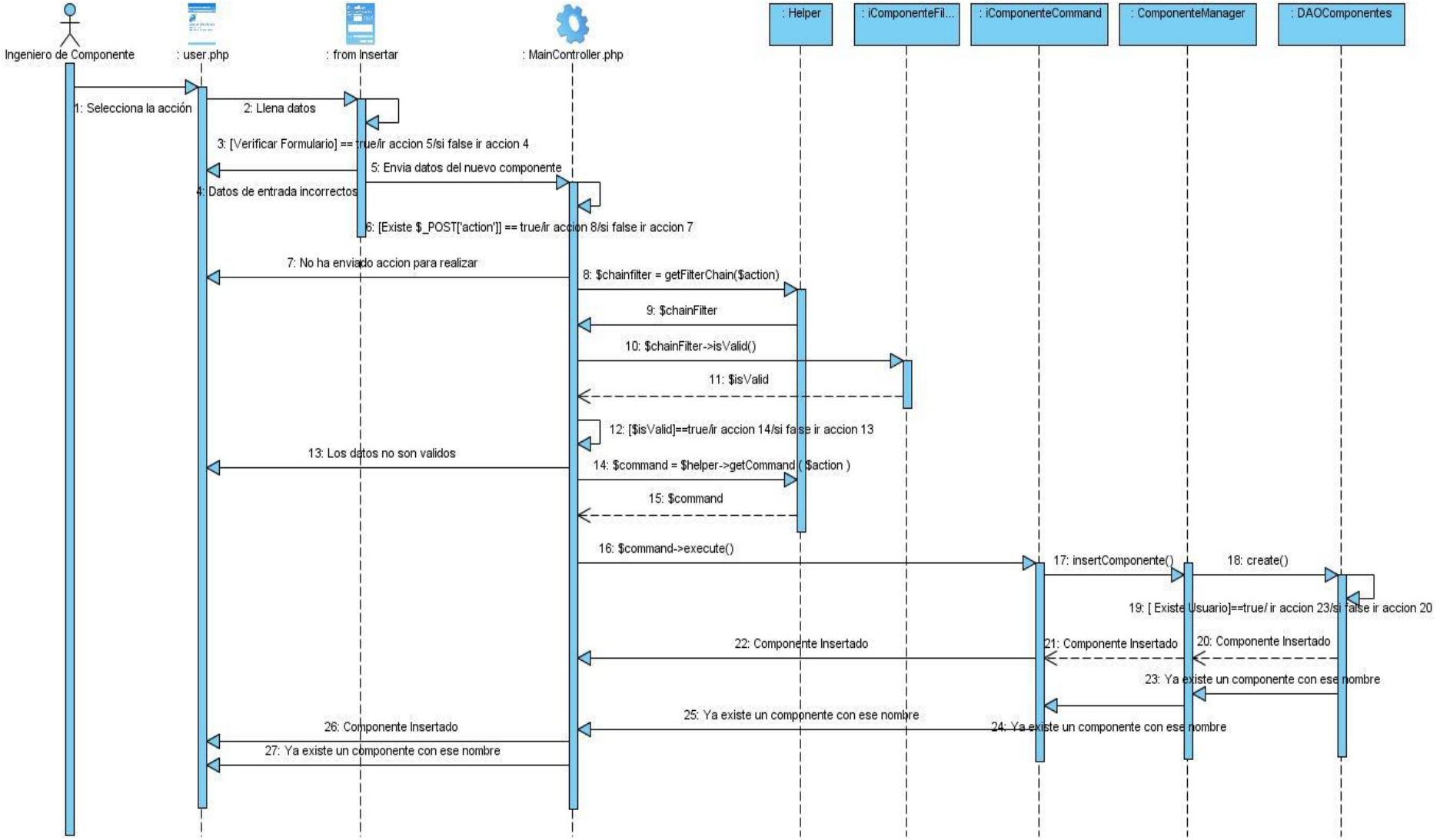


Figura 19: Diagrama de Secuencia "CU Insertar Componente"

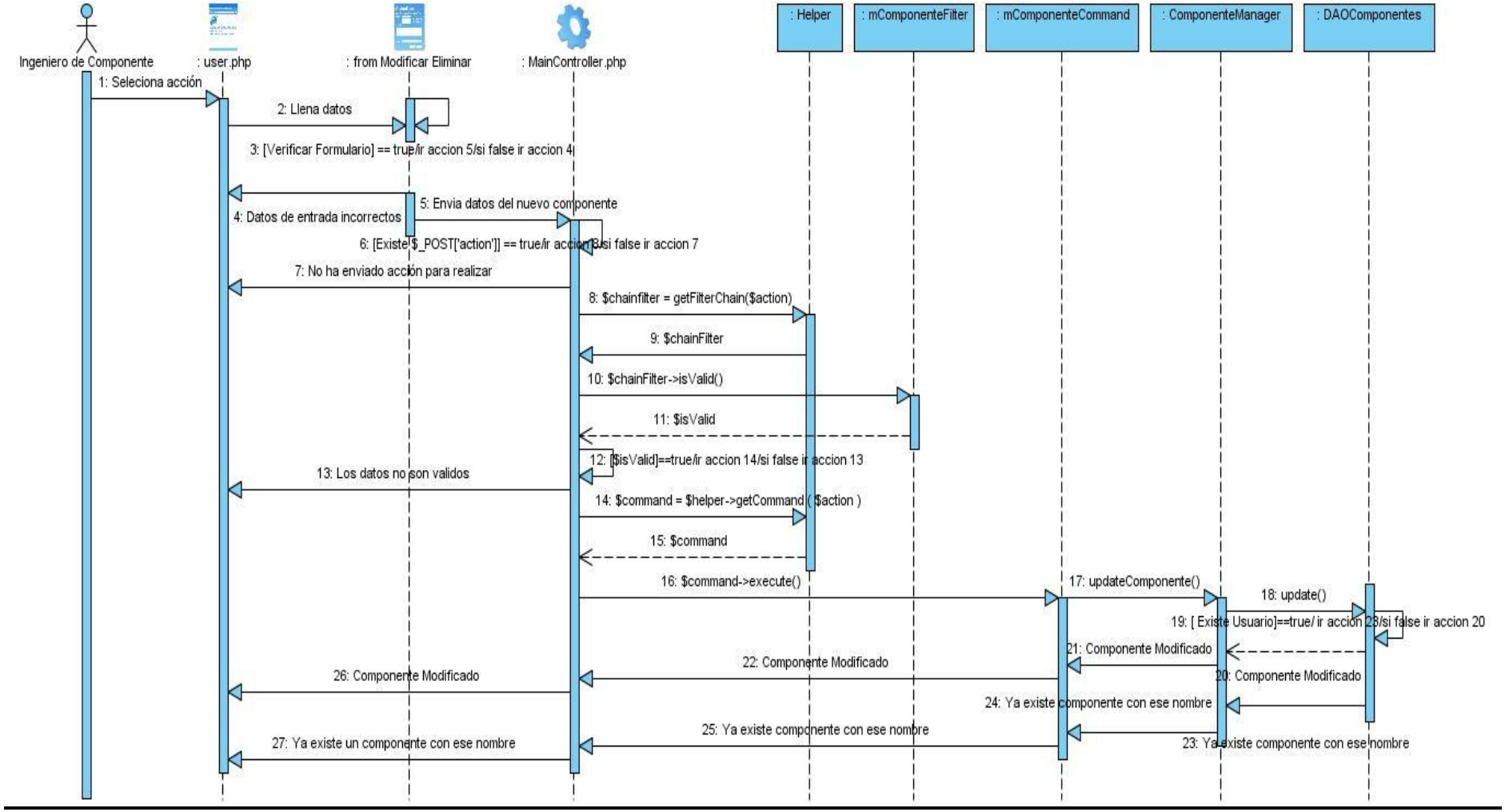


Figura 20: Diagrama de Secuencia "CU Modificar Componente"

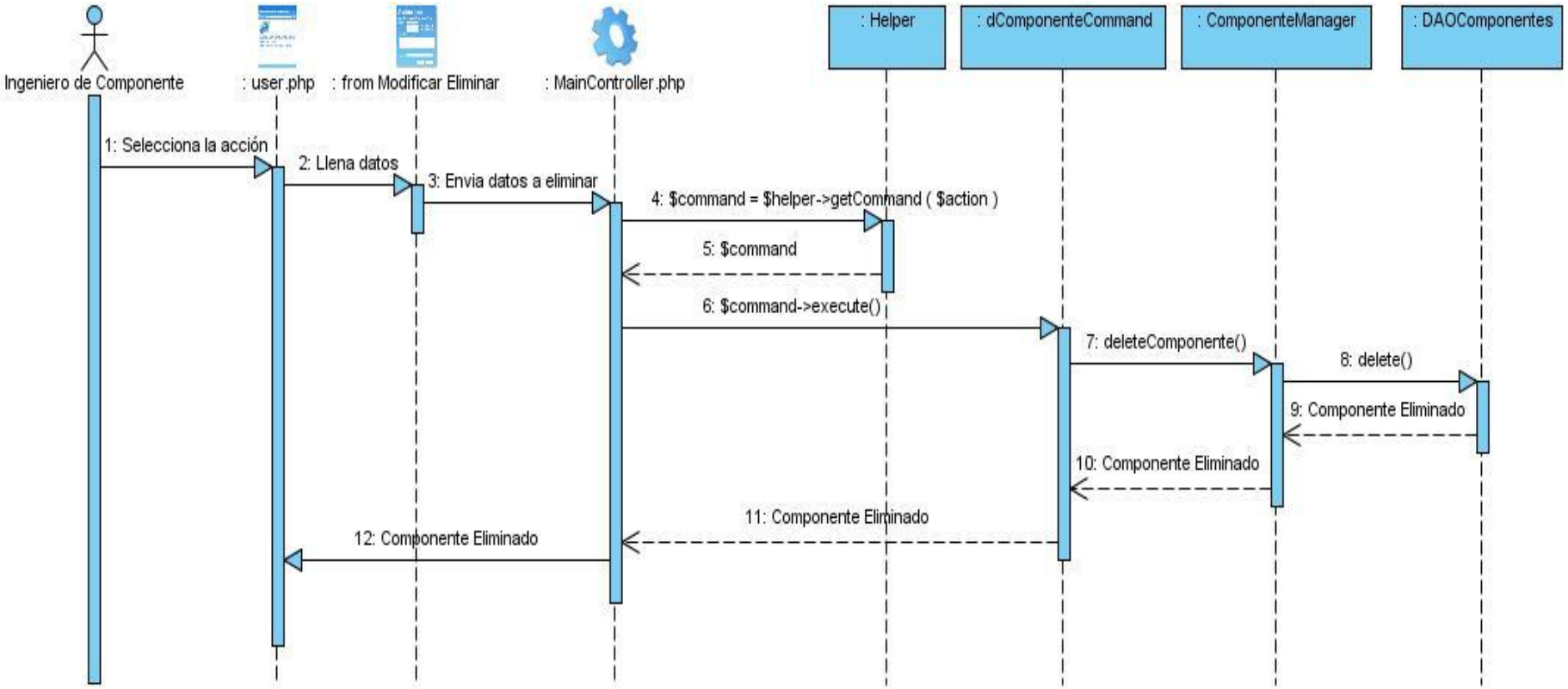


Figura 21: Diagrama de Secuencia "CU Eliminar Componente"

3.6 Diagrama de Clases Persistentes.

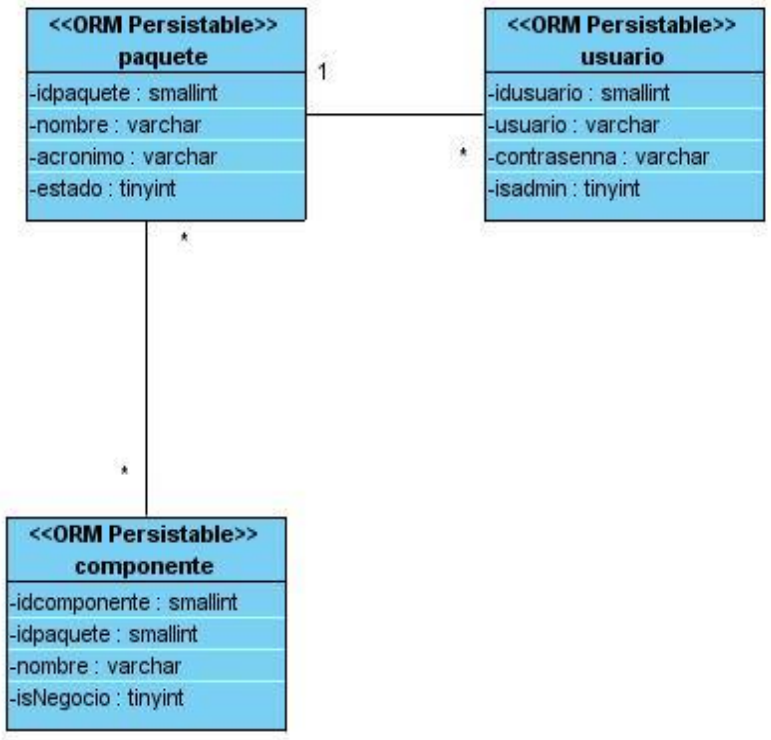


Figura 22: Diagrama de Clases Persistentes

3.7 Modelo de Datos.

El objetivo de construir un Modelo de Datos es identificar y representar las tablas importantes para el funcionamiento del negocio (entidades), sus propiedades (atributos), y la forma en que las tablas se relacionan entre sí (relaciones). El Modelo de Datos se va a generar a partir del Diagrama de Clases Persistentes hecho en la herramienta Visual Paradigm.

