Universidad de las Ciencias Informáticas
FACULTAD 7

Análisis y Diseño del componente Registro de Servicios.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yaciel Edelio Téllez Toledo

Tutores: Ing. Annia Arencibia Morales

Lic. Yasel Couce Sardiñas

Ciudad de la Habana, Junio de 2008

"Año 50 de la Revolución"

"Si se siembra la semilla con fe y se cuida con perseverancia, sólo será cuestión de tiempo recoger sus frutos".

Thomas Carlyle

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al grupo de desarrollo del Área Temática de Sistemas de Apoyo a la Salud de la Facultad 7 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 19 días del mes de Junio del año 2008.

_____

Yaciel Edelio Téllez Toledo

_____ _____

Ing. Annia Arencibia Morales Lic. Yasel Couce Sardiñas

DATOS DE CONTACTO

Ing. Annia Arencibia Morales (*aarencibia@uci.cu*)**:** Recién graduada de Ingeniera en Ciencias Informáticas en la UCI. Posee Categoría Docente de Profesor Adiestrado. Ha participado en proyectos de desarrollo de Sistemas Informáticos para la Salud desde el año 2005. Imparte la asignatura de Investigación de Operaciones. Actualmente se desempeña como Líder de Proyecto del Área Temática Sistemas de Apoyo a la Salud.

Lic. Yasel Couce Sardiñas (*yaselc@uci.cu*)**:** Graduado de Licenciado en Ciencias de la Computación en el año 2005 en la Universidad Central de Las Villas. Posee Categoría Docente de Profesor Adiestrado y cursa la Maestría en Gestión de Proyectos Informáticos. Ha impartido asignaturas de Sistemas Operativos y Seguridad Informática en la Facultad 7 desde el curso 2005 – 2006. Actualmente se desempeña como Jefe del Área Temática Sistemas de Apoyo a la Salud.

AGRADECIMIENTOS

Agradezco con todo mi corazón a Dios, por ser mi guía, inspiración, pastor, modelo y darme todo su amor.
A mis padres, por estar siempre a mi lado y ayudarme en todo lo que necesite.
A tía Nené, por ser un hombro más donde recostar la cabeza, estar siempre pendiente de mis estudios y necesidades.
A mis segundos padres, Rosa y Jurvis, por apoyarme todo el tiempo y tenerme como un hijo más.
A mi esposa Thelma por ser mi fiel compañera y estar siempre a mi lado.
A mi gordi Jessie, un regalo de Dios.
A mi hermano, abuelos, tíos y primos.
A Lourdes, por estar siempre dispuesta a ayudarme.
A Yasel, por inculcarme siempre el sentimiento de no darse por vencido ante las dificultades.
A Dannier, por su grata cooperación.

RESUMEN

El objetivo de la investigación es diseñar un componente que organice la información correspondiente a los servicios que se planifican en cualquier tipo de entidad. El mismo debe ser flexible ante posibles cambios en la estructura de la información que gestiona, sin necesidad de reimplementarlo.

Este componente orientado a servicios, es capaz de adaptarse a cualquier entorno de trabajo que satisfaga la demanda de soluciones de gestión empresarial. Permite a las empresas unificar las diferentes áreas de productividad. Para llevar a cabo el análisis y diseño del componente, se utilizó Visual Paradigm, que soporta de forma completa la especificación del Lenguaje Unificado de Modelado 2.0 (UML, por sus siglas en inglés).

El desarrollo del componente permite una mejor organización de los servicios que ofrece cualquier organización, integrando los aspectos funcionales de cualquier empresa, para la posterior planificación de los mismos. Tiene un modo de operación dinámico, precisión en los procesos, capacidad de manejar grandes volúmenes de información y rapidez de gestión de la misma. Está concebido para funcionar en un ambiente de integración de sistemas informáticos brindando y consumiendo Servicios Web.

PALABRAS CLAVES

Componente, Servicio, Planificación.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El presente trabajo de diploma constituye una investigación que tiene por objetivo diseñar un componente capaz de estructurar la información de los servicios de cualquier negocio. Este debe ser capaz de tolerar cualquier cambio en su estructura sin la necesidad de una reingeniería, dependiendo del ambiente de trabajo. La investigación es una nueva línea de trabajo trazada por el Área Temática Sistemas de Apoyo a la Salud (SAS), de la Facultad 7, con la idea de crear software flexible y robusto sin la necesidad de la intervención de un programador ante cualquier cambio futuro, haciendo uso de los protocolos y estándares que enmarcan la Arquitectura Orientada a Servicios (SOA).

El Componente Registro de Servicios tiene la capacidad de organizar los servicios que se van a planificar en una entidad. Éste integra y optimiza los recursos de diversas áreas como: recursos humanos, finanzas, operaciones, entre otros. A su vez, aumenta la productividad del negocio y mejora el control de los servicios de cualquier empresa en general.

Una de las premisas básicas del producto es que ofrece una herramienta de uso sencillo e intuitivo, de manera que el usuario pueda trabajar con un entrenamiento mínimo. Además, este componente garantiza la reusabilidad, extensibilidad y mantenimiento de la información que gestiona.

Situación problémica

En Cuba existen muchos procesos y servicios de las entidades estatales que no están informatizados aún, por lo que la mayoría se realizan manualmente o por sistemas con tecnologías obsoletas. Se han desarrollado aplicaciones específicas que resuelven algunas necesidades particulares. Pero todavía existe la necesidad de desarrollar componentes informáticos, que no se limiten a sectores específicos, sino que puedan operar en cualquier sector; sin importar el negocio en que se enmarquen. Hasta el momento, en los productos desarrollados cuando es necesario un cambio en la estructura de la información que manejan, se tiene que acudir al diseño e implementación para lograr la funcionalidad requerida.

La Facultad 7 y en específico SAS, se ha trazado una nueva línea para el desarrollo de productos, basada en componentes y orientada a servicios; con la idea de crear software que sean flexibles ante cualquier cambio en su estructura. Lo que ahorra tiempo y esfuerzo mediante la reutilización de componentes, permitiendo la creación y cambios de servicios de forma incremental.

Luego del análisis realizado, donde se evidencia la necesidad de solucionar la problemática descrita anteriormente, se plantea el siguiente **Problema a resolver**: ¿Cómo mejorar la gestión de los servicios que se manejan en un sistema de planificación de recursos y que el mismo sea flexible ante cambios en su estructura?

Para el desarrollo de la investigación se plantea como **Objeto de estudio** los procesos de planificación de recursos en las empresas.

Se establece como **Campo de Acción** la gestión de los servicios, en los procesos de planificación de recursos en las empresas.

El **Objetivo de la Investigación** es diseñar un componente que gestione los servicios a planificar en una organización.

Para darle cumplimiento al objetivo de la investigación, es necesario desarrollar las tareas que se muestran a continuación:

- Realizar el análisis del estado del arte de los sistemas de planificación de recursos, específicamente el módulo que gestiona los servicios.
- Realizar un estudio de las tendencias y tecnologías actuales para el análisis y diseño del componente.
- Proponer los patrones de arquitectura y diseño a tener en cuenta para el modelado del componente.
- Utilizar las herramientas definidas por SAS para el desarrollo de la aplicación.

Actualidad y necesidad de la investigación

Actualmente las empresas dependen completamente de las Tecnologías de la Información y las Comunicaciones (TICs), de las que esperan flexibilidad, agilidad y eficiencia en los costes. Las entidades están compuestas por un gran número de sistemas interdependientes, heterogéneos y muchas veces redundantes. La rapidez con que las TICs pueden adaptarse a los cambios en las necesidades del negocio no siempre es suficiente (falta de agilidad y flexibilidad). (1)

La estructura actual de las TICs puede hacer que los cambios introducidos cuesten más que los beneficios que aportan (baja eficiencia de costos). Estos retos actuales en el mundo informático se convierten además, en la necesidad de poder crear componentes orientados a satisfacer

requerimientos de negocio o tecnológicos que maximicen la eficiencia del mismo, incrementen la flexibilidad de las TICs y disminuyan los costos.

En el mundo contemporáneo la economía es global y competitiva, por lo cual es indispensable que los administradores de las empresas desarrollen estrategias que les ayuden a satisfacer las necesidades de los clientes, cada vez más exigentes y lograr anticiparse a sus requerimientos, dándoles un trato personalizado.

Las compañías buscan implementar sistemas que manejen todas las áreas del negocio de tal forma que estén integrados. Muchas empresas han buscado nuevas herramientas tecnológicas para poder optimizar los procesos operativos internos para así ahorrar costos y ser más eficientes, lo que tiene como consecuencia un mejor posicionamiento y la atracción o bien conservación de los clientes.

El mercado actual es global y evoluciona rápidamente tal como el mundo de los negocios, creando la necesidad de tener mayor agilidad tanto en los procesos comerciales como en la infraestructura informática. La capacidad de adaptar o crear procesos comerciales de una manera rápida, fácil y económica, es vital para el éxito en los negocios. Para satisfacer las exigencias siempre cambiantes de los clientes y del mercado, las empresas deben poder contar con la flexibilidad necesaria para hacer cambios en su gestión de trabajo, para que no sólo afecten sus operaciones internas sino que también extiendan sus procesos de colaboración a su red de socios de negocio. (2)

Sin embargo, la integración de los sistemas informáticos que las empresas han adquirido a través de los años para compartir datos y ejecutar nuevos procesos comerciales, se ha demostrado tradicionalmente como algo difícil y costoso. Los profesionales a cargo de la tecnología informática citan constantemente la integración de sistemas como uno de los grandes retos que deben enfrentar. Con el mantenimiento de estrictos controles en los costos de la tecnología de la información, la práctica de "desechar y reemplazar" en las inversiones informáticas no siempre es viable. En perspectiva, la solución a los problemas de integración e innovación de los procesos comerciales se logrará con el desarrollo de una arquitectura más flexible sobre la cual las aplicaciones de software y los procesos comerciales puedan fundarse. (3)

Las empresas tratan de dirigir sus entornos informáticos hacia arquitecturas orientadas a servicios empresariales (Enterprise Servicie Oriented Architecture – Enterprise SOA), para permitir un efectivo intercambio de información entre varias instalaciones de software, y reducir los costos de crear y mantener interfaces.

Antecedentes

El Sistema para la Planificación y Balance Material es una aplicación informática desarrollada en la Universidad de las Ciencias Informáticas (UCI) en el 2007. Posibilita la planificación de los materiales gastables de uso médico que serán consumidos en las entidades de salud del país. Permite que la información fluya de forma rápida entre los distintos niveles de organización en que se encuentra estructurado el Sistema Nacional de Salud (SNS). Facilita el conocimiento de la cantidad de materiales a utilizar en cada entidad de salud del país. Así como, realizar los cálculos pertinentes que permiten conocer el importe de la cantidad de materiales que el MINSAP, como organismo rector, debe garantizar para el buen funcionamiento del sistema de atención médica en el país.

Esta aplicación realiza la planificación basada en los servicios médicos que se brindan en las entidades del SNS. No tolera cambios en la estructura de la información que maneja y está diseñada para funcionar solamente en este tipo de entidades. Por todo lo anterior, surge la necesidad de desarrollar componentes informáticos flexibles que sean capaces de adaptarse al modo de operar de cualquier institución.

Aportes prácticos esperados

Al concluir la investigación, se espera como resultado que el componente diseñado sea capaz de controlar la información de los servicios que brinda una entidad determinada. Debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias y responder ágilmente a los cambios.

Concluido el ciclo completo de desarrollo de este producto, se obtendrán los siguientes beneficios una vez desplegado el componente en cualquier negocio:

- Mejor organización de los servicios, debido a la forma en que se encuentran estructurados.
- Obtención de un producto que responde ágilmente ante cambios efectuados en la estructura de la información que maneja.
- Obtención de un producto altamente competitivo y basado en patrones de arquitectura y diseño.
- El componente puede ser utilizado en cualquier organización que desee estructurar sus servicios e información correspondiente.
- Permite que otros sistemas consuman los servicios que el componente gestiona.

El trabajo está estructurado en 3 capítulos:

El Capítulo 1 trata sobre la Fundamentación teórica, se estudia el estado del arte, los sistemas existentes vinculados al campo de acción, las técnicas, tecnologías y metodologías usadas en las que se apoya la solución del problema. Menciona las herramientas que se utilizan para realizar el análisis y diseño de este componente y las que se deben emplear en el desarrollo del mismo.

El Capítulo 2 se refiere a las características del sistema y objeto de informatización. Se presenta la propuesta del sistema. A través del Modelo de Dominio se describen los procesos del negocio. Se especifican los requisitos funcionales y no funcionales del componente y se modela el Diagrama de Casos de Uso del Sistema.

El Capítulo 3 presenta los diagramas de clases de análisis y diseño del sistema, obteniéndose los diagramas de interacción para el diseño, así como el diagrama de clases persistentes y despliegue. Se realiza la descripción expandida de los casos de uso del sistema. Se representa la integración del componente con otros sistemas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

En este capítulo se hace un esbozo del estado del arte de los sistemas existentes vinculados al campo de acción a nivel nacional e internacional; así como tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad.

1.2 Definiciones previas

La definición de “componente” puede plantear un largo debate, puesto que no existe consenso al respecto. Existen varios puntos de vista relacionados al tema, de los cuales se muestran algunos:

Agustín Cernuda del Río en su tesis de doctorado, "Sistema de verificación de componentes software", define los componentes de software como "porciones ejecutables de distribución independiente, con una interfaz claramente definida, preparadas para ser utilizadas por terceros, que no tienen un estado persistente y que interactúan para formar un sistema funcional". (4)

Existen otras definiciones respecto al tema resumidas por Szyperski que para una mejor comprensión se divide en cuatro partes: (5)

- Definición I: Un componente software es una unidad de composición con interfaces especificadas y con dependencias contextuales explícitas.
- Definición II: Una interfaz es un conjunto de operaciones con nombre que pueden ser invocadas por los clientes.
- Definición III: Las dependencias contextuales son especificaciones de lo que el entorno en el que se usa el componente deberá proporcionar, de modo que los componentes puedan funcionar.

Szyperski aporta también características básicas que a su juicio tiene un componente, haciendo énfasis en aspectos comerciales:

Definición IV: Las propiedades características de los componentes son:

- Es una unidad de despliegue independiente.
- Es una unidad de composición para terceros.

- No tiene estado persistente.

De las definiciones anteriormente analizadas, la más cercana al concepto de componente utilizado en esta investigación es la definición de Szyperski.

Otro de los términos que se manejan en esta tesis es el de servicio, definido como la actividad que lleva a cabo una entidad y que generalmente es intangible. La prestación de un servicio puede implicar, por ejemplo:

- Una actividad realizada sobre un producto tangible suministrado por el cliente (por ejemplo, la reparación de un automóvil).
- La entrega de un producto tangible (por ejemplo, la entrega de materiales docentes a los estudiantes).