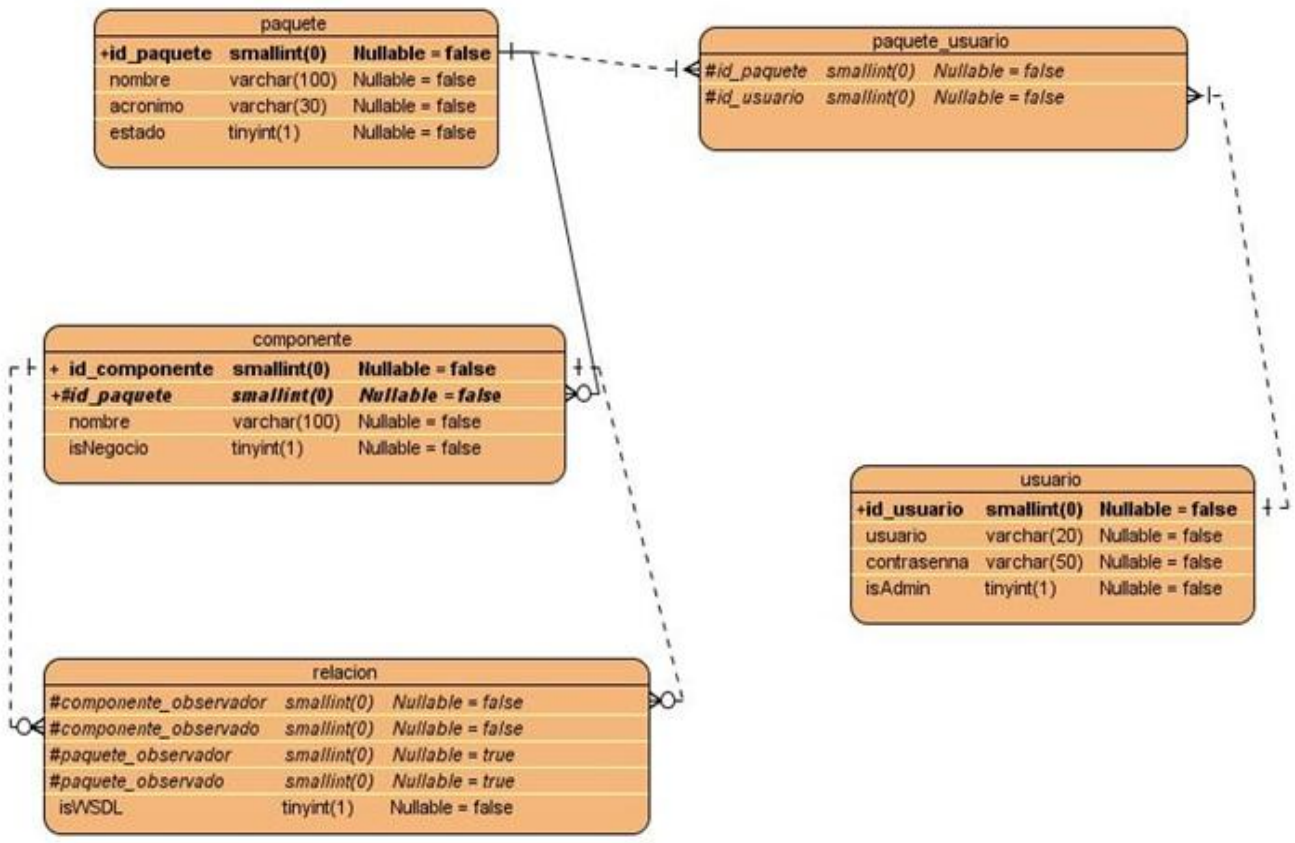


Figura 23: Modelo de Datos

3.8 Modelo de Despliegue.

El Modelo de Despliegue describe cómo y dónde el sistema será puesto en funcionamiento. Las estaciones de trabajo, dispositivos y procesadores son reflejados como nodos y su estructura interna puede ser representada adicionando otros nodos o artefactos. (23)

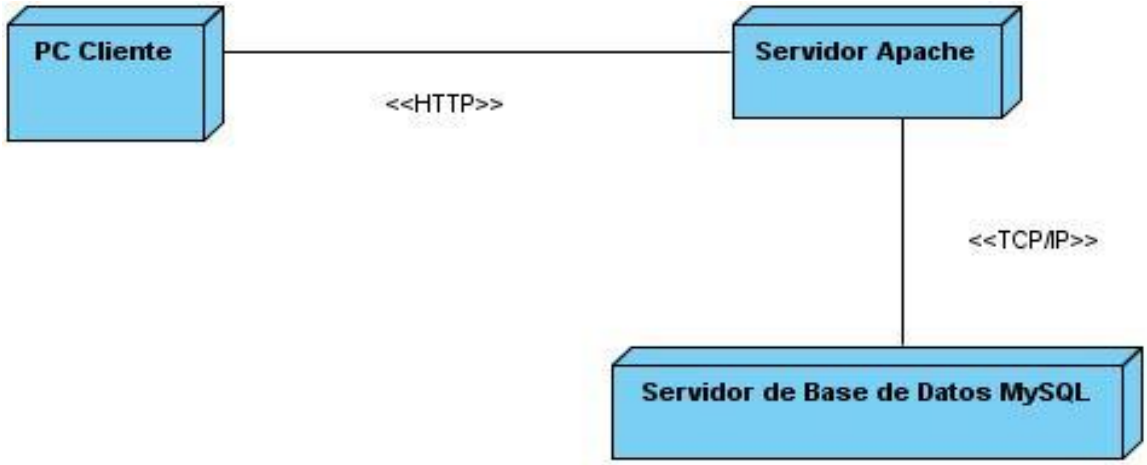


Figura 24: Modelo de Despliegue.

Conclusiones del Capítulo

En este capítulo se obtiene la estructura general del software, donde se aplicaron los patrones de arquitectura y diseño. Además de que se realizaron los diagramas de clases y de interacción para lograr que las clases, objetos, paquetes y otros elementos queden con la organización y detalles suficientes para que los programadores puedan desempeñar su rol correctamente.

Capítulo 4: Implementación del Sistema

Este capítulo contiene las principales características de la implementación del Sistema. Además de los principales componentes y sus relaciones, mostrados de forma sencilla a través del Diagrama de Componentes.

4.1 Diagrama de Componentes

Un Diagrama de Componentes contiene componentes, interfaces y relaciones entre ellos. Muestra las organizaciones y dependencias lógicas entre componentes de software. Normalmente los Diagramas de Componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas, entre otros. Cada componente debe tener un nombre que lo distinga de los demás.

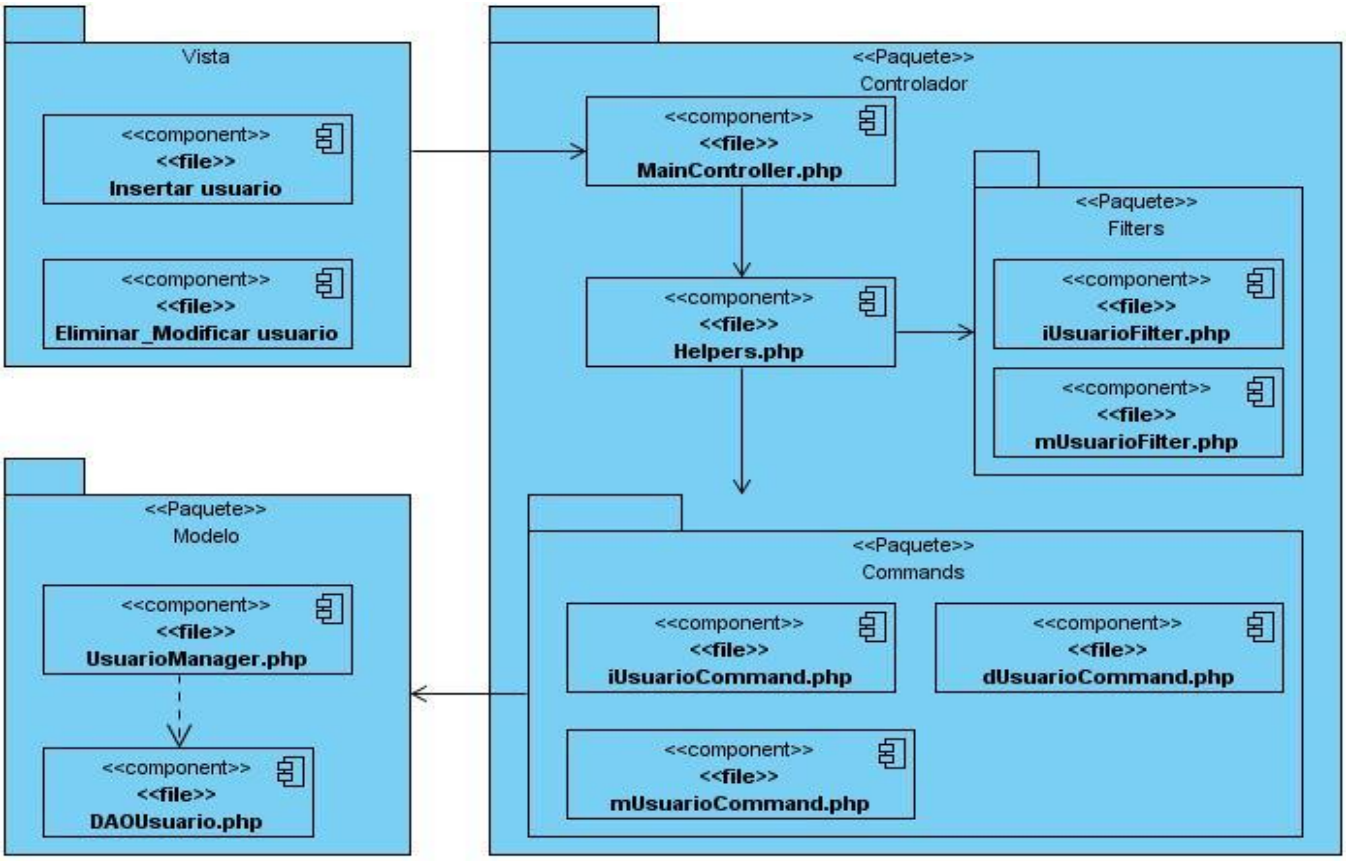


Figura 25: Diagrama de Componentes del "CU Gestionar Usuario"

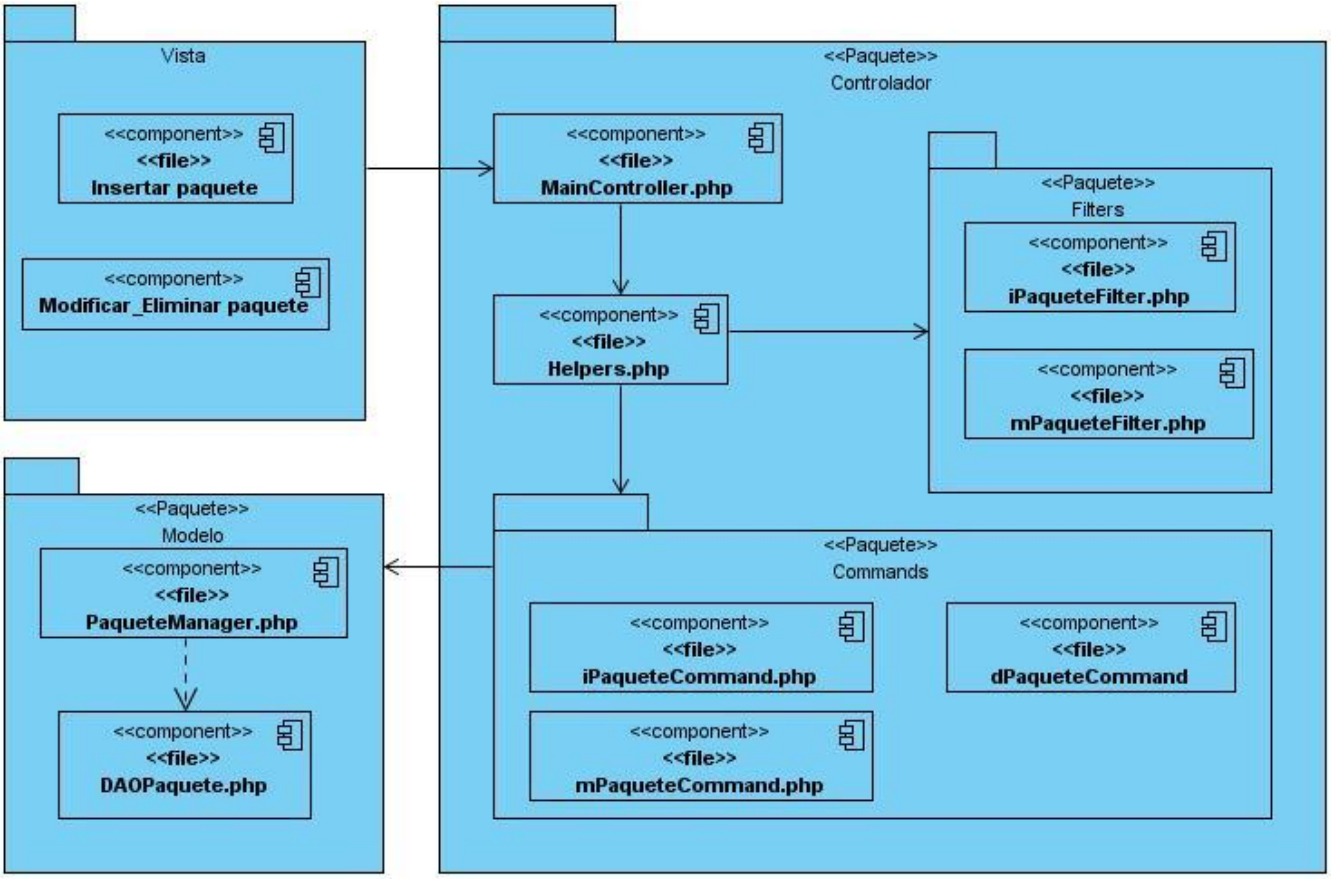


Figura 26: Diagrama de Componentes del "CU Gestionar Paquete"

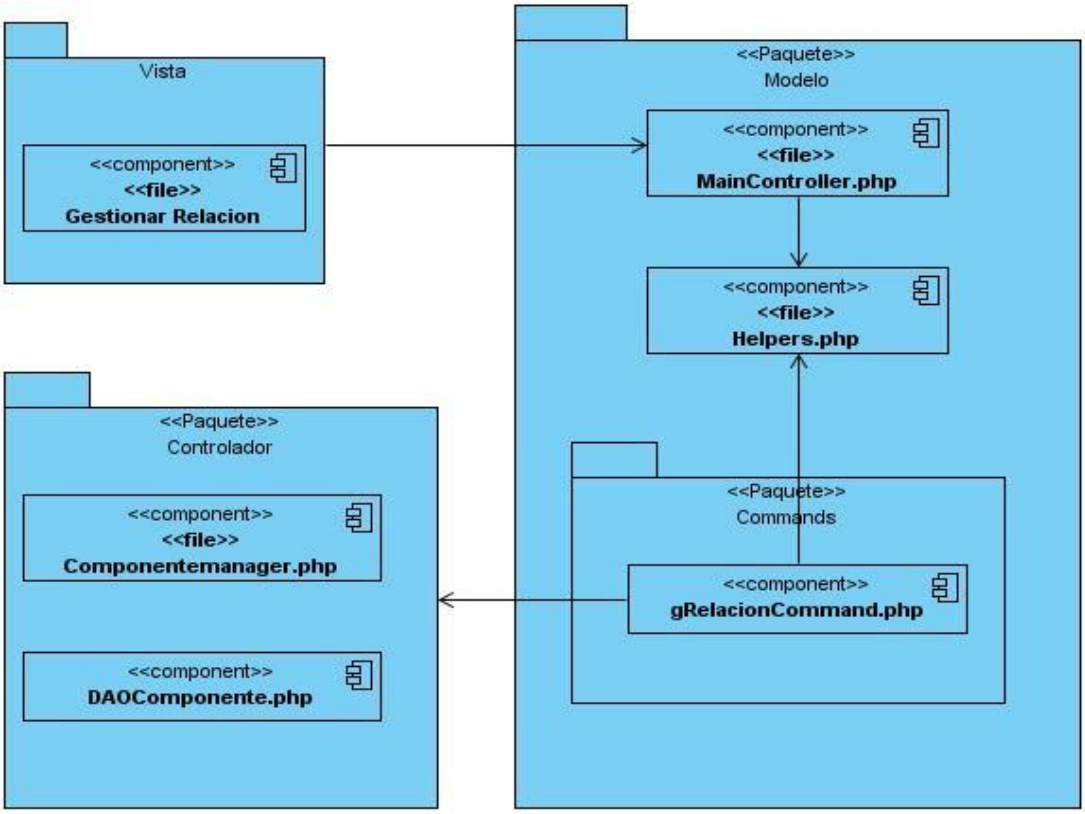


Figura 27: Diagrama de Componentes del "CU Gestionar Relaciones"

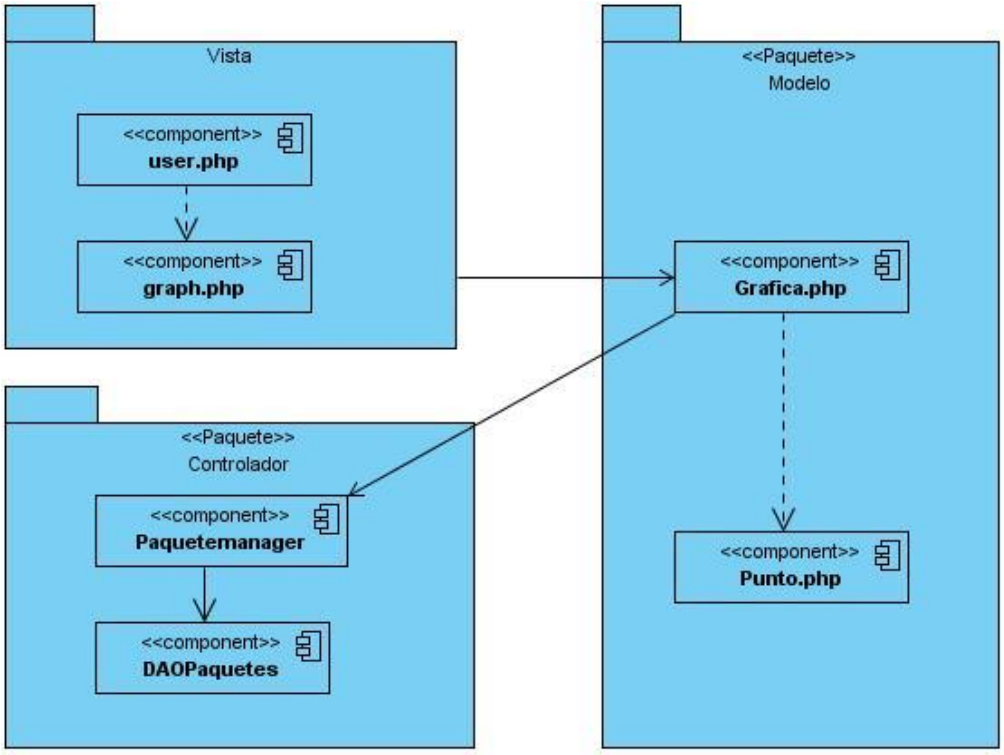


Figura 28: Diagrama de Componentes del "CU Generar Diagrama"

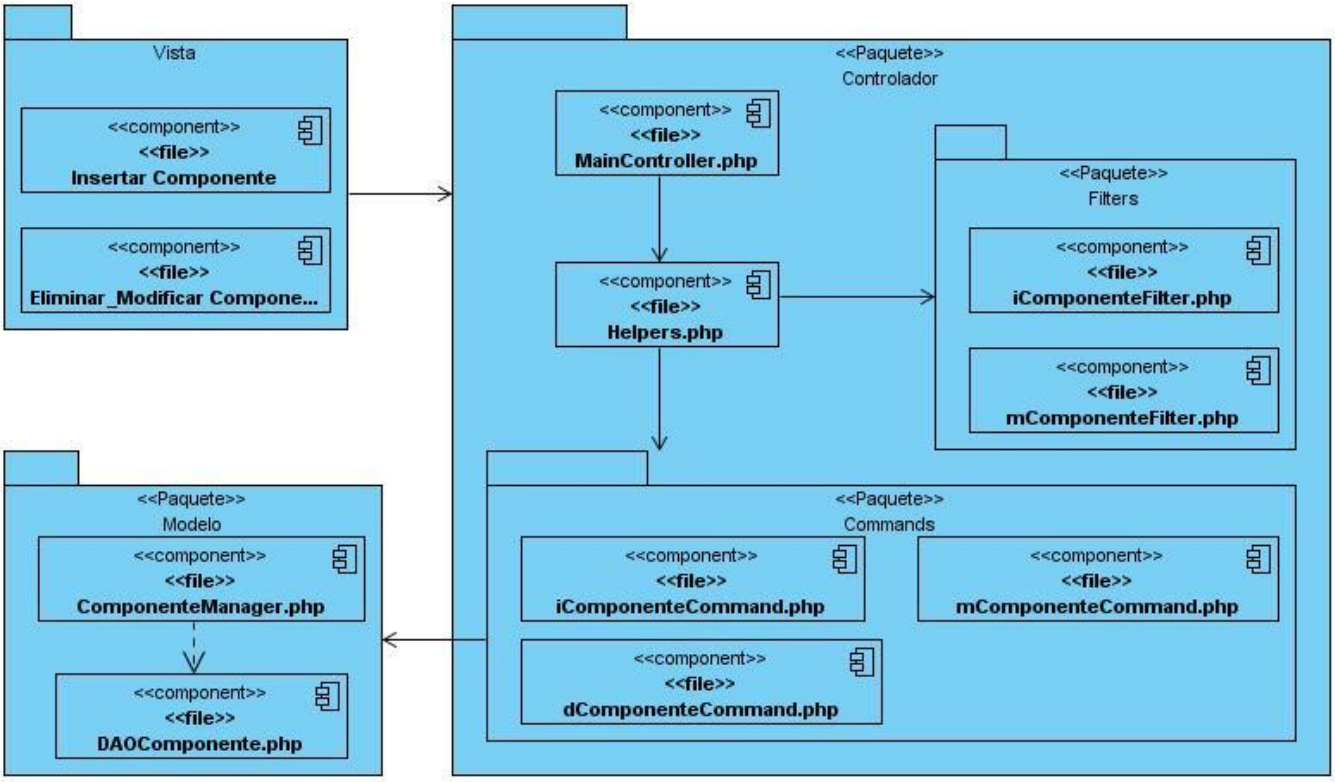


Figura 29: Diagrama de Componentes del "CU Gestionar Componente"