En otras palabras, un servicio es la acción de llevar a cabo una actividad determinada en un negocio que busca responder a las necesidades del cliente.

Otro de los conceptos claves que se manejan es el de Servicio Web, definiéndose según el glosario de términos del Rational Rose como "un recurso de software con una especificación de servicio externalizado. Esta especificación de servicio está disponible para búsquedas, enlaces e invocación por parte de un cliente de servicio. El proveedor de servicio realiza la implementación de la especificación de servicio y también ofrece los requisitos de calidad de servicio al cliente de servicio. Los servicios se regirán por políticas declarativas y, por tanto, darán soporte a un estilo de arquitectura reconfigurable dinámicamente".

1.3 Estado del arte a nivel internacional

La presentación del estado del arte incluye una breve reseña de los denominados Sistemas de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés) a nivel internacional y nacional, profundizando en la gestión de los servicios específicamente.

En la actualidad, la implantación de sistemas de gestión, que sirven de soporte para la realización de una administración eficiente, ha adquirido un auge significativo en el mercado empresarial, ya que las empresas buscan maximizar sus beneficios y minimizar sus costos. Los sistemas ERP son de gran utilidad, pues brindan el soporte necesario para alcanzar los objetivos deseados por cualquier empresa independientemente de su tamaño. (6)

Los ERP son sistemas que integran todos los aspectos funcionales de la empresa: gestión comercial, gestión financiera, gestión de entradas / salidas, gestión de producción, control de almacenes, etc. De esta forma el ahorro de tiempo y la minimización de errores son máximos, al no existir aplicaciones diferentes entre las cuales transferir datos, proceso que en muchos casos resulta imposible. (7)

Estos sistemas deben ser parametrizados y adaptados para responder a las necesidades específicas de cada organización. Una vez implementado un ERP, éste permite a los empleados de una empresa administrar y planificar los recursos de todas las áreas, simular distintos escenarios y obtener información consolidada en tiempo real. (8)

Entre los principales proveedores del mercado internacional de los ERP se encuentran: *SAP, Baan, Peoplesoft, JD Edwards, Oracle, Progress Software* y *QAD.* (9)

Según la experiencia de estas empresas las ventajas de las soluciones ERP son: menores costos, velocidad combinada con flexibilidad, toma de decisiones y ejecución mejorada, así como también seguridad en las plataformas que utilizan desarrollo de pronósticos acertados que darán una buena dirección a la empresa.

1.3.1 Baan

Baan es un software popular ERP que fue creado por Jan Baan en 1978, en Holanda; para informatizar varias actividades de una entidad, y lograr una mejor planificación de recursos. Es posible ajustar procedimientos y servicios de un sistema para especificar los requerimientos de una empresa, acercándose a sus necesidades. (10)

Bann tiene varios módulos, entre ellos: finanzas, manufacturas, servicios, conectividad, proyecto, comercio electrónico, entre otros. Este tipo de software está enfocado a medianas empresas.

Esta aplicación estructurada en forma de paquetes integrados, ayuda a la empresa a gestionar rigurosamente cada una de las áreas en las que se implique. La forma de gestionar los servicios es a través de una interfaz nombrada "Gestión de servicios" la que permite realizar la planificación de los recursos a usar para ofrecer los servicios del negocio donde se implante el sistema.

Entre las ventajas que ofrece, se encuentran: la reducción de duplicación de la información, aumento de productividad del negocio al tener estructurado mejor los servicios, incrementa la eficiencia, la información de los servicios que brinda están disponibles para la organización de forma rápida y ágil;

mejorando la administración de los mismos. Como desventajas se mencionan: el costo, el tiempo relativamente grande para ponerlo en explotación en cualquier entidad, las actualizaciones del software por parte del productor se torna difícil; pues no tolera cambios en su forma de implementación. (11)

1.3.2 JD Edwards

J.D. Edwards es una compañía de software fundada en marzo de 1977, en Denver (Colorado), por Jack Thompson, Dan Gregory y Ed McVaney. Tuvo éxito creando un programa de contabilidad para los miniordenadores Sistema/34 y Sistema/36 de IBM, centrándose en los miniordenadores Sistema/38 a mediados de los años 1980 hasta la aparición de los sistemas AS/400. (12)

La compañía fue añadiendo funciones, su software de contabilidad se convirtió en ERP; una aplicación independiente de la plataforma que en 1996 se llamó OneWorld.

En junio de 2003, el consejo de administración de J.D. Edwards accedió a la oferta de adquisición de PeopleSoft, completándose la adquisición en julio. OneWorld se añadió a la línea de productos de PeopleSoft. A finales de 2004, PeopleSoft fue adquirida a su vez por Oracle. (13)

Entre los productos orientados a servicios se encuentra: *JD Edwards EnterpriseOne*.

JD Edwards EnterpriseOne es un conjunto completo de aplicaciones comerciales modulares, previamente integradas, específicas de un sector, diseñadas para una rápida implementación y fácil administración de servicios de planificación. Es idealmente adecuado para las organizaciones que fabrican, construyen, distribuyen, planifican y brindan servicios o administran productos o activos físicos. Es ideal para la planificación de recursos de cualquier empresa de cualquier tamaño. Desarrolla componentes orientados a servicios que tienen como ventaja: (14)

- Fiabilidad total.
- Costo total de propiedad muy bajo: Los componentes JD Edwards World no precisan personal informático dedicado. Las aplicaciones funcionan como un conjunto unificado y sincronizado, estrechamente integradas y previamente agrupadas en una sola base de datos, lo que reduce los costos de implementación y complejidad.

- Solución empresarial completa: Los componentes JD Edwards World proporcionan funcionalidad avanzada, con procesos empresariales específicos totalmente integrados que permiten a las empresas pequeñas agilizar sus operaciones y aumentar la productividad.

Como desventajas se mencionan. (15)

- Tiempo significante para los procesamientos de la información
- Resistente ante cambios en su estructura.

Todos estos sistemas ERP tienen grandes ventajas de forma general en lo que a la gestión de servicios se refiere, entre las que se mencionan: (16)

- Optimización de los procesos empresariales.
- Acceso a la información confiable, precisa y oportuna.
- La posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias.
- Reducción de tiempos y costos de procesos.

La desventaja fundamental que se puede señalar a modo general, es la imposibilidad de adaptación total del sistema a los procesos tradicionales de la empresa que lo adquiere

A partir de aquí se desprenden otras series de desventajas que son ampliamente discutidas en la bibliografía en línea que se puede encontrar en Internet. No se puede olvidar tampoco el freno al cambio que hacen los empleados al resistirse a modificar sus rutinas de trabajo, necesitando también adquirir adiestramientos en Informática si no lo tenían antes.

1.4 Estado del arte a nivel nacional

La Empresa de Soluciones Informáticas para la Salud (SOFTEL) es una empresa fundada en marzo de 1986 que se establece en el mercado cubano. Se destaca en el desarrollo de aplicaciones informáticas, prestación de servicios informáticos en diversos sectores como: el turismo, la salud y la gestión empresarial. (17)

En el 2003, reorienta su trabajo y se especializa en Soluciones Informáticas para la Salud, para lo cual dispone de profesionales con experiencia en diseño, implantación y gestión de estas soluciones. Combina la experiencia en el desarrollo e integración de soluciones informáticas para el sector de la

salud, con la aplicación de modernas tecnologías. Ha desarrollado varios componentes informáticos para soluciones específicas, pero según investigación, no ha elaborado ningún producto con las características como las que se exponen en este trabajo de diploma.

La UCI, inmersa en la producción nacional de software, es la encargada de informatizar la mayor cantidad de organismos posibles para elevar la calidad de los procesos empresariales nacionales. La universidad ha desarrollado componentes a sectores específicos y con características peculiares. Citando algunos ejemplos:

En la Facultad 10 se está elaborando un sistema encaminado a eliminar todos los posibles errores o inconvenientes que presenta la Intranet de la universidad, en lo que a la prestación de servicios se refiere. Esta aplicación tiene como objetivo brindar servicios de gran estabilidad, rapidez, robustez y que además puedan integrarse con otros sistemas. Tiene como característica, que es diseñada para servicios predefinidos como: transporte, reservaciones de gas, reservaciones recreativas, para poder llevar a cabo la planificación de los mismos. Esta nueva versión de la Intranet no resuelve el problema que se plantea sobre la flexibilidad, ya que es para tipos de negocios particulares.

Otro proyecto también de la Facultad 10 es el Visión Portal, en contrato con la compañía petrolera PDVSA. El mismo constituye una nueva Intranet de servicios para esta compañía. Es una aplicación para este sector en específico. Permite estructurar los servicios que se brindan en esta entidad y poder llevar a cabo procesos de planificación, pero no cuenta con la flexibilidad requerida para funcionar en otro negocio. Está diseñada para integrarse con otros sistemas externos para intercambiar información. La estructura sobre la cual estarán registrados los servicios ya está predefinida.

Los proyectos anteriormente mencionados se basan en una arquitectura SOA, por las ventajas que ésta ofrece en un ambiente de integración de componentes, la flexibilidad y reutilización de procesos para estructurar un sistema de información.

1.5 Técnicas, Tecnologías y Metodologías usadas en la actualidad

Para los desarrolladores de software, ha sido un objetivo durante décadas el encontrar procesos o metodologías predecibles y repetibles que mejoren la productividad y la calidad.

1.5.1 Arquitectura Orientada a Servicios (SOA)

SOA es un modelo de componentes que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas servicios, a través de interfaces y contratos bien definidos. La interfaz se define de forma neutral, y debería ser independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme y universal. (18)

Las arquitecturas SOA están motivadas por la creciente necesidad de los negocios de responder con rapidez a los cambios en el entorno comercial en que se desenvuelven, lo que conduce a cambiar sus sistemas tecnológicos con esa misma rapidez; para lograrlo, es necesario que los componentes de la infraestructura sean reutilizables y poco interdependientes, que permitan una rápida reconstrucción de los mismos.

Los elementos básicos que conforman una arquitectura SOA son: (19)

- Proveedores de servicios[1].
- Consumidores de servicios[2].
- Bus Empresarial de Servicios (BES).

El BES es un elemento que permite la integración entre los servicios y distintos sistemas, así como el enrutamiento y transformación de mensajes distribuidos. Aísla a los servicios de la complejidad tecnológica de interconexión entre los componentes. Permite a los servicios consumir a otros sin saber la localización física del proveedor. Se basa en la mejor práctica de los patrones de diseño.

[1] Entidades de software que implementan una especificación de servicio.
[2] Entidades de software que llaman a un proveedor de servicios. Tradicionalmente se le llaman “clientes”. Puede ser una aplicación final u otro servicio.

SOA define las siguientes capas de software: (20)

- Aplicativa básica: sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- Exposición de funcionalidades: donde las funcionalidades de la capa aplicativa son expuestas en forma de Servicios Web.
- Integración de servicios: facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.
- Composición de procesos: define el proceso en términos del negocio y sus necesidades, y varía en función del negocio.
- De entrega: donde los servicios son desplegados a los usuarios finales.

Con SOA, las empresas pueden hoy integrar aplicaciones de manera más sencilla y desplegar nuevos servicios con mayor rapidez, gracias a la reutilización del software y a la independencia de las aplicaciones respecto a la infraestructura y plataforma tecnológica. Se trata de un enfoque que, ayudado por los estándares y los Servicios Web, se perfila como una de las tendencias en las tecnologías de la información más importantes para el presente y futuro cercano. (21)

Las ventajas de la filosofía SOA son múltiples y la mayoría de ellas se derivan de dos factores. En primer lugar, este enfoque hace posible la reutilización a gran escala del software, que implica además, sobre la calidad, costos y facilidad de despliegue. Independiza las aplicaciones de la infraestructura y plataforma tecnológica.

Al contrario de la Arquitectura Orientada a Objetos, SOA está formada por servicios de aplicación con acoplamiento débil y altamente interoperable. También se diferencia de una arquitectura cliente/servidor tradicional en que está orientada a procesos y enfocada al cambio. De esta forma, gracias a una arquitectura SOA, las empresas pueden reconfigurar rápidamente sus recursos de las TIC sin necesidad de realizar una integración profunda, lo que permite liberar recursos para abrir espacio a la innovación

A modo de resumen, los beneficios que puede obtener una compañía que adopte SOA son:(22)

- Mejora en los tiempos de realización de cambios en los procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.

- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).

1.5.2 Metodología de desarrollo: RUP

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, se obtienen clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

El proceso unificado es una versión libre y abierta del proceso iterativo e incremental de ingeniería de software propuesto por Jacobson, Booch y Rumbaugh (los “tres amigos”) en su libro “El proceso unificado de desarrollo de software”, publicado por Addisson-Wesley en 1999. El lenguaje para especificar y modelar en el RUP es UML, por lo cual puede apoyarse en cualquier herramienta que soporte UML. (23)

Esta metodología es usada por la Facultad 7 y SAS, aunque existen otras tales como: Extreme Programming XP y Microsoft Solution Framework (MSF).

RUP proporciona al equipo de proyecto, procedimientos y herramientas que promueven las siguientes prácticas:(24)

- Desarrollos iterativos.
- Uso de arquitectura basada en componentes.
- Verificación continua de la calidad del software.
- Gestión de cambios y requisitos.
- Implementa las mejores prácticas de desarrollo de software.

RUP es un proceso de ingeniería de software bien definido y estructurado, que provee un marco de proceso adaptable a las necesidades y características de cada proyecto específico. Sus tres características fundamentales:

- Dirigido por casos de uso.
- Proceso centrado en la arquitectura.
- Iterativo e incremental.

Que sea iterativo e incremental, reduce los riesgos basado en la retroalimentación temprana .Las pruebas continuas e iterativas promueven una mejor evaluación del estado del proyecto. Se pueden administrar mejor los cambios.

1.5.3 Herramienta CASE: Visual Paradigm 6.0

Es una herramienta de desarrollo que se integra al IDE de Eclipse. Diseñada para desarrollar software con Programación Orientada a Objetos (POO), busca reducir la duración del ciclo de desarrollo, brindando ayuda a arquitectos, analistas, diseñadores y desarrolladores. Es un producto distinguido que facilita la organización de los diagramas, su misión es diseñar, integrar y desplegar las aplicaciones de la empresa y sus bases de datos subyacentes. La herramienta ayuda al equipo de desarrollo de software a agilizar el modelado del software, aumentando al máximo y acelerando el trabajo en equipo y las contribuciones individuales. (25)

La herramienta CASE Visual Paradigm posibilita:(26)

- Generación de código (PHP).
- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Análisis de textos.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.

Básicamente, Visual Paradigm es una herramienta fácil de utilizar y que apoya las últimas versiones de UML. Éste surge impulsado por la necesidad de métodos más ágiles de desarrollo de software.