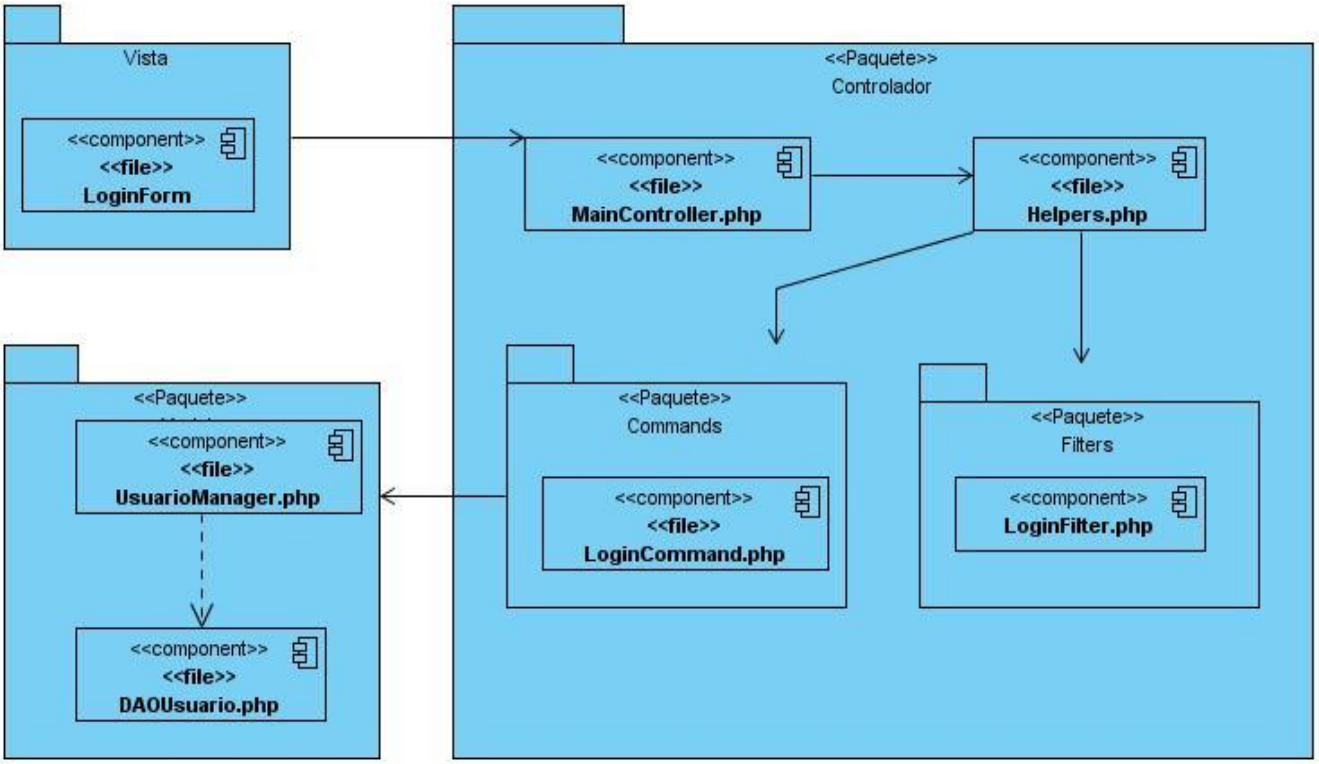


Figura 30: Diagrama de Componentes del "CU Autenticar Usuario"

4.2 Códigos Fuente.

Esta es la clase para el dibujo del Diagrama de Paquetes de Componentes de SISalud.

```
<?php
include_once 'Punto.php';
class Grafica {
      private $n;
      private $m;
      private $image;
      private $paquetes;
      private $altoImagen;
      private $anchoImagen;
      private $anchoCuadricula;
      private $altoCuadricula;
      const margen = 20;
      const anchoPestaña = 30;
      const altoPestaña = 15;
      const anchoPaquete = 90;
      const altoPaquete = 50;
```

```
	public function Grafica() {
		$pq = new PaquetesManager ( );
		$this->paquetes = $pq->getPaquetes ();
		$this->image = null;

		if (count ( $this->paquetes ) <= 5) {
			$this->altoImagen = 600;
			$this->anchoImagen = 800;
		} else if (count ( $this->paquetes ) > 5 && count ( $this->paquetes ) <=
15) {
			$this->altoImagen = 768;
			$this->anchoImagen = 1024;
		} else if (count ( $this->paquetes ) > 15 && count ( $this->paquetes )
<= 25) {
			$this->altoImagen = 900;
			$this->anchoImagen = 1440;
		} else {
			$this->altoImagen = 1024;
			$this->anchoImagen = 2000;
		}

		$this->anchoCuadricula = self::anchoPaquete + (2 * self::margen);
		$this->altoCuadricula = self::altoPestaña + self::altoPaquete + (2 *
self::margen);
		$this->m = round ( $this->anchoImagen / $this->anchoCuadricula );
		$this->n = round ( $this->altoImagen / $this->altoCuadricula );
	}



	private function crearImagen() {
		$this->image = imagecreate ( $this->anchoImagen, $this->altoImagen );
	}

	private function destruirImagen() {
		imagepng ( $this->image );
		imagedestroy ( $this->image );
	}


	private function crearMatriz() {
		for($y = 0; $y < $this->n; $y ++)
			for($x = 0; $x <= $this->m; $x ++)
				$matriz [$x] [$y] = new Punto ( $x * $this->anchoCuadricula,
$y * $this->altoCuadricula );
		return $matriz;
	}


	private function crearListaCandidataDiagramaGeneral() {
		$matriz = $this->crearMatriz ();
		$lista = array ();
		for($x = 0; $x < $this->n; $x ++)
			for($y = 0; $y <= $this->m - 2; $y ++)
				array_push ( &$lista, $matriz [$x] [$y] );
```

```
		return $lista;
	}

	private function crearListaCandidataObservadores() {
		$matriz = $this->crearMatriz ();
		$lista = array ();
		for($x = 0; $x < round ( $this->n / 2 ) - 1; $x ++)
			for($y = 0; $y <= $this->m - 2; $y ++)
				array_push ( &$lista, $matriz [$x] [$y] );
		return $lista;
	}

	private function crearListaCandidataObservados() {
		$matriz = $this->crearMatriz ();
		$lista = array ();
		for($x = round ( $this->n / 2 ) + 1; $x <= $this->n; $x ++)
			for($y = 0; $y <= $this->m - 2; $y ++)
				array_push ( &$lista, $matriz [$x] [$y] );
		return $lista;
	}

	private function dibujarPaquete($punto, $acronimo) {
		//Creando los puntos:
		$X1 = $punto->x;
		$Y1 = $punto->y; //Puntos iniciales
		$X2 = $punto->x + self::anchoPaquete;
		$Y2 = $punto->y + self::altoPaquete;
		$X3 = $punto->x + self::anchoPestaña;
		$Y3 = $punto->y - self::altoPestaña;
		// Creando la matriz de los verices del poligono:
		$matriz = array ($X1, $Y1, // Punto 1 (x, y)
				$X2, $Y1, // Punto 2 (x, y)
				$X2, $Y2, // Punto 3 (x, y)
				$X1, $Y2, // Punto 4 (x, y)
				$X1, $Y3, // Punto 5 (x, y)
				$X3, $Y3, // Punto 6 (x, y)
				$X3, $Y1 // Punto 7 (x,y)
				);

		imagecolorallocate ( $this->image, 255, 255, 255 );
		$color = imagecolorallocate ( $this->image, 168, 233, 158 );
		$text_color = imagecolorallocate ( $this->image, 0, 0, 0 );

		imagefilledpolygon ( $this->image, $matriz, 7, $color );

		imageline ( $this->image, $X1, $Y1, $X2, $Y1, $text_color );
		imageline ( $this->image, $X2, $Y1, $X2, $Y2, $text_color );
		imageline ( $this->image, $X2, $Y2, $X1, $Y2, $text_color );
		imageline ( $this->image, $X1, $Y2, $X1, $Y3, $text_color );
		imageline ( $this->image, $X1, $Y3, $X3, $Y3, $text_color );
		imageline ( $this->image, $X3, $Y3, $X3, $Y1, $text_color );

		imagestring ( $this->image, 1, $X1 + 15, $Y1 + 2, "<<Package>>",
$text_color );
```

```
            imagestring ( $this->image, 10, $X1 + 20, $Y1 + 20, $acronimo,
$text_color );
      }

      private function dibujarFlecha($puntosI, $puntosF) {
            $X1 = $puntosI->x;
            $Y1 = $puntosI->y;
            $X2 = $puntosF->x;
            $Y2 = $puntosF->y;

            if ($X1 == $X2) {
                  if ($Y1 < $Y2) {
                        imageline ( $this->image, $X1 + self::anchoPaquete / 2, $Y1
+ self::altoPaquete, $X2 + self::anchoPaquete / 2, $Y2, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete / 2, $Y2,
$X2 + self::anchoPaquete / 2 - 4, $Y2 - 15, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete / 2, $Y2,
$X2 + self::anchoPaquete / 2 + 4, $Y2 - 15, 255 );
                  } else if ($Y1 > $Y2) {
                        imageline ( $this->image, $X1 + self::anchoPaquete / 2, $Y1,
$X2 + self::anchoPaquete / 2, $Y2 + self::altoPaquete, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete / 2, $Y2
+ self::altoPaquete, $X2 + (self::anchoPaquete / 2) - 4, $Y2 + self::altoPaquete +
15, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete / 2, $Y2
+ self::altoPaquete, $X2 + (self::anchoPaquete / 2) + 4, $Y2 + self::altoPaquete +
15, 255 );
                  }
            } else if ($Y1 == $Y2) {
                  if ($X1 < $X2) {
                        imageline ( $this->image, $X1 + self::anchoPaquete, $Y1 +
self::altoPaquete / 2, $X2, $Y2 + self::altoPaquete / 2, 255 );
                        imageline ( $this->image, $X2, $Y2 + self::altoPaquete / 2,
$X2 - 15, $Y2 + (self::altoPaquete / 2) - 4, 255 );
                        imageline ( $this->image, $X2, $Y2 + self::altoPaquete / 2,
$X2 - 15, $Y2 + (self::altoPaquete / 2) + 4, 255 );
                  } else if ($X1 > $X2) {
                        imageline ( $this->image, $X1, $Y1 + self::altoPaquete / 2,
$X2 + self::anchoPaquete, $Y2 + self::altoPaquete / 2, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete, $Y2 +
self::altoPaquete / 2, $X2 + self::anchoPaquete + 15, $Y2 + (self::altoPaquete / 2)
- 4, 255 );
                        imageline ( $this->image, $X2 + self::anchoPaquete, $Y2 +
self::altoPaquete / 2, $X2 + self::anchoPaquete + 15, $Y2 + (self::altoPaquete / 2)
+ 4, 255 );
                  }
            } else if ($X1 > $X2) {
                  if ($Y1 > $Y2) {
                        imageline ( $this->image, $X1, $Y1, $X2 +
self::anchoPaquete, $Y2 + self::altoPaquete, 255 );
                        imagefilledellipse ( $this->image, $X2 + self::anchoPaquete,
$Y2 + self::altoPaquete, 10, 10, 255 );
                  } else if ($Y1 < $Y2) {
                        imageline ( $this->image, $X1, $Y1 + self::altoPaquete, $X2
+ self::anchoPaquete, $Y2 + self::altoPaquete / 2, 255 );
```

```
                        imagefilledellipse ( $this->image, $X2 + self::anchoPaquete,
$Y2 + self::altoPaquete / 2, 10, 10, 255 );
                  }
            } else if ($X1 < $X2) {
                  if ($Y1 < $Y2) {
                        imageline ( $this->image, $X1 + self::anchoPaquete, $Y1 +
self::altoPaquete, $X2, $Y2 + self::altoPaquete / 2, 255 );
                        imagefilledellipse ( $this->image, $X2, $Y2 +
self::altoPaquete / 2, 10, 10, 255 );
                  } else if ($Y1 > $Y2) {
                        imageline ( $this->image, $X1 + self::anchoPaquete, $Y1,
$X2, $Y2 + self::altoPaquete, 255 );
                        imagefilledellipse ( $this->image, $X2, $Y2 +
self::altoPaquete, 10, 10, 255 );
                  }
            }
      }

      private function seleccionarPunto() {
            $lista = array_reverse ( $this->crearListaCandidataDiagramaGeneral () );
            $rand = array_rand ( $lista );
            $x = $lista [$rand]->x + self::margen;
            $y = $lista [$rand]->y + self::margen + self::altoPestaña;
            $punto = new Punto ( $x, $y );
            return $punto;
      }

      private function seleccionarPuntoObservadores() {
            $lista = $this->crearListaCandidataObservadores ();
            $rand = array_rand ( $lista );
            $x = $lista [$rand]->x + self::margen;
            $y = $lista [$rand]->y + self::margen + self::altoPestaña;
            $punto = new Punto ( $x, $y );
            return $punto;
      }

      private function seleccionarPuntoObservados() {
            $lista = $this->crearListaCandidataObservados ();
            $rand = array_rand ( $lista );
            $x = $lista [$rand]->x + self::margen;
            $y = $lista [$rand]->y + self::margen + self::altoPestaña;
            $punto = new Punto ( $x, $y );
            return $punto;
      }

      public function dibujarDiagramaGeneral() {
            $ocupados = array ();
            $puntosPaquetes = array ();
            $this->crearImagen ();
            for($i = 0; $i < count ( $this->paquetes ); $i ++) {
                  $punto = $this->seleccionarPunto ();
                  while ( array_search ( $punto, $ocupados ) != '' ) {
                        $punto = $this->seleccionarPunto ();
                  }
```

```
                  $this->dibujarPaquete ( $punto, $this->paquetes [$i]->getAcronimo
() ); //$this->paquetes[$i]->getAcronimo()
                  array_push ( &$ocupados, $punto );
                  $puntosPaquetes [$this->paquetes [$i]->getId ()] = $punto;
            }

            foreach ( $this->paquetes as $pq ) {
                  foreach ( $pq->getListaComponentes () as $componente ) {
                        foreach ( $componente->getLComponentesObservados () as
$observado ) {
                              if (array_key_exists ( $observado->getIdPaquete (),
$puntosPaquetes ) && $observado->getIdPaquete () != $pq->getId ()) {
                                    $this->dibujarFlecha ( $puntosPaquetes [$pq-
>getId ()], $puntosPaquetes [$observado->getIdPaquete ()] );
                              }
                        }
                  }
            }
            $this->destruirImagen ();
      }

      public function dibujarDiagramaEspecifico($id_paquete) {
            $matriz = $this->crearMatriz ();
            $cuadriculaCentro = $matriz [round ( $this->n / 2 ) - 1] [round ( $this-
>m / 2 ) - 1];
            $puntoCentro = new Punto ( $cuadriculaCentro->x + self::margen,
$cuadriculaCentro->y + self::margen + self::altoPestaña );
            foreach ( $this->paquetes as $pq ) {
                  if ($pq->getId () == $id_paquete)
                        break;
            }
            //Creo la imagen
            $this->crearImagen ();
            //Dibujar el paquete del centro
            $this->dibujarPaquete ( $puntoCentro, $pq->getAcronimo () );
            $ocupadosObservadores = array ();
            $ocupadosObservados = array ();
            $arrayPintadosObservadores = array ();
            $arrayPintadosObservados = array();
            $countObservadores = -1;
            $countObservados = -1;
            //Dibujar los Observadores
            if (count ( $pq->getListaComponentes () ) > 0) {
                  foreach ( $pq->getListaComponentes () as $cmp ) {
                        foreach($cmp->getLComponentesObservadores() as $observador){
                              $countObservadores ++;
                              $paqueteObservador = $this->getPaquete ( $observador-
>getIdPaquete () );
                              if ($paqueteObservador->getId () != $pq->getId () &&
array_search ( $paqueteObservador->getId (), $arrayPintadosObservadores ) === false)
{
                                    $puntoObservador = $this-
>seleccionarPuntoObservadores ();
                                    while ( array_search ( $puntoObservador,
$ocupadosObservadores ) != '' ) {
```

```
                                        $puntoObservador = $this-
>seleccionarPuntoObservadores ();
                                    }
                                    $this->dibujarPaquete ( $puntoObservador,
$paqueteObservador->getAcronimo () );
                                    $this->dibujarFlecha ( $puntoObservador,
$puntoCentro );
                                    array_push ( $ocupadosObservadores,
$puntoObservador );
                                    array_push ( $arrayPintadosObservadores,
$paqueteObservador->getId () );
                                }
                        }
                        foreach ( $cmp->getLComponentesObservados () as $observados
) {
                              $countObservados ++;
                              $paqueteObservado = $this->getPaquete ( $observados-
>getIdPaquete () );
                              if ($paqueteObservado->getId () != $pq->getId () &&
array_search ( $paqueteObservado->getId (), $arrayPintadosObservados ) === false) {
                                    $puntoObservado= $this-
>seleccionarPuntoObservados ();
                                    while ( array_search ( $puntoObservado,
$ocupadosObservados ) != '' ) {
                                          $puntoObservadoObservador = $this-
>seleccionarPuntoObservadores ();
                                    }
                                    $this->dibujarPaquete ( $puntoObservado,
$paqueteObservado->getAcronimo () );
                                    $this->dibujarFlecha ( $puntoCentro,
$puntoObservado  );
                                    array_push ( $ocupadosObservadores,
$puntoObservadoObservador );
                                    array_push ( $arrayPintadosObservados,
$paqueteObservado->getId () );
                              }
                        }
                  }
                  if ($countObservadores < 0)
                        imagestring ( $this->image, 20, $puntoCentro->x - 300,
$puntoCentro->y, 'No tiene observadores', 255 );
                  if ($countObservados < 0)
                        imagestring ( $this->image, 20, $puntoCentro->x + 200,
$puntoCentro->y, 'No observa a ningun Paquete', 255 );

            } else {
                  imagestring ( $this->image, 20, $puntoCentro->x - 300,
$puntoCentro->y, 'No tiene observadores', 255 );
                  imagestring ( $this->image, 20, $puntoCentro->x + 200,
$puntoCentro->y, 'No observa a ningun Paquete', 255 );
            }
            $this->destruirImagen ();
      }

      private function getPaquete($id_cmp) {
```

```
            foreach ( $this->paquetes as $pq ) {
                  if ($pq->getId () == $id_cmp) {
                        return $pq;
                        break;
                  }
            }
            return ;
      }
}
?>
```

4.3 Interfaces de la Aplicación.

Figura 31: Inicio del Sistema

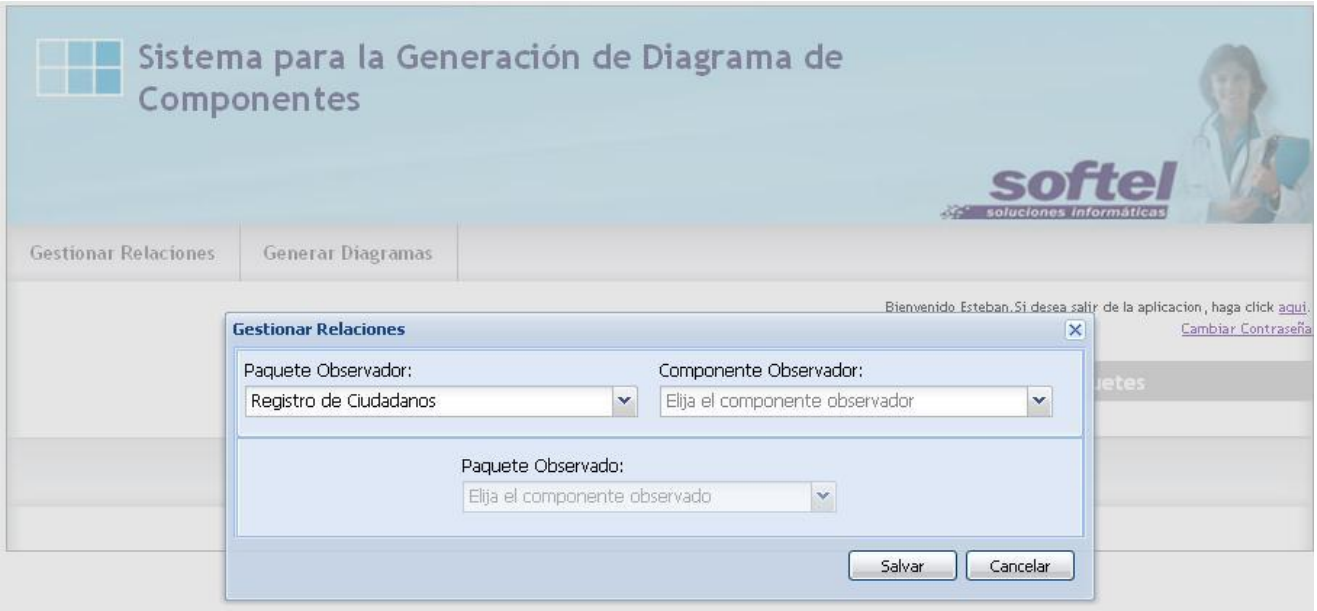


Figura 32: Gestionar Relaciones

4.4 Validación del Sistema

Para comprobar el eficiente funcionamiento del sistema desarrollado se decidió hacer una serie de pruebas exploratorias que fueron aplicadas con el objetivo fundamental de verificar el cumplimiento de cada una de las descripciones textuales de los casos de uso del sistema expuestas con anterioridad.

4.4.1 Prueba al Caso de Uso: Autenticar Usuario

Figura 33: Inicio del Sistema

Para desarrollar las pruebas de validación en el caso de uso “Autenticar Usuario” se decidió seleccionar un usuario del sistema en este caso el usuario “Admin” y con los datos de este completar los campos vacios para realizar su autenticación y de esta manera comprobar el buen funcionamiento de la aplicación. En las Figuras 33 y 34 se muestra el sistema antes y después de la entrada del administrador “Admin”.

Figura 34: Sección de Admin

4.4.2 Prueba al Caso de Uso: Gestionar Usuario

Figura 35: Insertar Usuario

En el Caso de Uso “Gestionar Usuario” se decidió realizarle la prueba al escenario de este caso de uso “Insertar Usuario” de esta forma evidenciando que se obtuvo un correcto funcionamiento tras la inserción del nuevo usuario. En las Figuras 35 y 36 se muestra como se realizó esta acción.

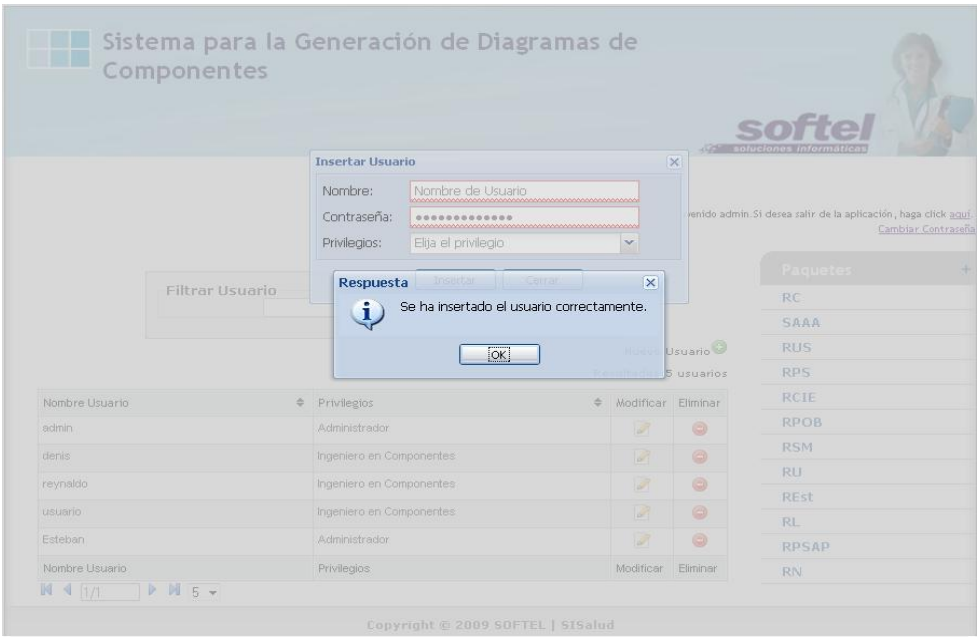


Figura 36: Mensaje del sistema de correcta inserción del nuevo usuario “Esteban”

4.4.3 Prueba al Caso de Uso: Gestionar Paquetes de Componente

Figura 37: Modificar Paquetes

Para el caso de uso “Gestionar Paquete de Componente” se seleccionó uno de los tres escenarios que presenta Gestionar Paquete de Componente en este caso “Modificar Componente” y se decidió modificarle sus datos actuales. En las Figuras 37 y 38 citamos ejemplos de imágenes de cómo se realizó lo antes comentado.