1.5.4 Lenguaje de modelado: UML

UML es el lenguaje de modelado más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el Grupo de Administración de Objetos (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como: procesos de negocios, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (27)

Es importante remarcar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema, documentar y construir.

La versión que usa este trabajo es la 2.0 debido a las ventajas que ofrece con respecto a su anterior (1.0). En UML 2.0 se definen una serie de diagramas adicionales a los establecidos en su versión anterior. Compárese las figuras que se muestran a continuación:

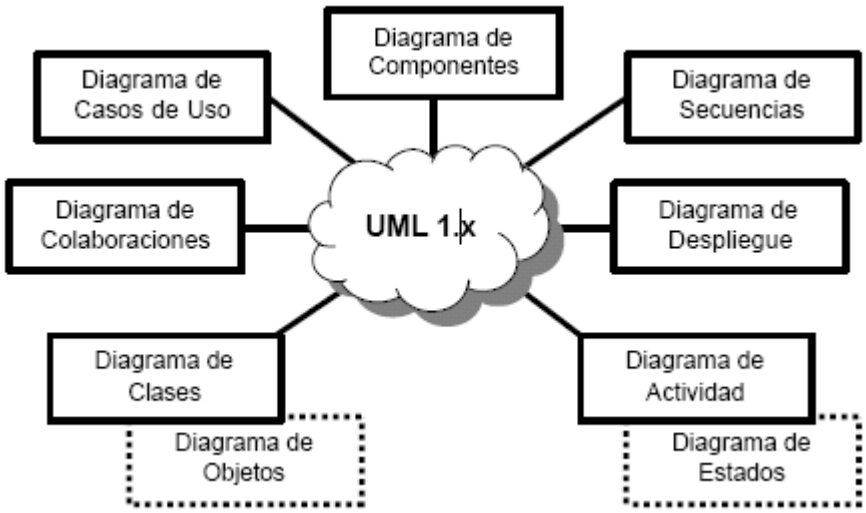


Fig.1 Diagramas de UML 1.0

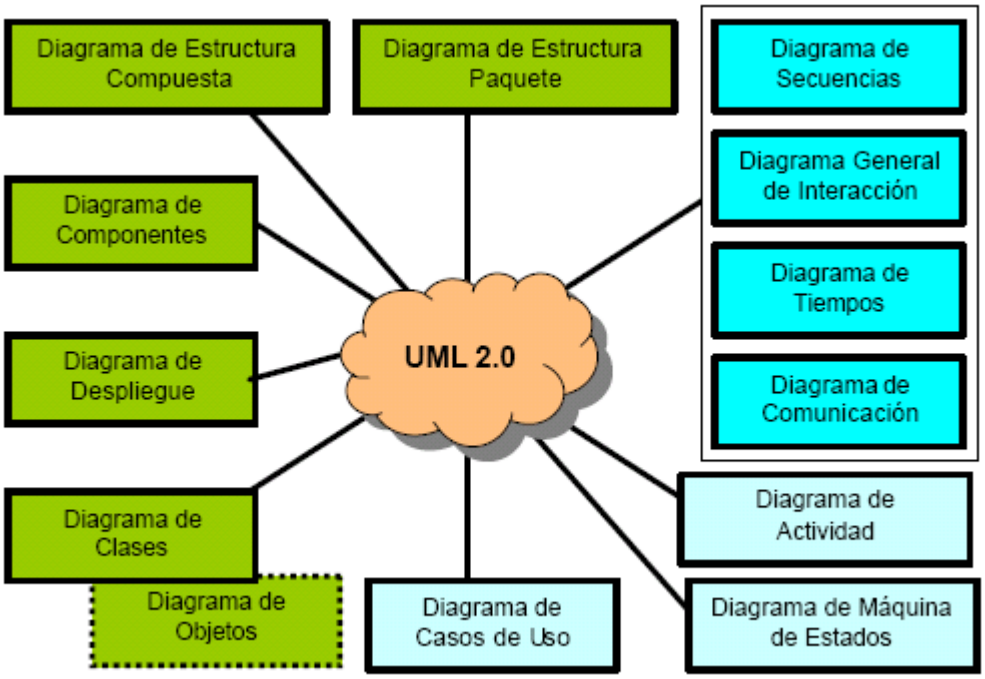


Fig.2 Diagramas de UML 2.0

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados.

La nueva versión permitirá a los fabricantes de herramientas CASE, proporcionar a los analistas, arquitectos y desarrolladores, herramientas cada vez más potentes; que les permitan aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código; reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

1.5.5 Patrón de arquitectura : Modelo Vista Controlador (MVC)

Los patrones de software son soluciones reutilizables a los problemas que ocurren durante el desarrollo de un sistema de software. Éstos proporcionan un proceso consistente o diseño que uno o más desarrolladores pueden utilizar para alcanzar sus objetivos. También proporcionan una arquitectura uniforme que permite una fácil expansión, mantenimiento y modificación de una aplicación.

Las aplicaciones Web pueden desarrollarse utilizando cualquier arquitectura. La arquitectura del patrón MVC es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica.

El patrón MVC es un patrón de arquitectura de software en el cual todo el proceso está dividido en tres capas, típicamente estas capas son el Modelo, la Vista y el Controlador. (28)

El *Modelo* incorpora la capa del dominio y persistencia, es el encargado de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto, XML[3], registro, etc.). Se refiere a la lógica del negocio o servicio y los datos asociados con la aplicación.

La *Vista* se encarga de presentar la interfaz al usuario, en sistemas Web, esto es típicamente HTML, aunque pueden existir otros tipos de vistas. En la vista, sólo se deben realizar operaciones simples. Es la encargada de la presentación de los datos.

El *Controlador* es el que escucha los cambios en la vista y los envía al modelo, este último retorna los datos a la vista. Es el que atiende las peticiones y componentes para la toma de decisiones de la aplicación.

El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que separa los datos y lógica de negocio de la lógica de presentación. A continuación se muestra un esquema de este modelo:

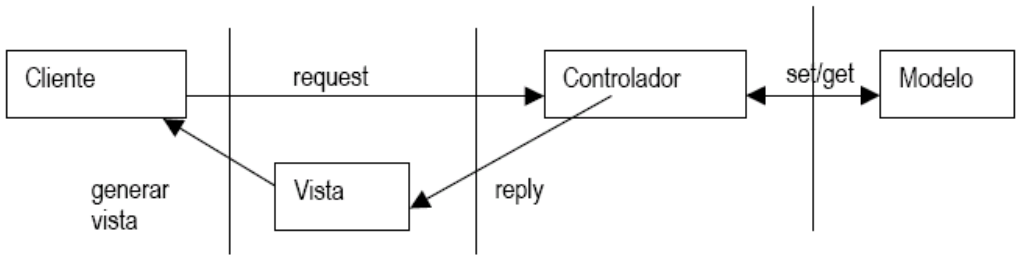


Fig.3 Esquema del Patrón MVC.

[3] Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML, pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones, Así como estructurar, almacenar e intercambiar información.

1.5.6 Patrón de diseño : Composite

El patrón Composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Compone objetos en una estructura jerárquica, donde los objetos compuestos se tratan de forma similar a los objetos simples. (29)

Este patrón describe cómo usar una composición recursiva para no tener que distinguir entre componentes simples y compuestos. Tiene como propósitos:(30)

- Manipular todos los objetos contenidos en el árbol de forma uniforme, ya que todos ellos poseen una interfaz común definida en la clase raíz.
- Permitir a los clientes tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva.
- Construir objetos de complejidad mayor mediante otros más sencillos de forma recursiva, formando una estructura similar a un árbol.

Composite es aplicable cuando:

- Se pretende representar una jerarquía recursiva de objetos.
- Se pretende que los clientes no distingan las diferencias entre objetos simples y compuestos.

Este patrón de diseño posibilita: (31)

- A las clases cliente utilizar tanto a los objetos compuestos como a los individuales de igual forma.
- Definir una jerarquía de objetos hoja y objetos compuestos que se van componiendo de forma recursiva.
- Simplificar la interacción de los clientes, ya que trata los objetos hoja y los compuestos de la misma forma.
- La inclusión de nuevas clases hoja o clases compuestas no modifica la estructura anterior ni el código del cliente.
- Favorecer la extensibilidad, ya que es muy fácil añadir nuevos tipos de componentes, tanto simples como compuestos.
- Hacer el diseño más general.

A continuación se muestra un esquema de la estructura de este patrón:

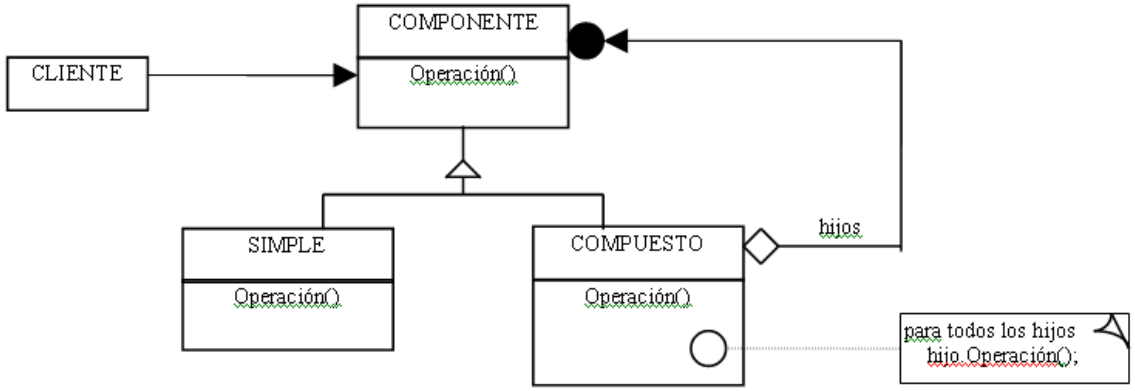


Fig.4 Esquema del Patrón Composite.

Cliente:

- Clase que crea todos los objetos.
- Maneja los objetos que forman parte del Compuesto como Componente.

Componente:

- Clase abstracta común a todos los objetos.
- Define los métodos u operaciones que pueden realizar tanto los objetos individuales como los compuestos.
- Declara la interfaz de todos los objetos de la composición.
- Declara una interfaz de acceso y manipulación de los componentes hijos.

Simple:

- Hoja del árbol de objetos que representa a las clases individuales

Compuesto:

- Define el comportamiento de los componentes compuestos.
- Almacena a los hijos.
- Implementa las operaciones de manejo de los componentes hijos.

Composite desarrolla una forma flexible de crear jerarquías en estructura de árbol de una complejidad arbitraria, permitiendo a la vez que todos los elementos de la estructura funcionen con una interfaz uniforme.

1.5.7 Framework: Symfony

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos patrones utilizados para resolver tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Symfony es un framework diseñado para optimizar el desarrollo de las aplicaciones Web creado con PHP 5. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (32)

Symfony se diseñó para que se ajustara a los siguientes requisitos: (33)

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía que funciona correctamente en los sistemas Windows).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la Web.
- Preparado para aplicaciones empresariales y es adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

Este framework toma lo mejor de la arquitectura MVC y la implementa de forma tal que el desarrollo de la aplicación sea rápido y sencillo. El flujo de trabajo de Symfony es el que se muestra en la siguiente figura:

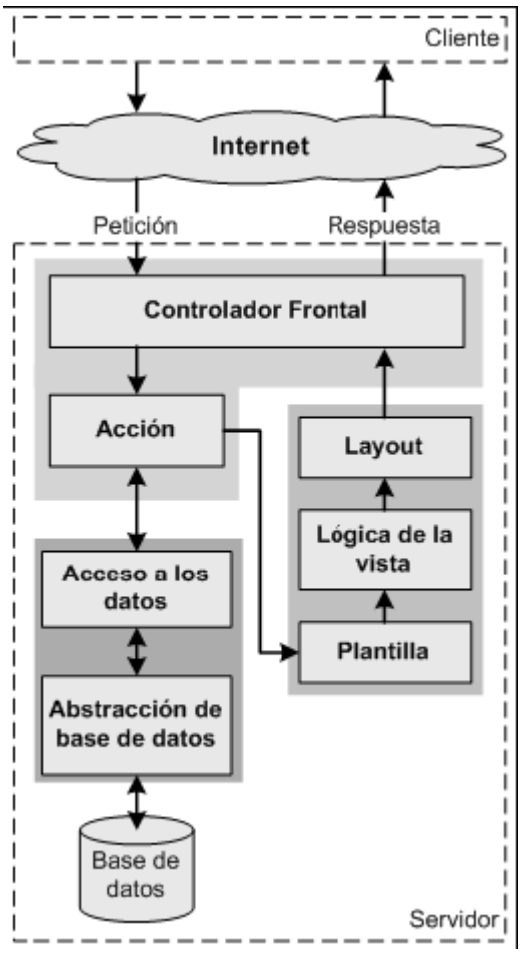


Fig.5 Flujo de trabajo de Symfony.

El uso de un framework que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador. Aplicar el patrón MVC a una aplicación resulta bastante útil además de restrictivo.

Symfony facilita el mantenimiento más sencillo de las aplicaciones, la encapsulación, y mayor seguridad en el acceso a la base de datos. Por tales razones, se propone este framework como una de las herramientas a usar para simplificar el desarrollo de este componente.

1.5.8 Lenguaje de Descripción de Servicios Web (WSDL, por sus siglas en inglés)

WSDL es un lenguaje basado en XML, que permite la descripción de los Servicios Web. Se utiliza para la localización y ubicación de estos servicios en la red. No es más que un documento XML que describe ciertas características propias de un Servicio Web, así como aquellos parámetros y métodos que soporta. (34)

Los documentos WSDL definen los *servicios* como colecciones de puntos finales de red o *puertos*. La definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: *mensajes*, que son descripciones de los datos que se están intercambiando y *tipos de puertos*, que son colecciones abstractas de *operaciones*. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un *enlace* reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en su definición: (35)

- <types> define los tipos de datos utilizados en los mensajes. Se utilizan los tipos definidos en la especificación de esquemas XML.
- <message> define los datos del mensaje que participan en una operación. Este puede estar compuesto por varias partes, y cada una de ellas puede ser de un tipo diferente.
- <operation> descripción abstracta de una acción admitida por el servicio.
- <portType> define las operaciones que proporciona el Servicio Web. Conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- <binding> define el formato del mensaje y detalles del protocolo para cada portType.
- <port> punto final único que se define como la combinación de un enlace y una dirección de red.
- <service> colección de puntos finales relacionados.

Un documento WSDL tiene una estructura semejante a la siguiente:

<definitions>

<types>

los tipos de datos...

</types>

<message>

las definiciones del mensaje...

</message>

<portType>

las definiciones de operación ...

</portType>

<binding>

las definiciones del protocolo...