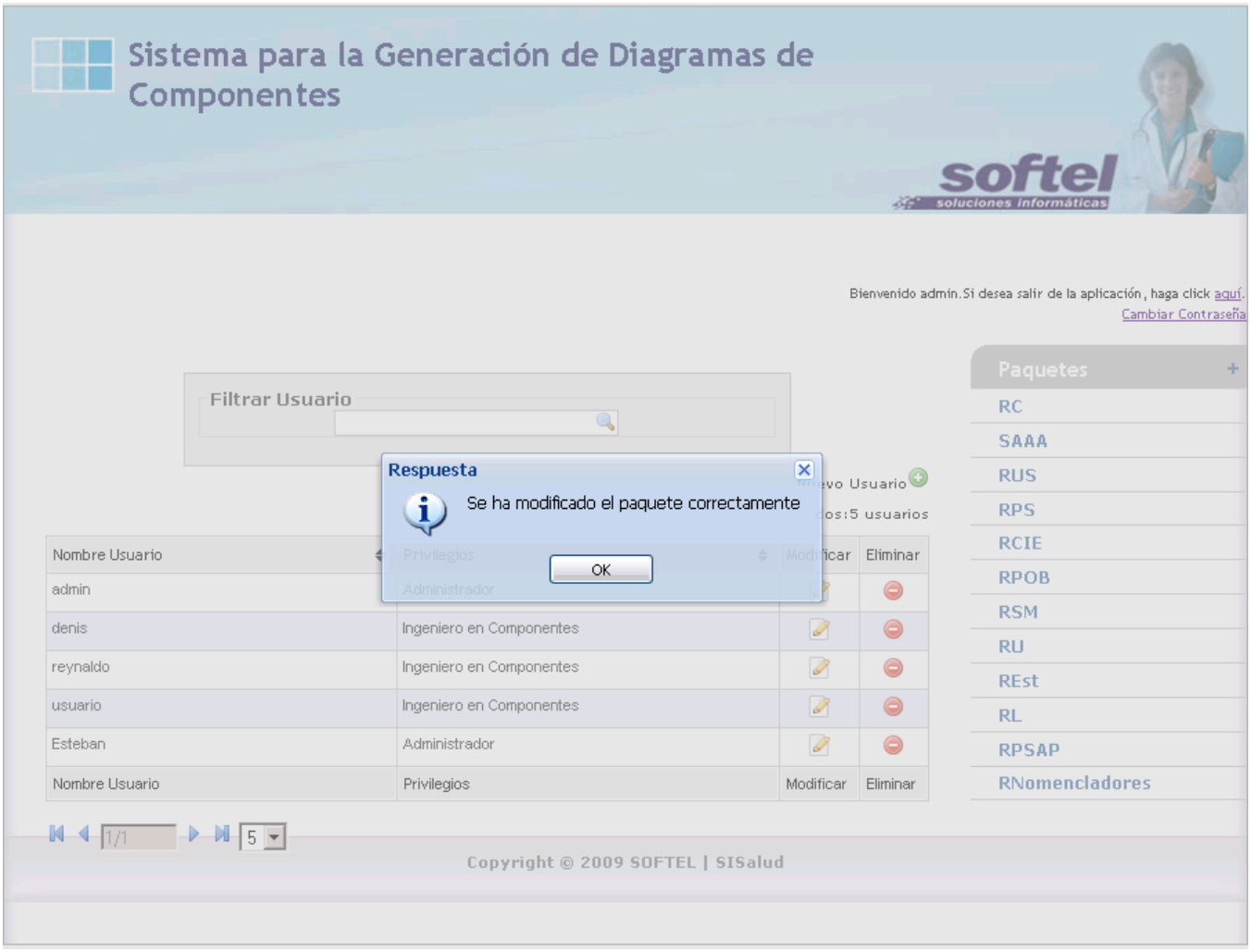


Figura 38: Mensaje del sistema de correcta modificación del paquete “RNomencladores”

4.4.4 Prueba al Caso de Uso: “Gestionar Componente”

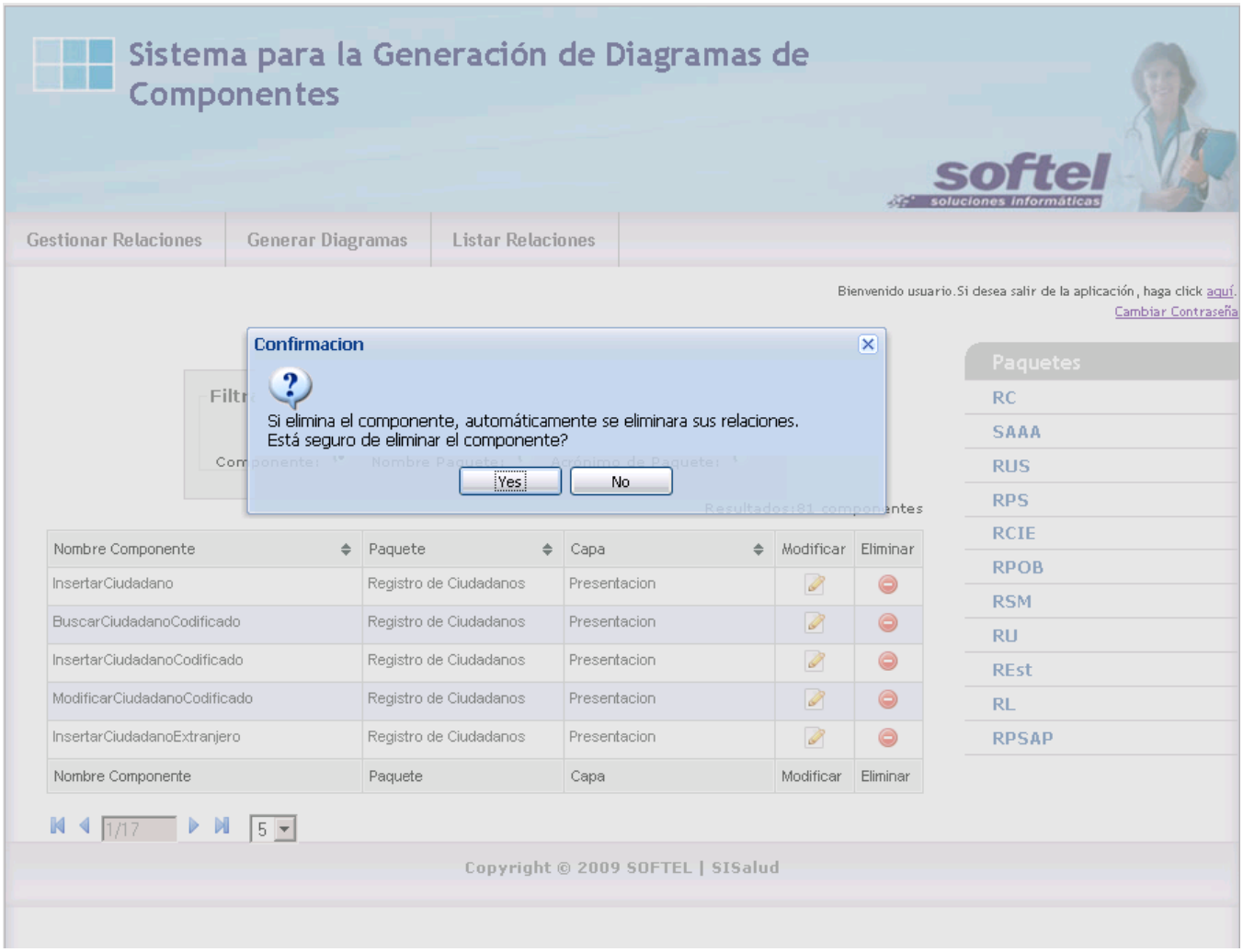


Figura 39: Mensaje de Confirmación para la eliminación del componente

En el caso de uso “Gestionar Componente” se le realizó la prueba de validación al escenario “Eliminar Componente” eliminando el componente “InsertarCiudadano”. En las Figuras 39 y 40 se muestra como se ejecutó esta prueba.

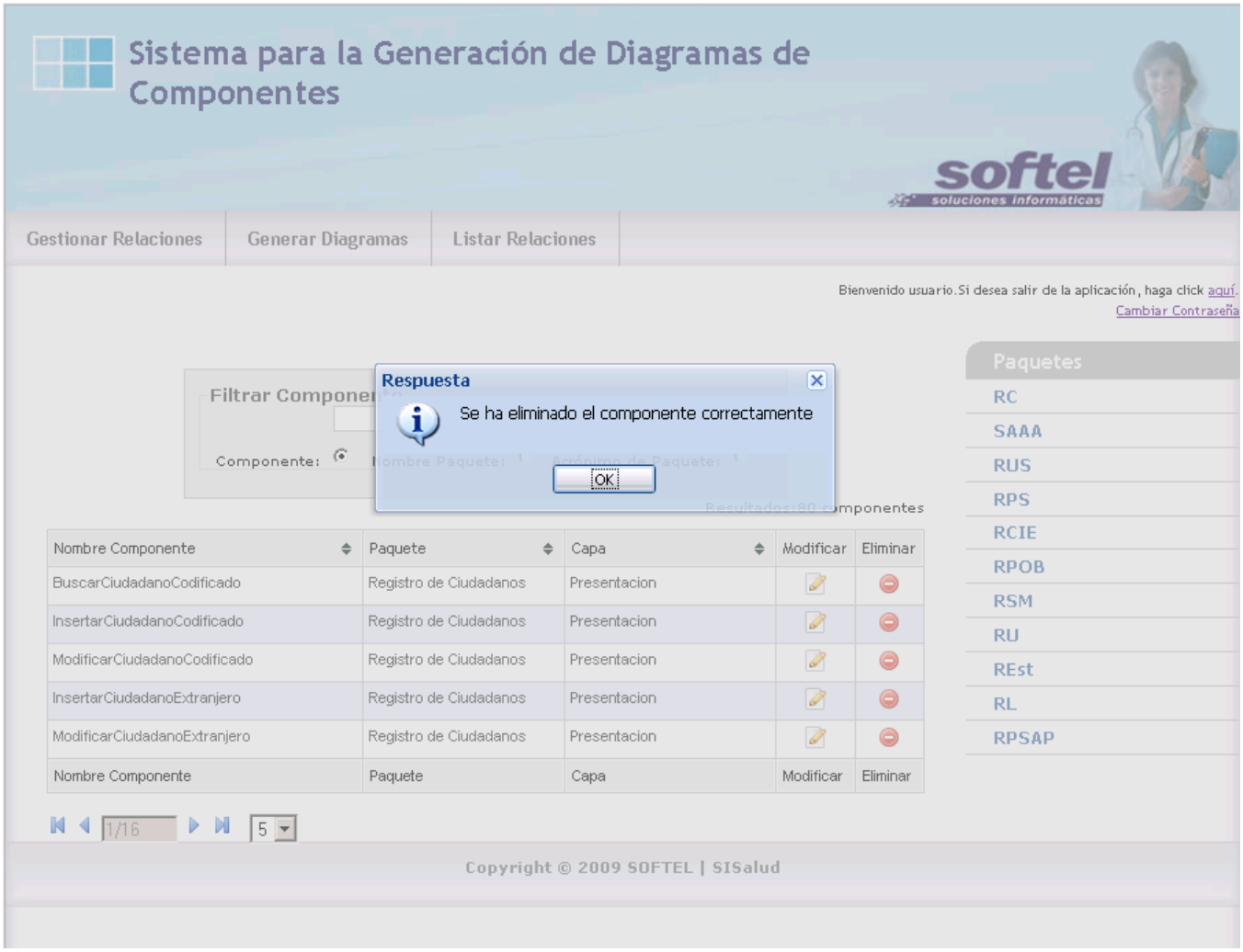


Figura 40: Mensaje del sistema que se eliminó el componente

4.4.5 Prueba al Caso de Uso: “Gestionar Relaciones”

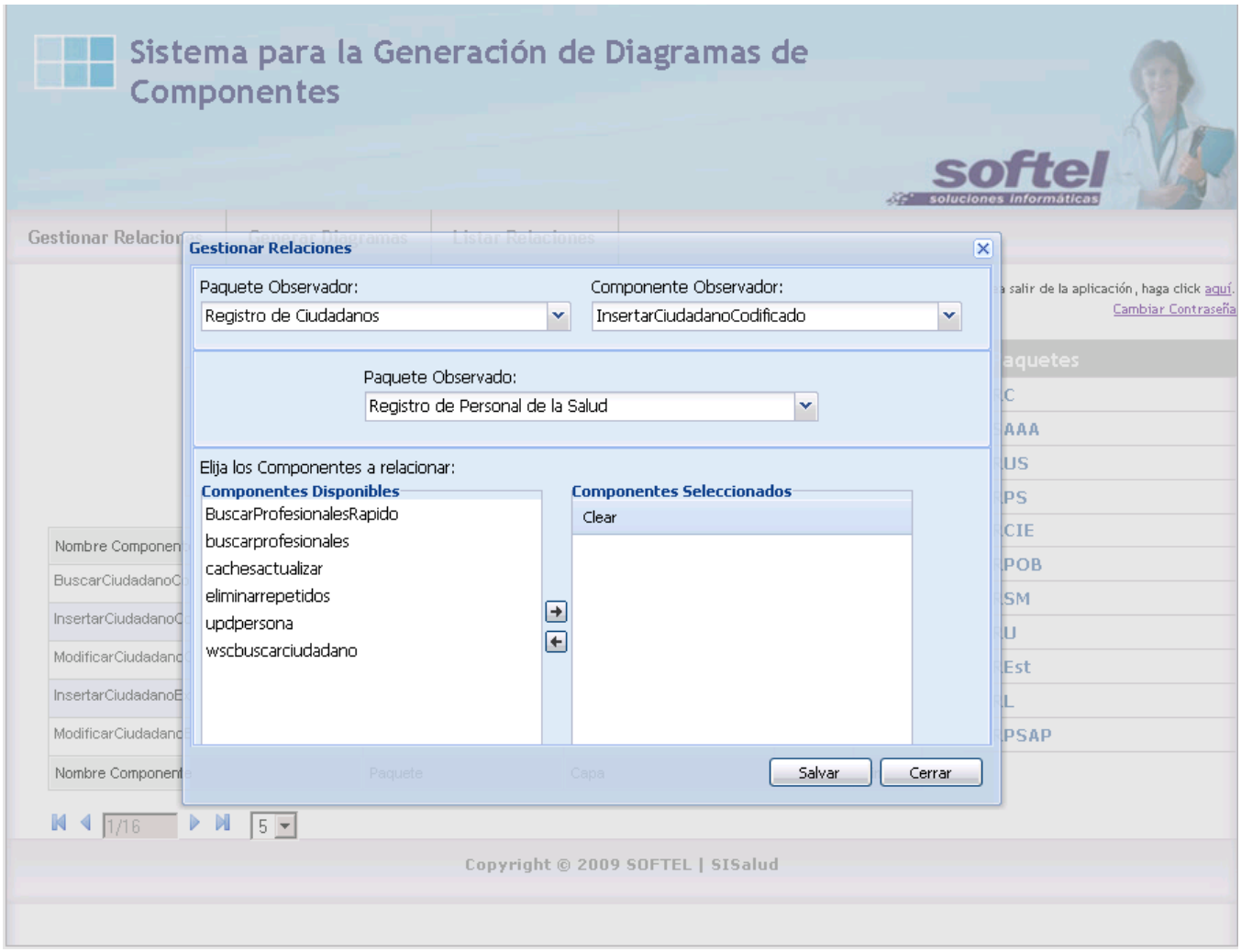


Figura 41: Gestionar Relaciones

Se le aplicó la prueba al caso de uso “Gestionar Relaciones” gestionando las relaciones entre el los paquetes de componentes “Paquete”