</binding>

</definitions>

EL modelo de arquitectura de un Servicio Web se detalla en la figura 6. Los servicios son descritos en un documento WSDL, donde se especifica el formato de los mensajes intercambiados entre el cliente y proveedor. El Protocolo Simple de Acceso a Datos (SOAP, por sus siglas en inglés) es el que se establece para dicha comunicación. Una vez desarrollado el Servicio Web, se registra para que sea accesible por los clientes en el UDDI[4], en el cual es posible la publicación y búsqueda de los Servicios Web.

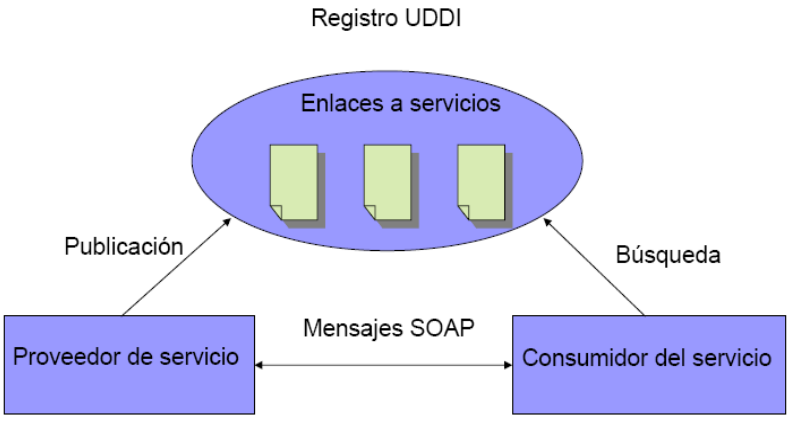


Fig.6 Arquitectura de los Servicios Web.

[4] Universal Description, Discovery and Integration (UDDI): Es uno de los estándares básicos de los Servicios Web, cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

De forma general, los WSDL: (36)

- Facilitan escribir y mantener los Servicios Web.
- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Permiten la interoperabilidad entre plataformas diferentes por medio de protocolos estándares.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten describir y localizar los Servicios Web.

1.5.9 Herramientas y tecnologías a utilizar

Se propone como herramientas a utilizar para el desarrollo del componente, según el Documento de Arquitectura de Software 1.2 de la Facultad 7:

- Sistema Gestor de Base de Datos: PostgreSQL 8.3.0.1
- Lenguaje de Marca: XML 1.0
- Lenguaje de programación: PHP 5.2.5
- Framework PHP : Symfony 1.0.8
- Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés): Zend Studio 5.5
- Servidor Web: HTTP Apache 2.2.6

1.6 Conclusiones

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la compresión de este trabajo. Además, se realizó un estudio de las tendencias de las tecnologías y técnicas usadas para la solución del problema en la actualidad. Se propuso como patrón de arquitectura a tener en cuenta para el modelado del componente el MVC, y el Composite como patrón de diseño.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

El Capítulo se refiere a las características del sistema, presenta el objeto de estudio, la situación problémica, problema, objeto de informatización y la propuesta del sistema. Se definen los requisitos funcionales y no funcionales. Se realiza el modelado del sistema y se describen brevemente los casos de uso.

2.2 Situación problémica y problema

Con el desarrollo de la informática, las entidades del país han visto la necesidad de informatizar la mayor cantidad de procesos posibles para lograr mejor eficiencia en sus procesos internos. Las empresas ofrecen diferentes servicios, éstos son actividades que se ejecutan y necesitan de una planificación de recursos para poder ofrecerlos a los clientes o para el funcionamiento de la entidad.

La actividad de planificar es el proceso de cálculo de necesidades de los materiales o recursos destinados a garantizar la prestación de los servicios del negocio. Los recursos a planificar en cualquier entidad pueden ser de varios tipos en dependencia de los servicios que brinda.

La mayoría de los procesos de planificación de recursos de estos servicios se llevan a cabo manualmente o con tecnología obsoleta. Las escasas aplicaciones existentes no son flexibles a la hora de asimilar cualquier cambio en su estructura y son desarrolladas a la medida, en pos de las necesidades de un cliente específico.

¿Cómo funciona actualmente el proceso de planificación de recursos de los servicios en las entidades?

No todas las empresas tienen informatizados los procesos de planificación de recursos de los servicios que brindan. Las aplicaciones informáticas con las que cuentan son rígidas, esto significa que no toleran cambios en su modo de operar, tienen prefijados los servicios a brindar y para crear otros, se tiene que acudir a una reingeniería y/o reformulación del código para adaptarse a las nuevas necesidades funcionales. Estas aplicaciones, basadas en la filosofía de software a la medida, no son configurables a cualquier entorno de trabajo empresarial.

2.3 Objeto de informatización

El objeto a informatizar es la gestión de los servicios en los procesos de planificación de recursos. Los servicios de las entidades necesitan ser estructurados para una mejor organización y seguimiento de los mismos.

2.4 Información que se maneja

- *Índice de Consumo*: Constituye la base normativa de las diferentes especialidades de cualquier entidad para la planificación de los servicios que brindan, en los que se establecen las necesidades de consumo de los diferentes recursos por cada actividad. (37)
- *Nivel de Actividad*: Se refiere a la cantidad de recursos que se planifica para realizar un servicio determinado. Sobre la base de los niveles de actividad y los índices de consumo, se determinan las necesidades reales de los materiales necesarios. (38)
- *Servicio***:** Es el conjunto de actividades interrelacionadas que ofrece un suministrador con el fin de que el cliente obtenga un beneficio determinado. Son actividades intangibles, resultado de esfuerzos humanos o mecánicos que producen un hecho, un desempeño o un esfuerzo. Implican generalmente la participación del cliente y que no es posible poseer físicamente, ni transportarlos o almacenarlos. (39)
- *Taxonomía:* Es una forma de clasificar y categorizar un grupo de elementos en forma de jerarquías; es simplemente una estructura en forma de árbol con ramificaciones y cada punto de ésta constituye un nodo. Es una jerarquía semántica en la que las entidades de información se relacionan mediante clases y subclases. (40)

2.5 Propuesta del sistema

Se propone el componente Registro de Servicios para informatizar la gestión de los servicios que se ofrecen en cualquier entorno de negocio. El sistema permite la creación de una o varias taxonomías, que jerarquiza los diferentes servicios a planificar. Tiene un carácter genérico, lo que significa que puede operar en cualquier entidad, adaptándose a las nuevas características del modo de operar de cualquier institución. El componente centra su atención en cómo organizar los servicios, permitiendo además, la reestructuración de los mismos cada vez que se disponga.

El desarrollo de este software implica una novedad a nivel nacional e internacional, debido a las características que posee, se habla de un componente flexible ante cambios en su estructura, que constantemente asimile cambios sin la necesidad de una reprogramación.

El estudio del arte reveló que los sistemas informáticos que se encargan de planificar servicios no poseen esta propiedad, ya que tienen una forma de trabajar estática. Esto significa que son resistentes al cambio, no toleran modificaciones en su estructura.

Esta aplicación dará un paso significativo en el esfuerzo por lograr la informatización de la sociedad cubana, en particular, a las empresas que realizan una planificación de sus recursos, para satisfacer los beneficios buscados por los clientes.

El sistema debe usar los Servicios Web brindados por los componentes de Seguridad y Registro de Materiales con los que necesariamente debe estar integrado, dadas las políticas definidas por SAS y el documento de arquitectura de la Facultad 7.

Este componente debe brindar interfaces Web Services para la interacción con otros sistemas externos, quienes son los clientes que consumen los servicios que brinda la aplicación que se propone. Con la propuesta del uso de los Web Services, se ofrece mayor flexibilidad a los sistemas externos que interactúan con el componente, debido a que no importa la plataforma en que resida quien invoca el servicio.

Se sugiere para el óptimo funcionamiento del sistema, que sea desarrollado sobre la tecnología Web, con el objetivo de que los usuarios no necesiten instalar ninguna aplicación cliente en sus puestos de trabajo, de manera que sólo con una computadora y un navegador puedan acceder fácilmente, por la forma centralizada que estarán los datos y para un mejor uso en la red.

2.6 Modelo de dominio

El objetivo del modelado del dominio es comprender y describir los conceptos más importantes dentro del contexto del problema. El modelo del dominio ayuda a los usuarios, clientes, desarrolladores, y otros interesados a utilizar un vocabulario común. Éste es usado en el desarrollo de los modelos de casos de uso y de análisis.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (41)

A continuación se muestra la figura 7, en la que se detallan los conceptos más importantes que se manejan dentro del contexto del problema:

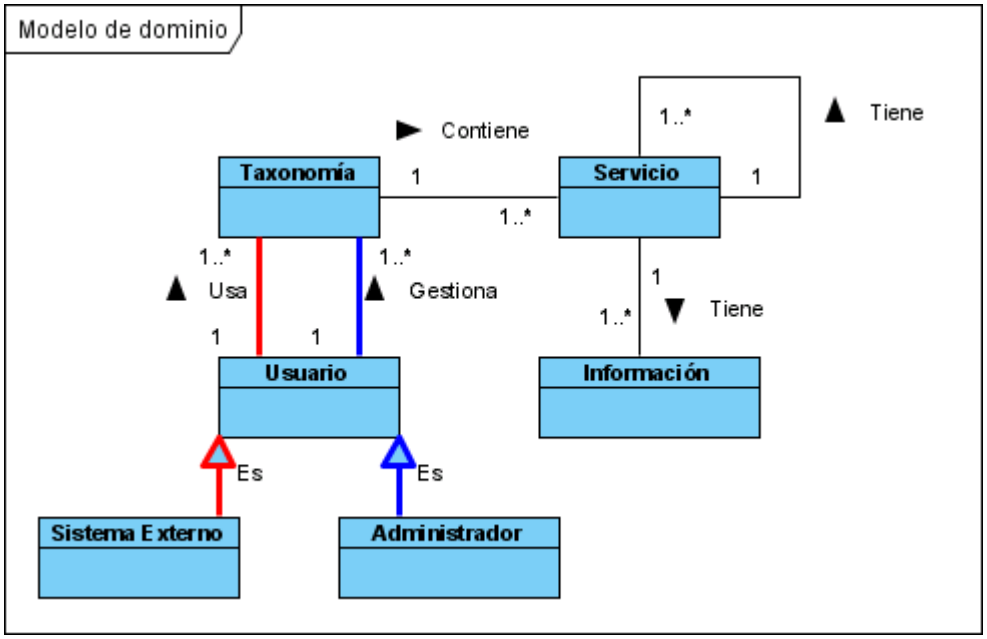


Fig.7 Modelo de dominio.

2.7 Caso de estudio: Peluquería- Barbería

Las barberías – peluquerías del país ofrecen un conjunto de servicios al cliente orientados a cubrir sus expectativas y dedicados fundamentalmente a los servicios de cuidado y belleza tanto masculina como femenina.

Los servicios que ofrecen se agrupan en dos tipos: peluquería y barbería respectivamente.

Dentro de los servicios de peluquería, se ofrecen:

- Peinado
- Secado
- Corte
- Pedicure
- Manicure
- Cosmetología
  - Maquillaje
- Lavado de cabello

Dentro de los servicios de barbería, se ofrecen:

- Afeitado
- Corte
- Tratamiento de los pies
- Peinado
- Lavado de cabello

Para llevar a cabo cada uno de estos servicios, se requiere de algunos recursos indispensables para éxito de los mismos. Por ejemplo, los servicios de barbería necesitan para su funcionamiento el siguiente listado de materiales:

- Cremas para afeitar
- Navajas
- Afeitadoras eléctricas
- Limas de uñas
- Tijeras
- Corta- uñas
- Peines
- Cepillos
- Máquinas eléctricas, entre otros.

Para ilustrar cómo quedaría la taxonomía para el negocio correspondiente, se utilizará una parte de los servicios que se mencionaron anteriormente para minimizar el tamaño de la estructura. Ver figura 8:

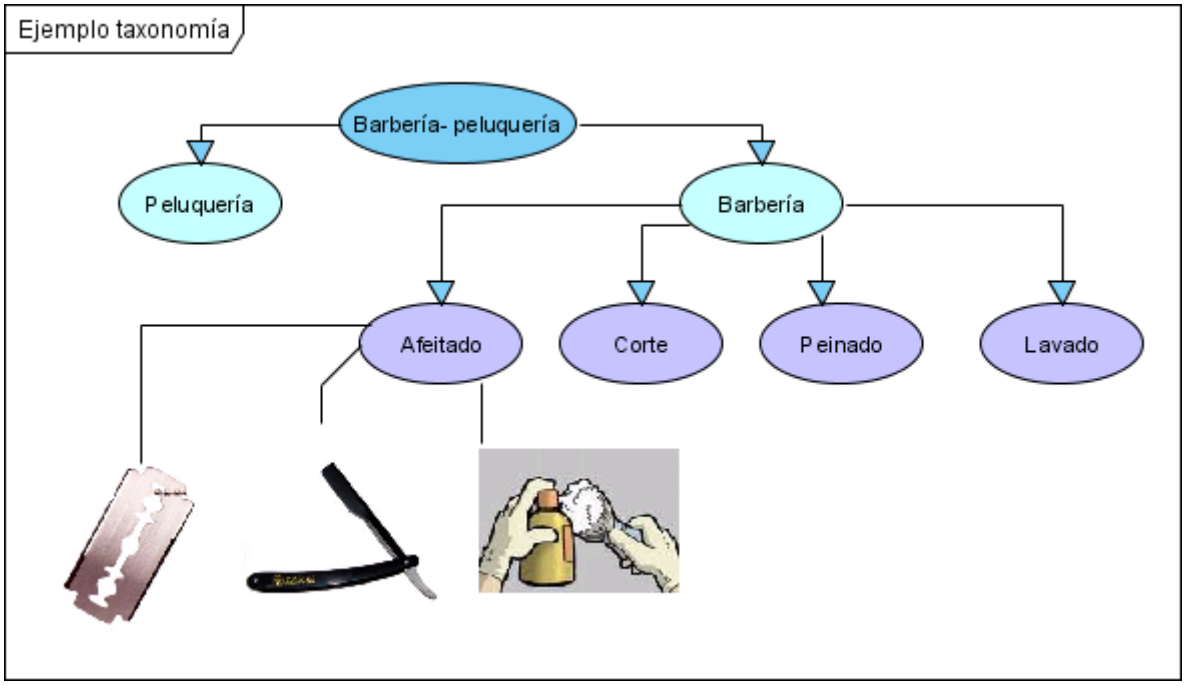


Fig.8 Taxonomía para el caso de estudio: Peluquería-Barbería.