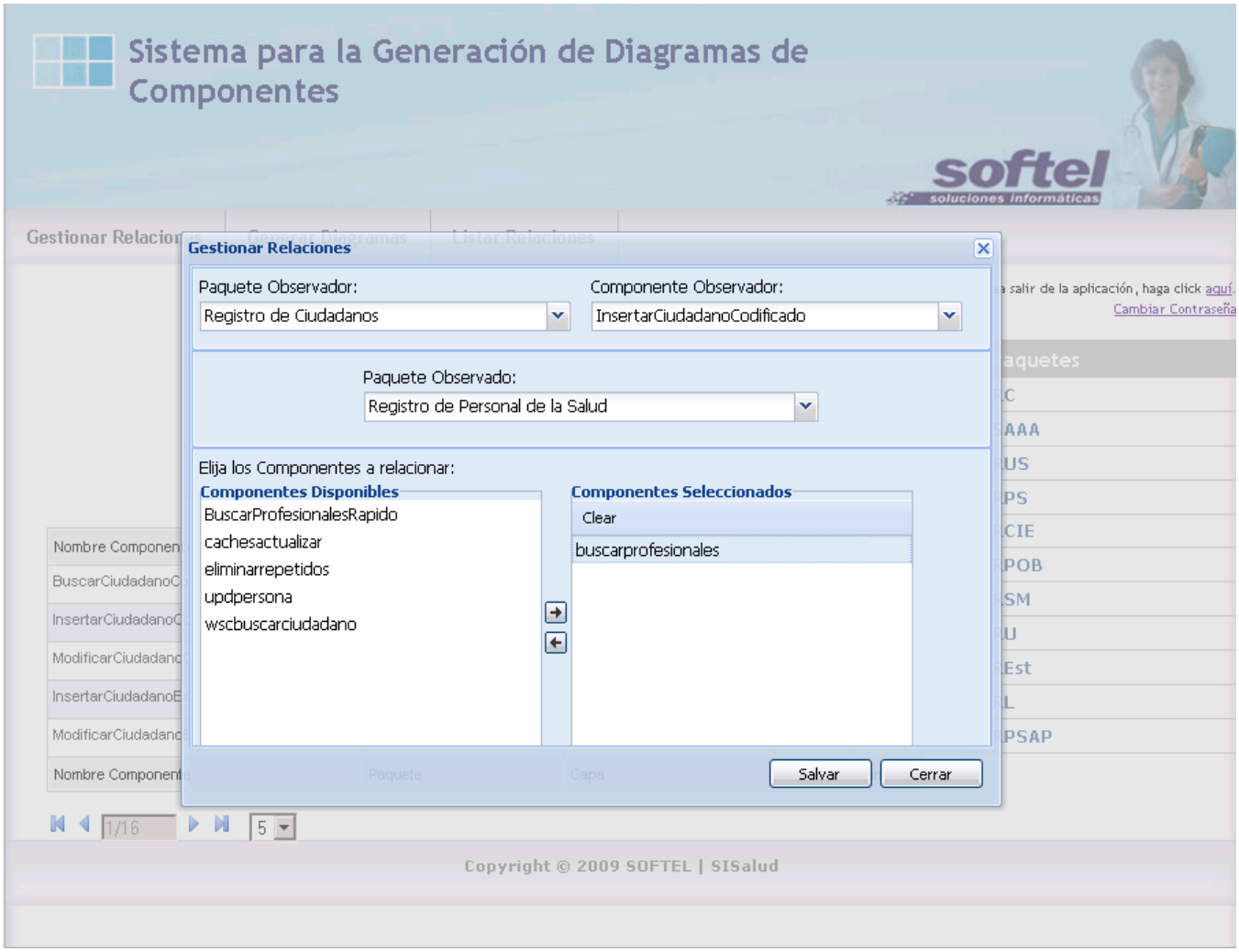


Figura 42: Nueva Relación

4.4.6 Prueba al Caso de Uso: “Generar Diagramas”

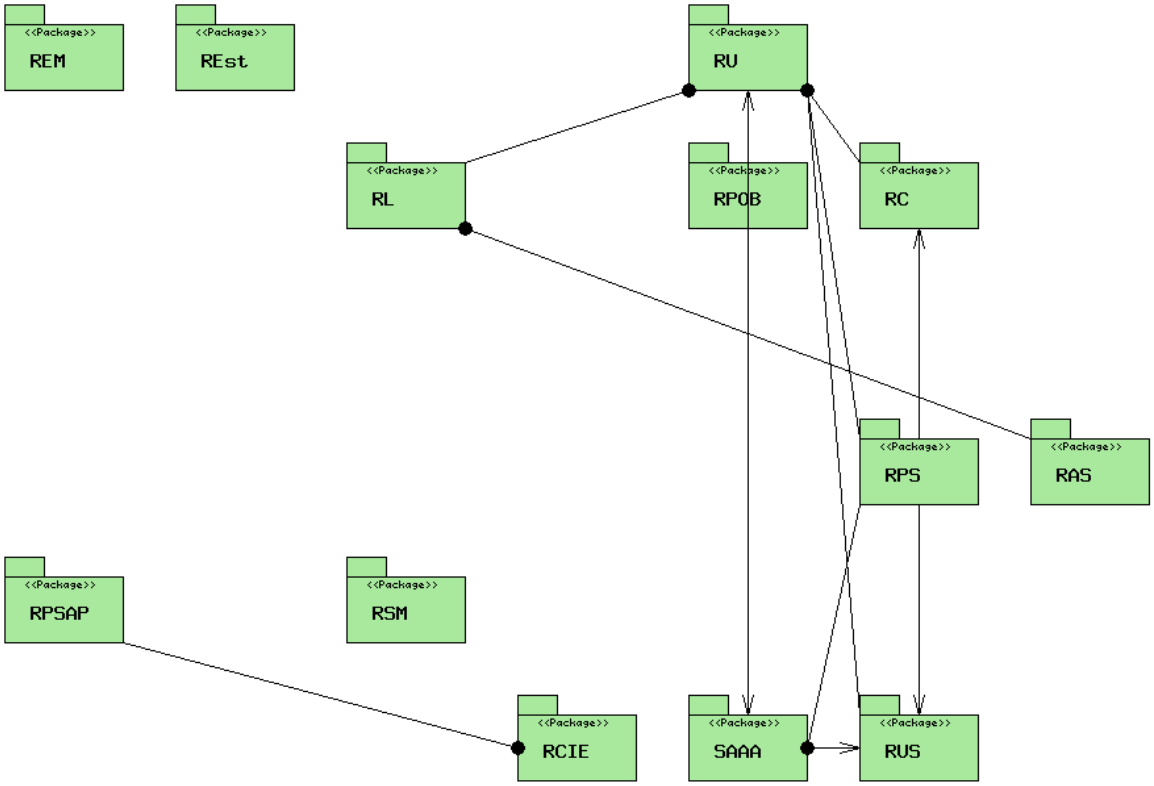


Figura 43: Diagrama Global de Componentes.

Por último se le desarrolló la prueba al caso de uso “Generar Diagramas” específicamente al escenario “Diagrama Global de Componentes” mostrando en este caso de uso las relaciones entre los paquetes de componentes como es mostrado en la Figura 43.

Conclusiones del Capítulo

La implementación es una etapa fundamental para lograr materializar todo el proceso de creación de un software, este no es el final, pero es una parte muy importante para todo el equipo de trabajo involucrado. En este capítulo se presentaron los resultados obtenidos en el presente trabajo, y el código fuente de los métodos más importantes que reflejan las características que se definieron desde

el inicio del proceso de creación de este software, para dar cumplimiento a los requisitos del cliente. Cumpliendo con los patrones y la arquitectura definidos

Conclusiones

Al concluir el trabajo se ha dado cumplimiento al objetivo propuesto, pues se dispone de una aplicación web que garantiza la gestión de la información de los componentes desarrollados para el Sistema de Información para la Salud (SISalud), que puede ser utilizada por la Empresa SOFTEL para el proceso de integración de las aplicaciones informáticas para la salud. Se cumplieron, de igual forma, las tareas de investigación planteadas:

1. Se investigó sobre el estado del conocimiento del tema relacionado con la generación de las Vistas de Implementación.
2. Se modelaron, siguiendo RUP, las Disciplinas de Requerimientos, Diseño e Implementación, teniendo en cuenta la arquitectura definida por el MINSAP, la integración de los componentes del Sistema de Información para la Salud y las tecnologías actuales para el desarrollo.
3. Se implementó la aplicación web “Catálogo de Componentes para la generación de las vistas de Implementación de la Arquitectura de SISalud”.
4. La herramienta apoya el trabajo del proceso de integración de las diferentes aplicaciones informáticas para la salud, ya que ofrece las relaciones de dependencia entre los componentes a partir de la representación de las vistas de implementación.
5. Se despliega la aplicación en la Empresa para su utilización y consulta por parte de los integrantes del Grupo de Integración de Soluciones.

Recomendaciones

Tomando como base la investigación realizada y la experiencia acumulada durante la realización de este trabajo, se proponen las siguientes recomendaciones:

1. Realizar las pruebas de calidad, para certificar la eficiencia de la aplicación desarrollada, con el fin de entregar al cliente un producto de excelencia.

2. Incorporar la aplicación al Portal de Arquitectura MINSAP-MIC, con el objetivo de publicar a la comunidad de desarrolladores del Sistema Nacional de Salud, la información relacionada con las dependencias entre los componentes de SISALUD.

3. Desplegar la aplicación a solicitud de la Dirección Nacional de Informática del MINSAP para consultar la información que brinda y tomar decisiones relacionadas con el proceso de integración.

Referencias Bibliográficas

1. **Cabrera, Mirna Hernandez.** Plataforma para la administración, procesamiento y transmisión de la información en el sistema de salud: SISalud. Ciudad de la habana : s.n.

2. **Ariel Delgado Ramos, Mirna Cabrera Hernadez, Alfredo Rodríguez Díaz.** Estrategia de Informatización del Sistema Nacional de Salud.

3. **Ruz, Fidel Castro.** Discurso pronunciado por el Presidente de la República de Cuba, Fidel Castro Ruz en el acto conmemorativo del aniversario 40 del Instituto de Ciencias Básicas y Pre clínicas Victoria de Girón. *Granma.* 17 de octubre de 2002.

4. *La semilla del desarrollo de la salud pública en Cuba.* **Osa, José A. de la.** Ciudad de la Habana : s.n.

5. *El cuidado de la Salud en Cuba.* Ministerio de Salud Pública, Escuela Nacional de Salud Pública : s.n., 2003.

6. **Ruz, Fidel Castro.** Discurso pronunciado por el presidente de la República de Cuba, Fidel Castro Ruz, en la Tercera graduación del Contingente del Instituto Superior de Ciencias Medicas de la Habana. Teatro “Carlos Marx”. *Granma.* 27 de agosto de 1990.