2.8 Especificación de los requisitos de software

2.8.1 Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. (42)

A continuación se enumeran los requisitos funcionales que debe cumplir el sistema:

RF.1 Autenticar
RF.2 Autorizar
RF.3 Gestionar taxonomía
- RF.3.1 Insertar taxonomía
- RF.3.2 Modificar taxonomía
- RF.3.3 Eliminar taxonomía

RF.4 Gestionar servicios
- RF.4.1 Insertar servicios
- RF.4.2 Modificar servicios
- RF.4.3 Eliminar servicios

RF.5 Gestionar información de los servicios

RF.5.1 Insertar información de los servicios

RF.5.2 Modificar información de los servicios

RF.5.3 Eliminar información de los servicios

RF.6 Buscar servicios

RF.6.1 Proveer Servicios Web

RF.7 Listar taxonomía

2.8.2 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, características que hacen al producto atractivo, usable, rápido y confiable. En muchos casos, los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo debe comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. (43)

- *Apariencia o interfaz externa.*

La aplicación se realizará en ambiente Web. La interfaz debe ser sencilla, amigable y asequible al usuario, teniendo en cuenta que éste puede no ser especialista en informática. Debe ser fácil de usar y a la vez poseer un ambiente profesional.

- *Usabilidad*

El sistema podrá ser usado por el administrador de la entidad, quien realiza la configuración de la aplicación y por los sistemas externos que necesiten consumir los Servicios Web que el componente provee.

- *Rendimiento*

El sistema deberá ser rápido en el procesamiento de la información. La eficiencia de la aplicación estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo Cliente/Servidor, y la velocidad de las consultas a la base de datos. Se realizará la validación de los datos en el cliente y en el servidor aquellas que por cuestiones de seguridad, o de acceso a los datos lo requieran. Lográndose así un tiempo de respuesta más rápido, mayor velocidad de procesamiento y aprovechamiento de los recursos.

- *Soporte*

El sistema gestor de base de datos que utilizará el sistema como medio de almacenamiento de la información será PostgreSQL 8.3.0.1. Se debe lograr la solidez de los datos realizando mantenimientos automatizados en la base de datos, orientados a la actualización y corrección de la información. El sistema debe contar con una capa de abstracción de datos que le brinde soporte a éste para varios gestores de base de datos. Es decir, el modelo de datos debe ser independiente de la base de datos a utilizar.

- *Seguridad*

El Componente de Seguridad proveerá seguridad a la aplicación otorgando los permisos predefinidos a los usuarios, además de brindar servicios de auditoría de los que se pueden hacer reportes para conocer las acciones realizadas por los usuarios históricamente.

- *Requerimientos de software*

  - Se utilizará tecnología Apache versión 2.2.6 o superior para el servidor Web.
  - En las computadoras de los clientes se requiere de un navegador Web (Internet Explorer versión 6.0 o superior, Mozilla Firefox versión 2.0 o superior).

- *Requerimientos de hardware*
  - *En el lado del cliente:*
    - Ordenador Pentium o superior.
    - Sistema Operativo: Linux Ubuntu 8.04
    - 256 MB RAM.
    - Procesador 486DX / 66 MHZ o superior.
    - Disco duro de 20 GB.
  - *En el lado del servidor:*
    - Sistema Operativo: Linux/Debian 4 Etch.
    - Server de rack de 19 ‘con: NIC de 1 Gbit Ethernet.
    - Array scsi de 3 Hard Disk > 70 Gb.
    - Dual processor.
    - Fuente redundante.

2.9 Modelo del sistema

2.9.1 Definición de los actores

Actores	Justificación
Administrador	Es el encargado de crear la taxonomía, la estructura de cómo van a ser organizados los servicios del negocio en concreto. Inserta los servicios en la estructura definida (taxonomía), además de poder modificarlos y/o eliminarlos. Inserta el índice de consumo de los materiales necesarios para el funcionamiento de los servicios.
Componente de Seguridad para productos del Área Temática Sistemas de Apoyo a la Salud (CS_SAS).	Permite a los usuarios del sistema acceder a la aplicación con los permisos predefinidos en dependencia del rol que desempeña en el negocio, además de proporcionarle seguridad al componente Registro de Servicios.
Registro de Materiales	Registro externo que gestiona la información de los materiales que se planifican, de los que se conocen el nombre, el código y la descripción.
Sistema externo	Es el cliente que consume según sus credenciales, los servicios que necesita en su negocio para la realización del mismo.
Usuario	Tiene permiso según el tipo de usuario (administrador o sistema externo) a realizar una búsqueda de todos los servicios que existen en su(s) taxonomía(s) correspondiente(s).

2.9.2 Diagrama de casos de uso

Un diagrama de casos de uso del sistema representa gráficamente los procesos y su interacción con los actores. Es una representación gráfica del entrono del sistema (actores) y su funcionalidad principal (casos de uso). Muestra los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

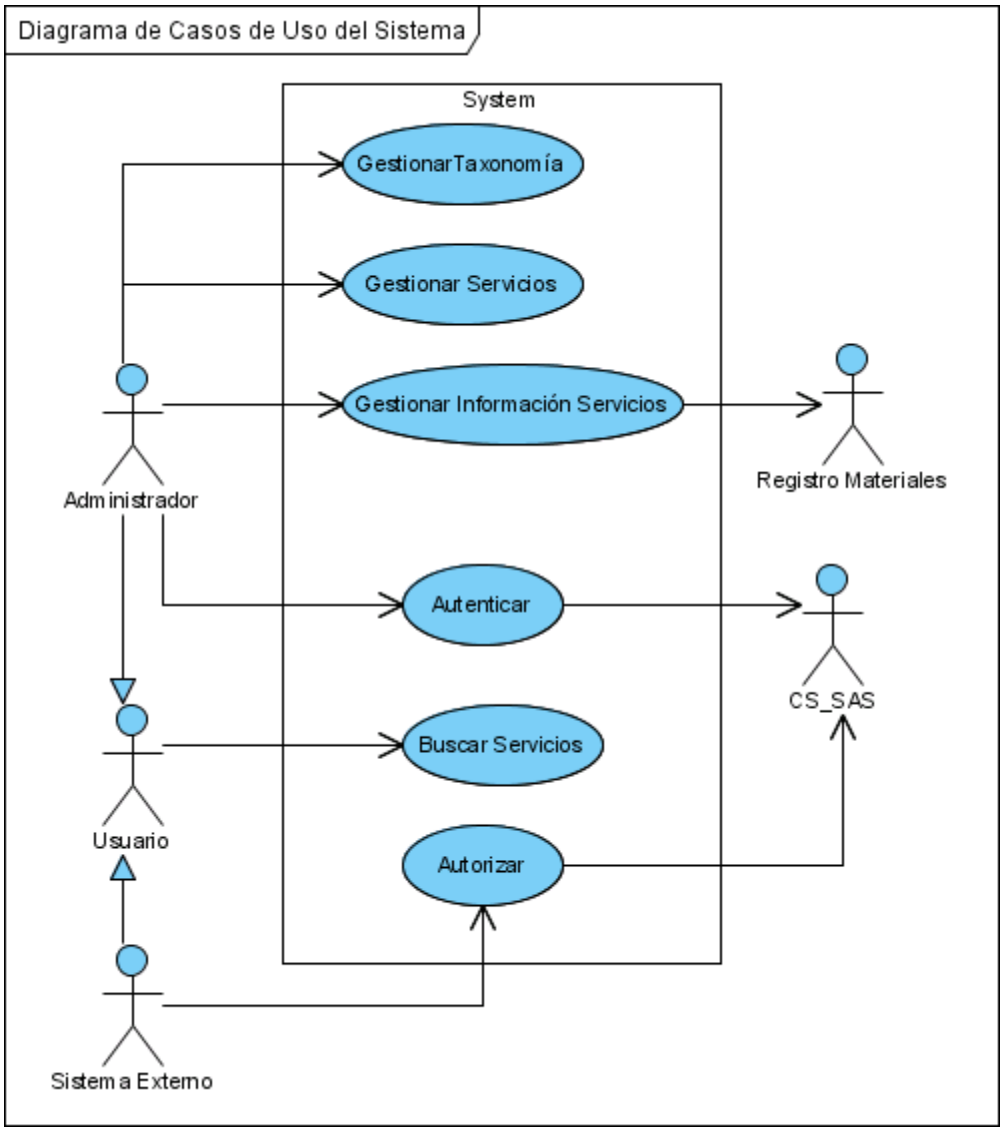


Fig.9 Diagrama de casos de uso del sistema

2.1.1 Descripción de casos de uso

2.1.1.1 CU: Autenticar

CUS-1: Autenticar	
Propósito:	Permitir la entrada al sistema.
Actores:	**Administrador** (Inicia)
Resumen:	El caso de uso inicia cuando un usuario intenta acceder al Componente Registro de Servicios, introduce su nombre de usuario y contraseña, accediendo al Componente de Seguridad, devolviendo los privilegios que le fueron otorgados al usuario en función del rol que desempeña. El caso de uso termina cuando el actor haya accedido al sistema.
Precondición:	El usuario debe conocer el nombre de usuario y contraseña que le fue asignado.
Referencias:	RF1
Prioridad:	Secundario

2.1.1.2 CU: Autorizar

CUS-2: Autorizar	
Propósito:	Permitir a un Sistema Externo acceder al componente Registro de Servicios con los derechos que le fueron otorgados en el Componente de Seguridad.
Actores:	**Sistema Externo** (Inicia)
Resumen:	El caso de uso inicia cuando un usuario desde un Sistema Externo intenta acceder al Componente Registro de Servicios para consumir los servicios que brinda. Éste debe acceder al Componente de Seguridad, quien devuelve las credenciales para poder realizar la búsqueda de los servicios que necesita consumir para su negocio. El caso de uso termina cuando el actor haya accedido al componente Registro de Servicios.
Precondición:	El usuario que se encuentra accediendo desde un Sistema Externo tiene que estar autenticado en el Componente de Seguridad.
Referencias:	RF2
Prioridad:	Secundario

2.1.1.3 CU: Gestionar Taxonomía

CUS-3: Gestionar Taxonomía	
Propósito:	Permitir crear, modificar y eliminar la estructura para registrar los servicios.
Actores:	**Administrador** (Inicia)
Resumen:	El caso de uso inicia cuando el administrador necesita definir una taxonomía para registrar los servicios. Estas taxonomías pueden modificarse y/o eliminarse. Finaliza cuando están creadas todas las taxonomías para el negocio en cuestión.
Precondición:	El administrador debe autenticarse.
Referencias:	RF 3.1, RF 3.2, RF 3.3
Prioridad:	Crítico

2.1.1.4 CU: Gestionar Servicios

CUS-4: Gestionar Servicios	
Propósito:	Permitir insertar, modificar y/o eliminar los servicios en la taxonomía al cual pertenecen.
Actores:	**Administrador** (Inicia)
Resumen:	El caso de uso inicia cuando el administrador necesita insertar los servicios del negocio en la taxonomía correspondiente. El administrador tiene la posibilidad de modificarlos y/ o eliminarlos cuando estime conveniente. El caso de uso finaliza cuando se han insertado todos los servicios.
Precondición:	Debe haberse creado la taxonomía necesaria para insertar los servicios.
Referencias:	RF 4.1, RF 4.2, RF 4.3
Prioridad:	Crítico

2.1.1.5 CU: Gestionar Información Servicios

CUS-5: Gestionar Información Servicios	
Propósito:	Permitir insertar, modificar y/o eliminar la información de los servicios.
Actores:	**Administrador** (Inicia)
Resumen:	El caso de uso inicia cuando el administrador necesita insertar toda la información que le corresponde a un servicio determinado. El caso de uso finaliza cuando se ha insertado la información de los servicios.
Precondición:	Existir la taxonomía y servicios creados.
Referencias:	RF 5.1, RF 5.2, RF 5.3
Prioridad:	Crítico

2.1.1.6 CU: Buscar Servicios

CUS-6: Buscar Servicios	
Propósito:	Permitir realizar la búsqueda de los servicios.
Actores:	Usuario(Inicia)
Resumen:	El caso de uso inicia cuando el usuario (Administrador o Sistema Externo) necesita realizar la búsqueda de un servicio específico de las taxonomías que le corresponden. Debe listarse previamente la taxonomía a la que corresponde el servicio buscado. El caso de uso termina cuando se muestra los resultados de la búsqueda.
Precondición:	Existir el servicio que se necesita conocer la información.
Poscondición:	Se muestran los resultados de la búsqueda.
Referencias:	RF 6, RF 6.1, RF 7
Prioridad:	Crítico

2.10 Conclusiones

En este capítulo se expusieron las características del sistema que se desea obtener. Se analizó la situación problémica y el problema existente, se desarrolló todo el proceso de negocio que existe con respecto a la gestión de servicios en los procesos de planificación de recursos en las empresas. Se puntualizaron los requisitos funcionales y no funcionales necesarios para definir el ámbito del sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se modelan los principales artefactos que se obtienen como resultado del flujo de trabajo de análisis y diseño. En la fase de análisis se muestran los diagramas de clases de análisis, y en la fase de diseño se obtienen los diagramas de secuencia y de clases de diseño. Se representa el flujo de información de la integración de los componentes Registro de Servicios, Registro de Materiales, Componente de Seguridad y los sistemas externos que consumen los servicios que provee Registro de Servicios.

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

3.2 Diagrama de clases de análisis

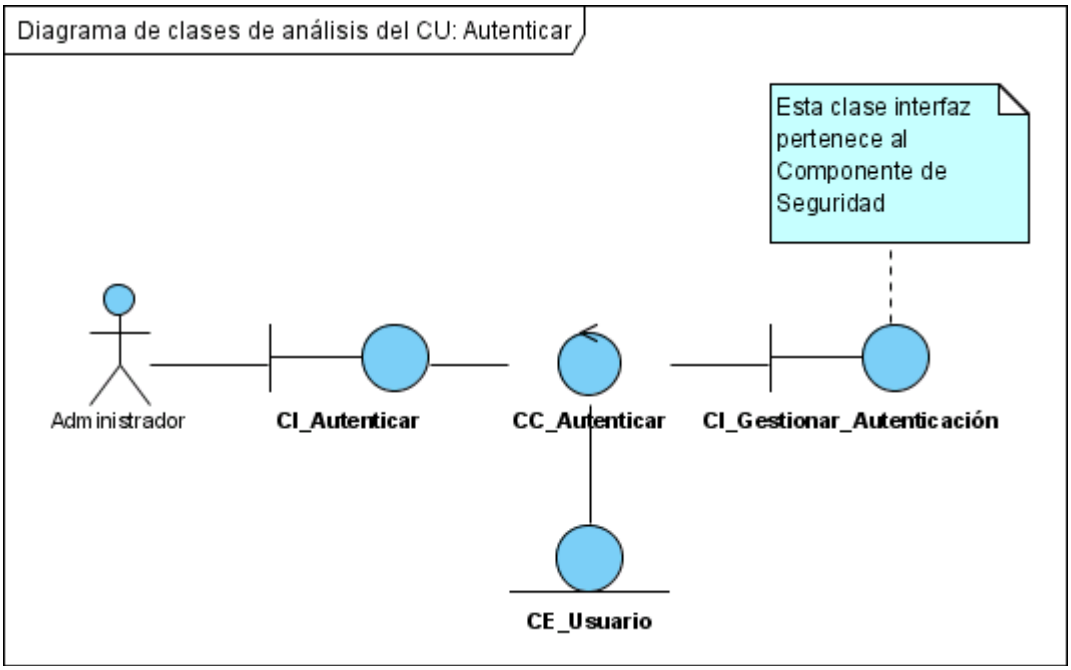
Un diagrama de clases de análisis es un artefacto en el que se representan los conceptos en un dominio del problema, representa las cosas del mundo real. El modelo de análisis puede considerarse como una primera aproximación al modelo de diseño.