7. Discurso pronunciado en la Clausura del VI Seminario Internacional de Atención Primaria. *Granma.* 28 de noviembre de 1997.

8. **Ruz, Fidel Castro.** Discurso en el acto de inauguración de obras del extraordinario programa de salud. Teatro Astral. . *Granma.* 7 de abril del 2003.

9. *Presentación de Informatización del Sistema Nacional de Salud. Dirección nacional de Registros Médicos y Estadísticas de Salud.* **Ramos, Dr. Ariel Delgado.** La Habana. Cuba : s.n., 2006.

10. *Presentación Nuevas definiciones para el grupo de trabajo DTA.* La Habana : s.n., 2009.

11. Microsoft Solution Framework. [Online] http://www.microsoft.com/technet/itsolutions/msf/.

12. eXtreme Programming. [Online] http://www.extremeprogramming.org/.

13. **Booch, Ivar Jacobson James Rumbaugh y Grady.** El Proceso Unificado de Desarrollo de Software. Addison Wesley : s.n., 1999.

14. *Ingeniería de Software I. Conferencia 1 Introducción a la Ingeniería de Software.* UCI : s.n., 2007-2008.

15. **Revisado Febrero 2009. URL http://www.uml.org/.** Unified Modeling Language. [Online] http://www.uml.org/.

16. **Booch, Ivar Jacobson James Rumbaugh y Grady.** El Lenguaje Unificado de Modelado. Addison Wesley : s.n., 2007.

17. Magic Draw. [Online] http:// www.magicdraw.com/.

18. Umbrello. [Online] http://uml.sourceforge.net/.

19. Rational Rose. [Online] http://www.ibm.com/software/rational.

20. **Reynoso Carlos, Kicillof Nicolás.** Estilos y Patrones en la Arquitectura de Microsoft. Universidad de Buenos Aires : s.n., 2004.

21. Ingeniería de Software II. Conferencia de Diseño. . [Online] 2008. http://teleformacion.uci.cu/mod/resource/view.php?id=21363.

22. [Online] http://tvdi.det.uvigo.es/~avilas/UML/node41.html.

23. Conferencia 4 Flujo de Implementación, Ingeniería de Software II. [Online] 2007. http://teleformacion.uci.cu/mod/resource/view.php?id=22199.

Bibliografía

1. Delgado Ramos, Ariel. *Informatización Sistema Nacional de Salud.* La Habana, MINSAP, 2003.

2. Gilfillan, Ian. *La Biblia MySQL.* ISBN: 8441515581.

3. Jacobson, Ivan, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software, Volumen I.* La Habana, Editorial Félix Varela, 2004.

4. Jacobson, Ivan, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software, Volumen II.* La Habana, Editorial Félix Varela, 2004.

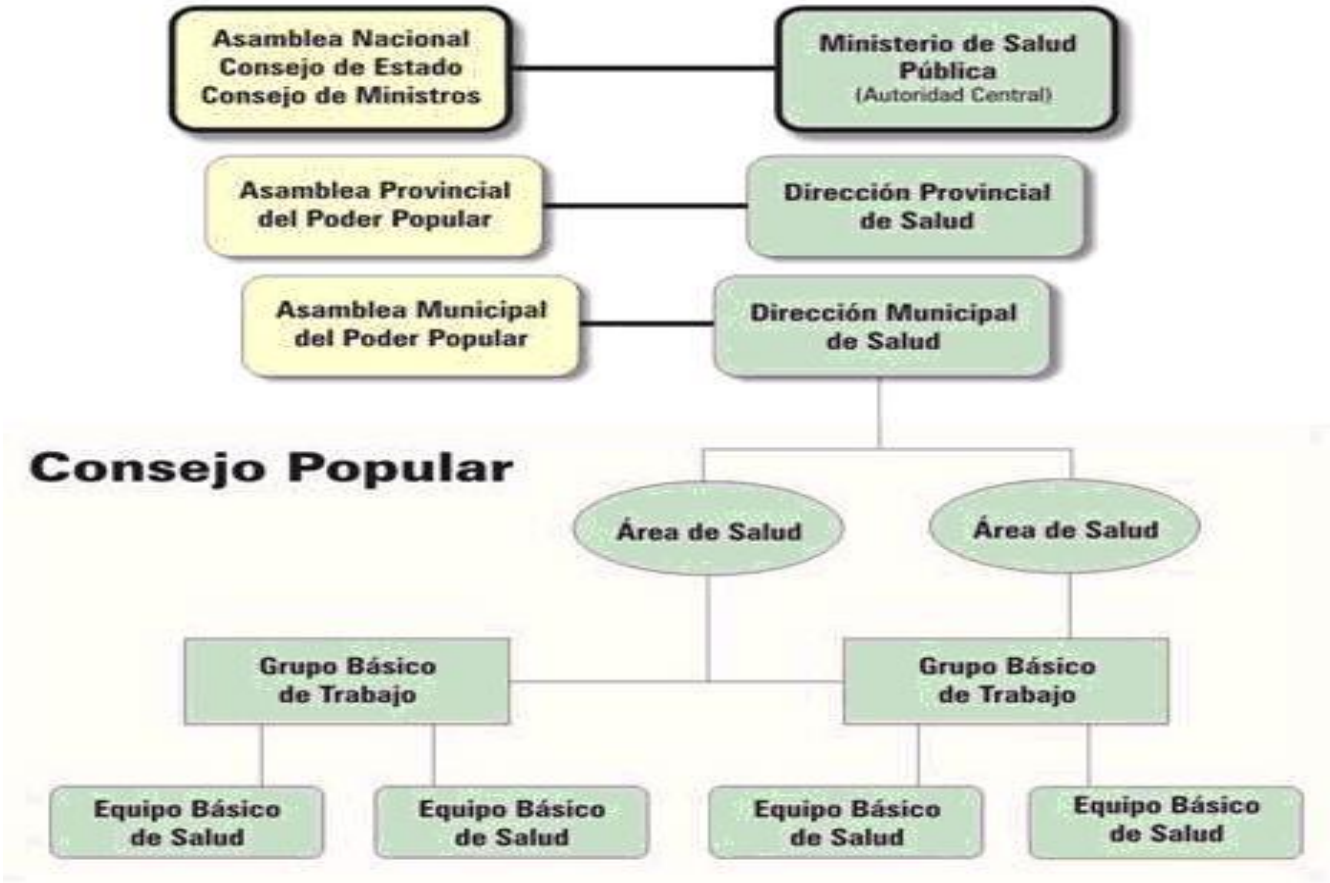
5. Microsoft ® Encarta ® 2009. © 1993-2008 Microsoft Corporation. Reservados todos los derechos.

6. Pressman, Roger. *Ingeniería del Software un enfoque práctico, Parte 1.* La Habana, Editorial Félix Varela, 2005.

7. Pressman, Roger. *Ingeniería del Software un enfoque práctico, Parte 2.* La Habana, Editorial Félix Varela, 2005.

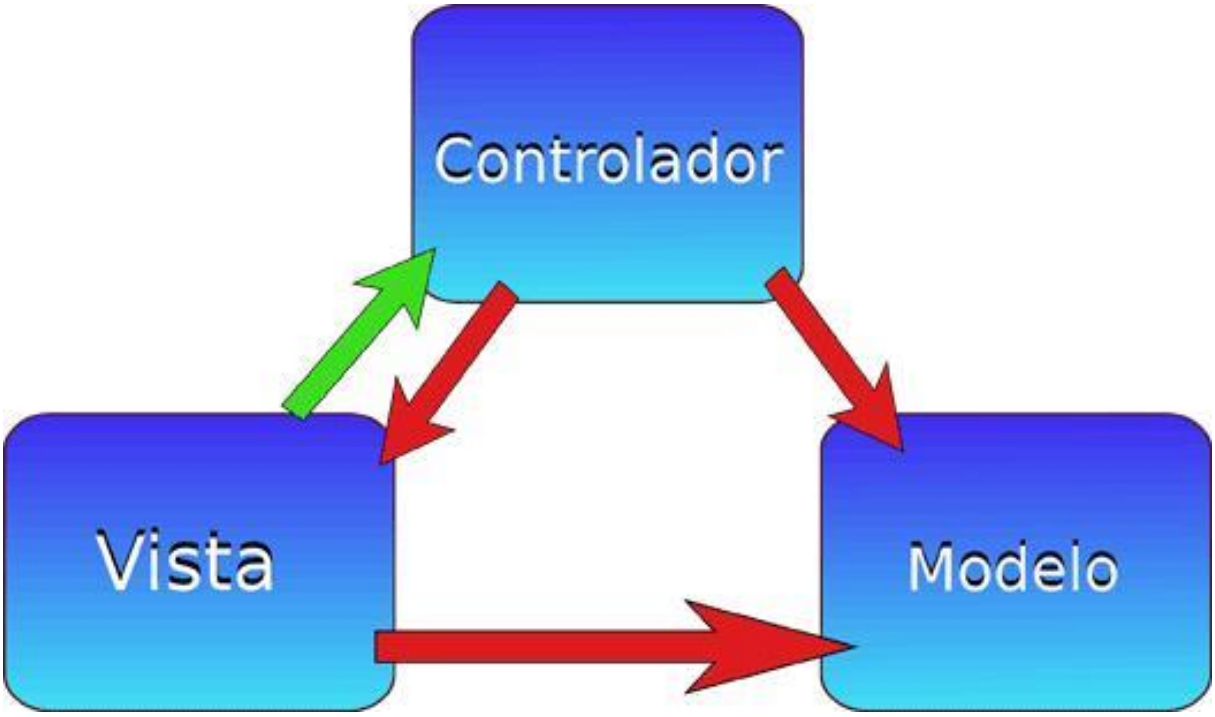
Anexos

Anexo I. Estructura administrativa del Sistema Nacional de Salud.



Anexo II. Estructura. Sistema de Información para la Salud

Anexo III. Modelo Vista Controlador.



Glosario de Términos

Aplicación Web: Es una aplicación informática que los usuarios utilizan accediendo a un servidor Web a través de un navegador o browser. Estas son muy son populares debido a la habilidad para actualizar y mantener la información manipulada sin distribuir e instalar el software en miles de potenciales clientes.

Arquitectura: Conjunto de decisiones significativas acerca de la organización de un sistema de software, la selección de los elementos estructurales a partir de los cuales se componen el sistema. La misma se interesa no sólo por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, comprensión, economía y tecnología, así como por aspectos estéticos.

Base de Datos: Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso.

Caso de Uso: Descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Componente: Parte física y reemplazable de un sistema que se ajusta a, y proporciona la realización de, un conjunto de interfaces.

Dependencia: Relación semántica entre dos elementos, en la cual un cambio en uno puede afectar al otro.

Diagrama: Presentación gráfica de un conjunto de elementos y sus relaciones.

Dominio: Área de conocimiento o actividad caracterizada por un conjunto de conceptos y terminología comprendidos por los practicantes de ese dominio.

Informática: Es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales.

Informatizar: Proceso de aplicar sistemas o equipos informáticos al tratamiento de la información.

Paquete: Mecanismo de propósito general para organizar elementos en grupos.

PDF (del inglés, Portable Document Format, Formato de Documento Portátil): Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

Policlínico: Unidad de salud donde se brindan servicios médicos a una población geográficamente determinada perteneciente al nivel asistencial de Atención Primaria de Salud.

Proyecto: Esfuerzo de desarrollo para llevar un sistema a lo largo de un ciclo de vida.

SOFTEL (Empresa de Soluciones Informáticas): Entidad del Ministerio de Informática y las Comunicaciones (MIC).

Servicio: Unidad de software que encapsula alguna funcionalidad de negocio y proporciona estas a otros servicios a través de interfaces públicas bien definidas.

Unidad de Salud: Centro de trabajo del Ministerio de Salud Pública (MINSAP).

Artefacto: Una parte de la información que es producida, modificada, o usada por un proceso, define un área de responsabilidad, y está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos.

Metodología de desarrollo: Se refiere a los métodos de investigación en una ciencia.

Modelo: Cosa que ha de servir de objeto de imitación. Objeto, construcción u otra cosa con un diseño del que se reproduce más iguales. Esquema teórico de un sistema o de una realidad compleja que se elabora para facilitar su comprensión y estudio.

Requisitos: Conjunto de características que debe tener un producto o servicio para satisfacer las necesidades y expectativas del cliente.

Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

Sistema: Conjunto de procedimientos, normas o métodos integrados para la construcción de un fin.

Web: Red de documentos HTML intercomunicados y distribuidos entre servidores web.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Formulario: Plantilla o página con espacios vacíos que han de ser rellenados con alguna finalidad.

Interfaz: Colección de operaciones que son usadas para especificar un servicio de una clase o un componente.