Para la representación de las clases de análisis, los estereotipos de las clases son los que se muestran en la siguiente tabla:

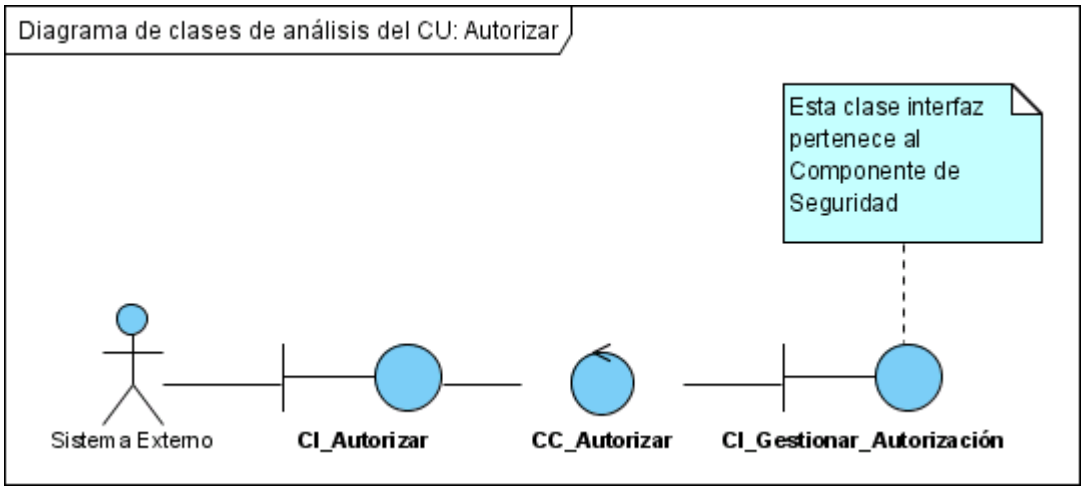
NOMBRE	CARATERÍSTICAS	REPRESENTACIÓN
Entidad	Modelan información que posee larga vida y que es a menudo persistente.	
Interfaz	Modelan la interacción entre el sistema y sus actores.	

Control	Coordinan la realización de uno o unos pocos casos de uso, regulando las actividades de los objetos que implementan la funcionalidad del caso de uso.	

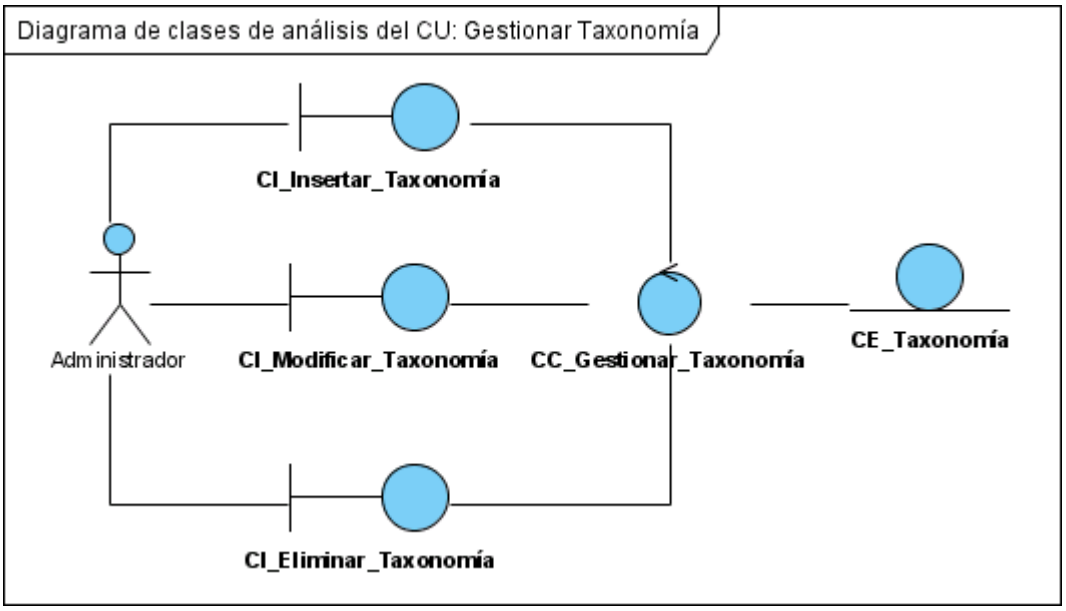
3.2.1 Diagrama de clases de análisis del caso de uso: Autenticar



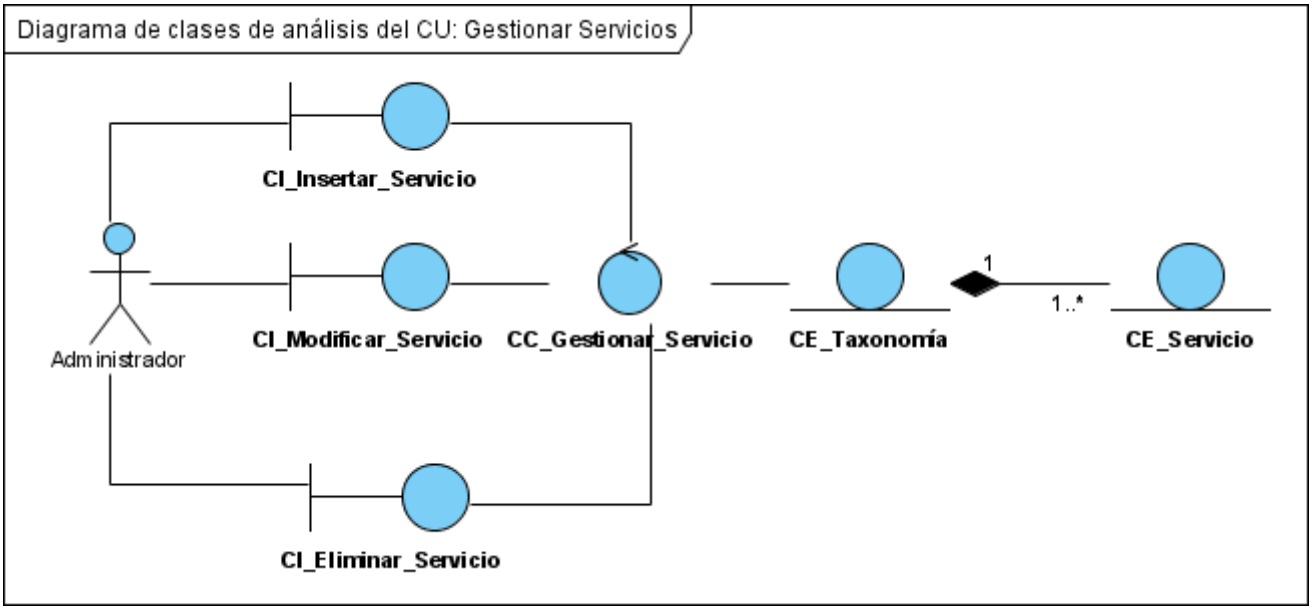
3.2.2 Diagrama de clases de análisis del caso de uso: Autorizar



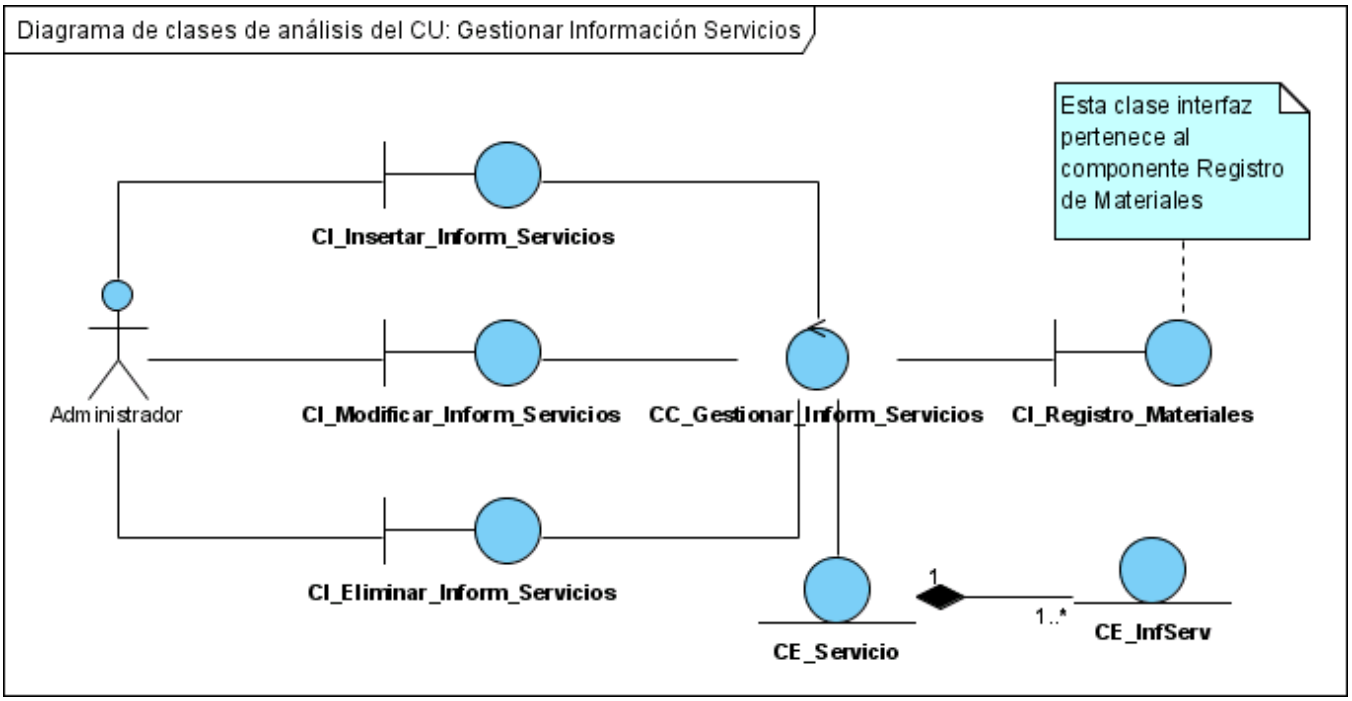
3.2.3 Diagrama de clases de análisis del caso de uso: Gestionar Taxonomía



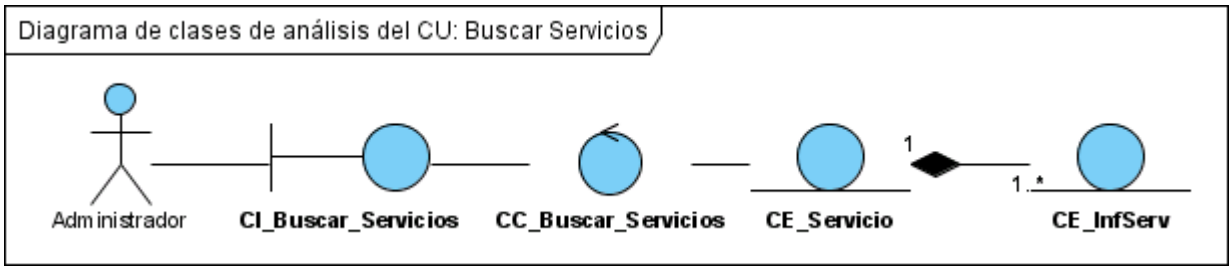
3.2.4 Diagrama de clases de análisis del caso de uso: Gestionar Servicios



3.2.5 Diagrama de clases de análisis del caso de uso: Gestionar Información Servicios



3.2.6 Diagrama de clases de análisis del caso de uso: Buscar Servicios



3.3 Diagrama de clases de diseño

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos.

Para la realización de los diagramas de clases de diseño se utilizó la extensión de UML para el modelado de aplicaciones Web. Esta extensión presenta como elementos más significativos a tres clases estereotipadas “Server Page”, “Client Page” y “Form” empleadas para el código servidor, código cliente y formularios respectivamente, permitiendo además representar ficheros contenedores de sentencias script como por ejemplo PHP y JavaScript. (44)

Se utilizaron los estereotipos Web que se emplean en Visual Paradigm para las clases de diseño, así como el conjunto de relaciones entre ellas, especificadas en la siguiente tabla:

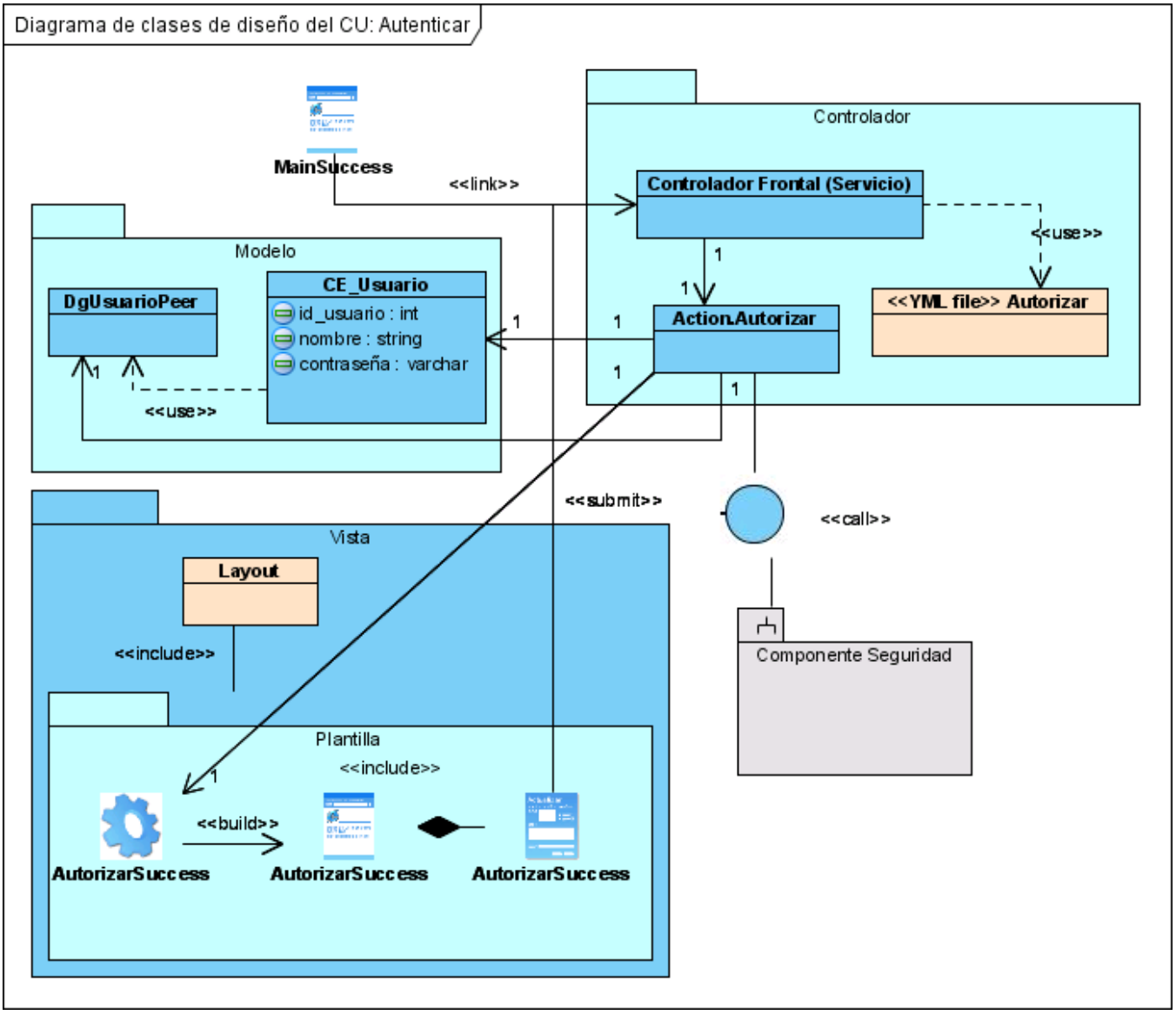
Estereotipos / Desde	Hasta	Client Page	Form	Server Page
	**Client Page**	<<link>>, <<redirect>>	aggregation	<<link>>, <<redirect>>
	**Form**	aggregated by		<<submit>>
	**Server Page**	<<build>>, <<redirect>>		<<redirect>>,<<include>>

Fig.10 Estereotipos y relaciones entre las clases principales que conforman la extensión de UML para aplicaciones Web.

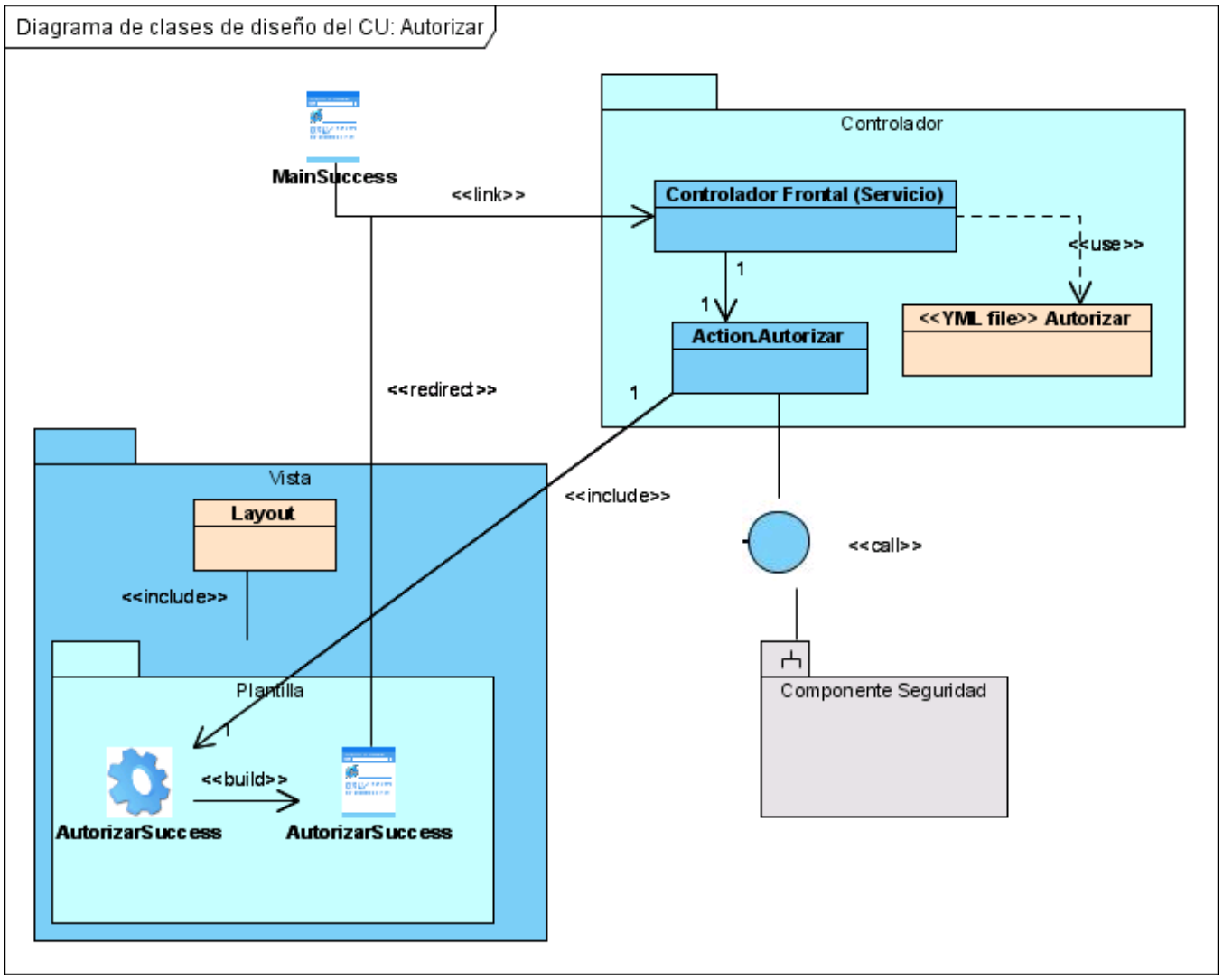
El código servidor se encarga de construir o generar el resultado XHTML que conforma el código cliente (<<build>>). Los formularios envían sus datos al código servidor para ser procesados los pedidos (<<submit>>), además forman parte del código cliente o resultado XHTML, es por esto, que la relación entre la clase empleada para el código cliente y la clase para el formulario es de agregación. Entre páginas clientes pueden existir vínculos (<<link>>) o redireccionamientos (<<redirect>>). Es importante destacar que una página cliente es construida por una sola página servidora. Esta a su vez, puede completar su funcionamiento incluyendo código existente en otra página de este mismo tipo,

utilizando la relación de inclusión (<<include>>), que aunque no es propia de la extensión de UML, las herramientas para el modelado la consideran en aras de representar todas las relaciones existentes en el modelo. (45)

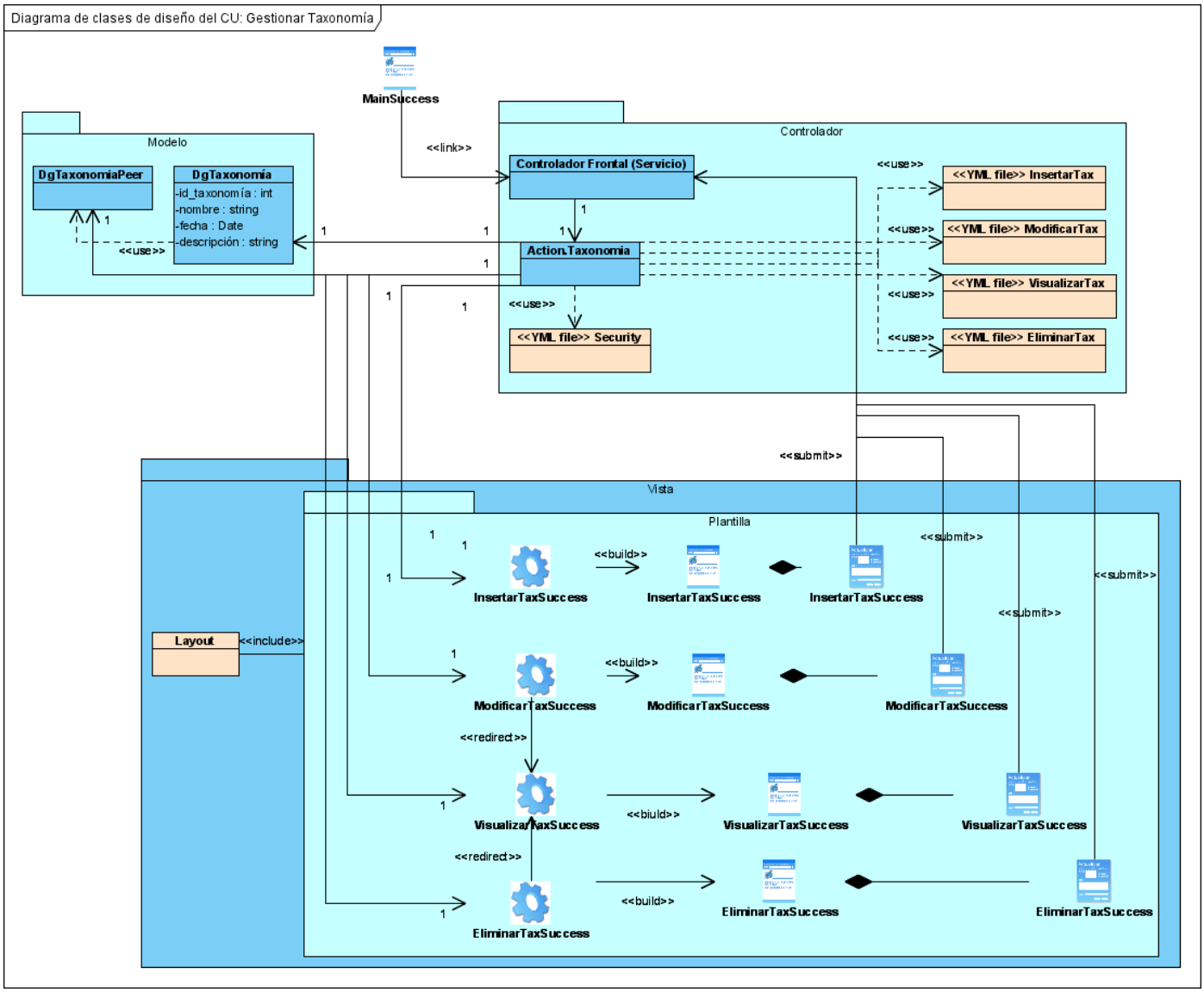
3.3.1 Diagrama de clases de diseño del caso de uso: Autenticar



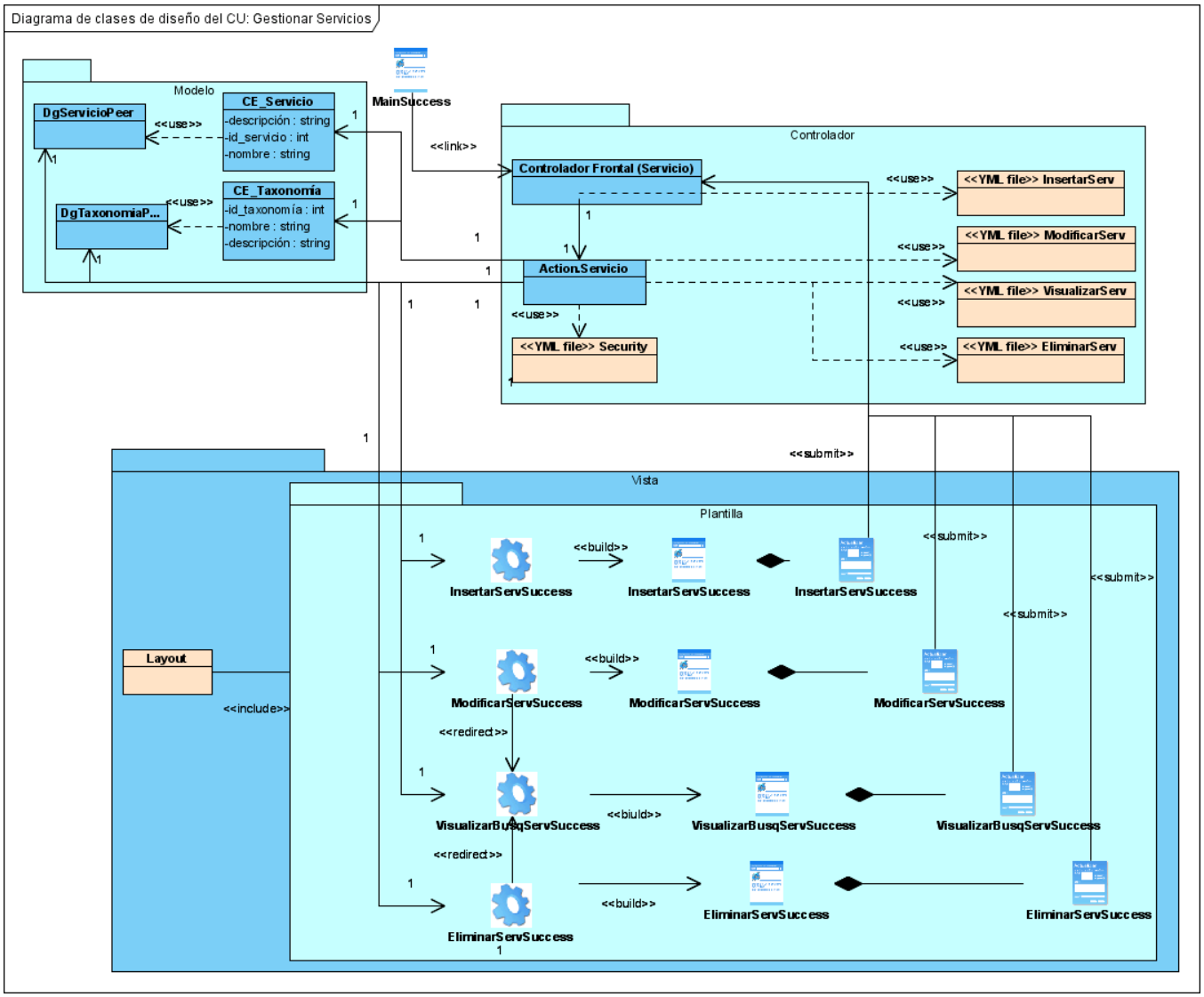
3.3.2 Diagrama de clases de diseño del caso de uso: Autorizar



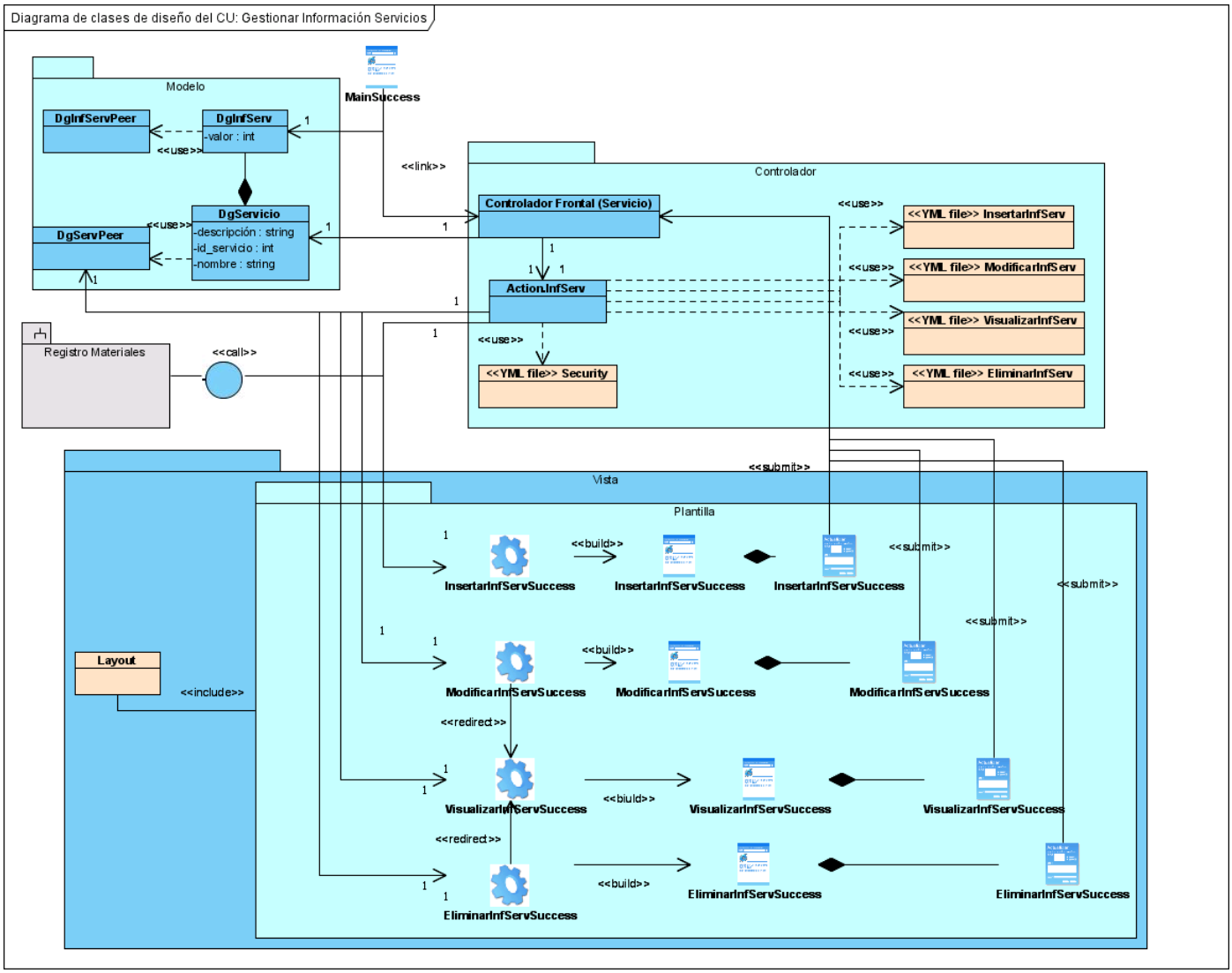
3.3.3 Diagrama de clases de diseño del caso de uso: Gestionar Taxonomía



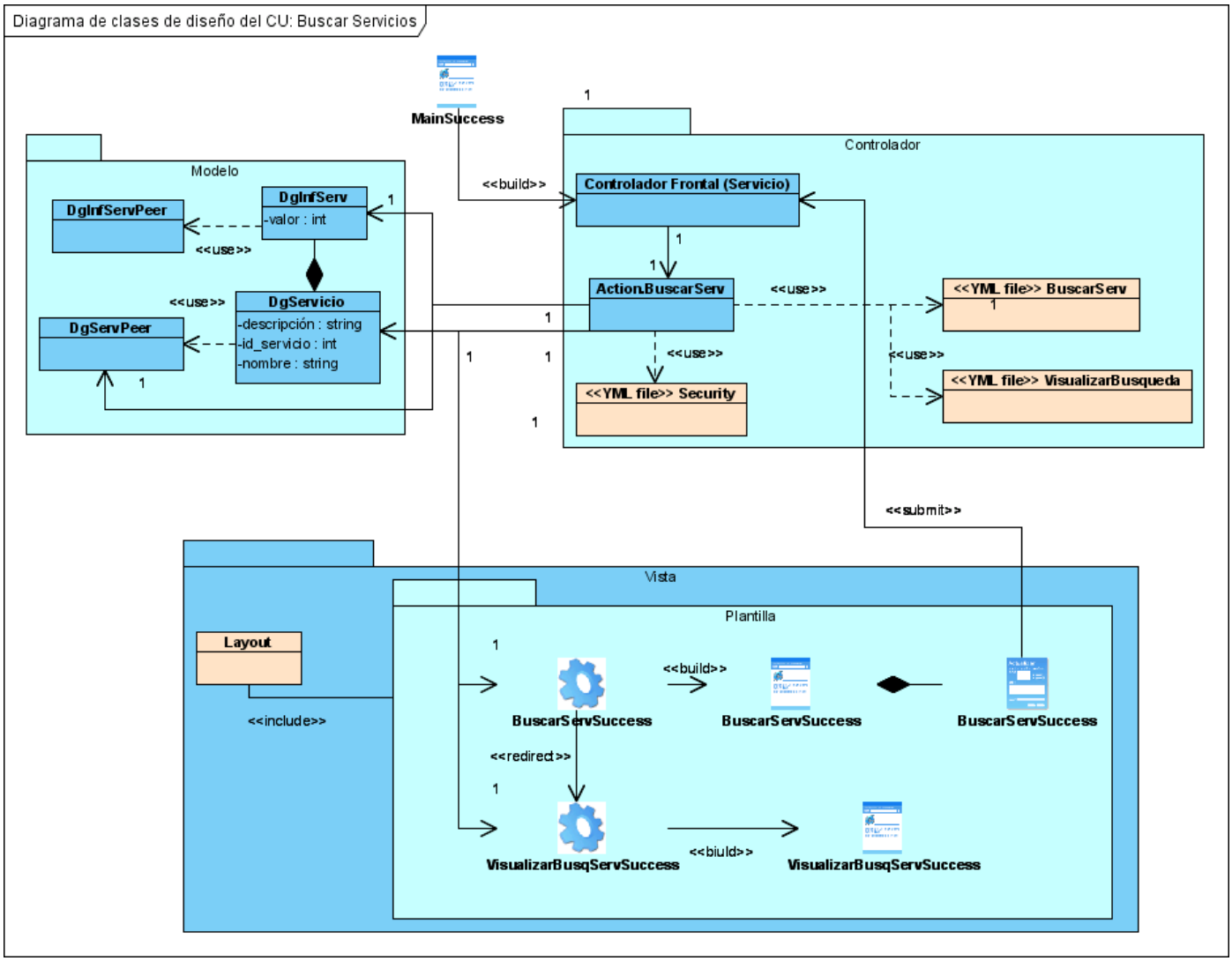
3.3.4 Diagrama de clases de diseño del caso de uso: Gestionar Servicios



3.3.5 Diagrama de clases de diseño del caso de uso: Gestionar Información Servicios



3.3.6 Diagrama de clases de diseño del caso de uso: Buscar Servicios



3.4 Descripción textual de los casos de uso

Cada caso de uso se centra en describir cómo alcanzar una única meta o tarea del negocio. Un caso del uso describe una característica del sistema, el comportamiento del software o de los sistemas, y contiene una descripción textual de todas las maneras que los actores previstos pueden trabajar con el software o sistema.

3.4.1 Descripción textual del caso de uso: Gestionar Taxonomía

Nombre del CU	**Gestionar Taxonomía (CU-3)**
Actores	Administrador (Inicia)
Propósito	Permite insertar, modificar y eliminar la taxonomía en la que se van a registrar los servicios.
Resumen	El caso de uso inicia cuando el administrador necesita crear la estructura en la cual serán registrados los servicios. Ésta puede ser modificada y/o eliminada. El caso de uso finaliza cuando están creadas las taxonomías para el negocio en cuestión.
Referencias	RF 3.1, RF 3.2, RF 3.3
Precondiciones	Para crear la taxonomía es necesario que el administrador se haya autenticado.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Crear Taxonomía que aparece en la parte derecha de la pantalla. (Ver figura 9) 3. Introduce la información de la taxonomía que va a crear y define su estructura. Para insertar los niveles, debe pinchar el botón  . Una vez creada la taxonomía, presiona el botón Aceptar. 6. Si decide modificar la taxonomía, ir a la sección "Modificar Taxonomía". 7. Si decide eliminar la taxonomía, ir a la sección "Eliminar	2. Muestra una ventana nombrada "Crear Taxonomía" en la que el administrador debe insertar los datos relacionados a la taxonomía (nombre, descripción), así como definir la estructura en la que serán organizados los servicios, o sea, los niveles de la taxonomía. (Ver figura 10) 4. Guarda la información introducida por el administrador y muestra un mensaje indicando que la operación se realizó correctamente. 5. Muestra el nombre de la taxonomía en la parte izquierda de la pantalla.

Taxonomía".	
Sección "Modificar Taxonomía"	
Acciones del Actor	**Respuesta del Sistema**
1 El administrador debe seleccionar la taxonomía a modificar en el menú de la parte izquierda y presionar clic derecho sobre la taxonomía. (Ver figura 11) 3. Selecciona la opción "Modificar Taxonomía". 5. Realiza los cambios a la taxonomía y pulsa el botón Aceptar.	2. Despliega un menú indicando las operaciones que el administrador puede realizar sobre la taxonomía. 4. Muestra una ventana con la información relacionada a la taxonomía, donde el administrador puede realizar los cambios deseados. 6. Guarda la información introducida por el administrador y muestra un mensaje indicando que la operación se realizó correctamente.
Sección "Eliminar Taxonomía"	
Acciones del Actor	**Respuesta del Sistema**
1 El administrador debe seleccionar la taxonomía a eliminar en el menú de la parte izquierda y presionar clic derecho sobre la taxonomía. (Ver figura 11) 3. Selecciona la opción "Eliminar Taxonomía"	2. Despliega un menú indicando las operaciones que el administrador puede realizar sobre la taxonomía. 4. Elimina la taxonomía seleccionada con toda su información.
Prioridad: Crítica	

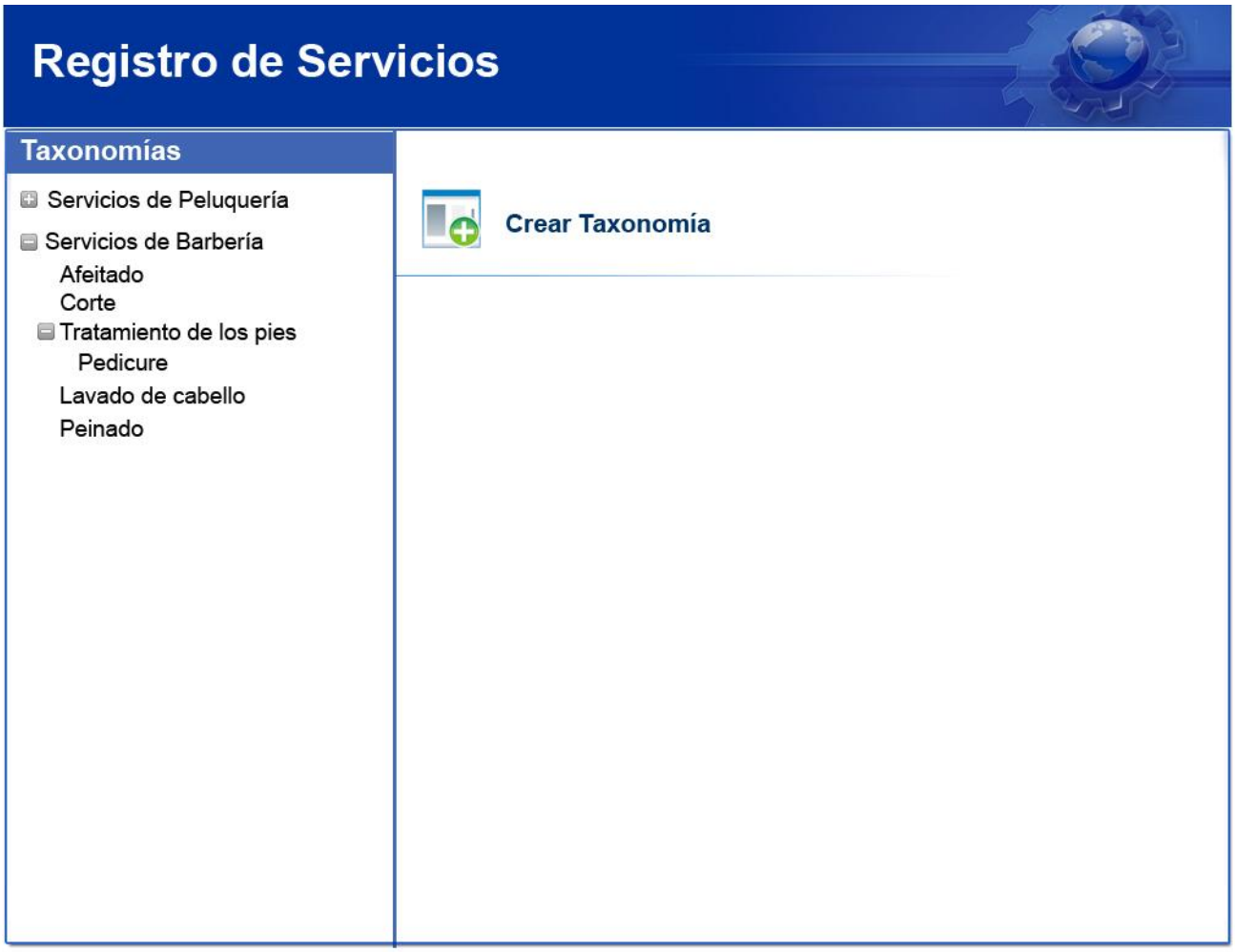


Fig.11 Pantalla *Taxonomía.*

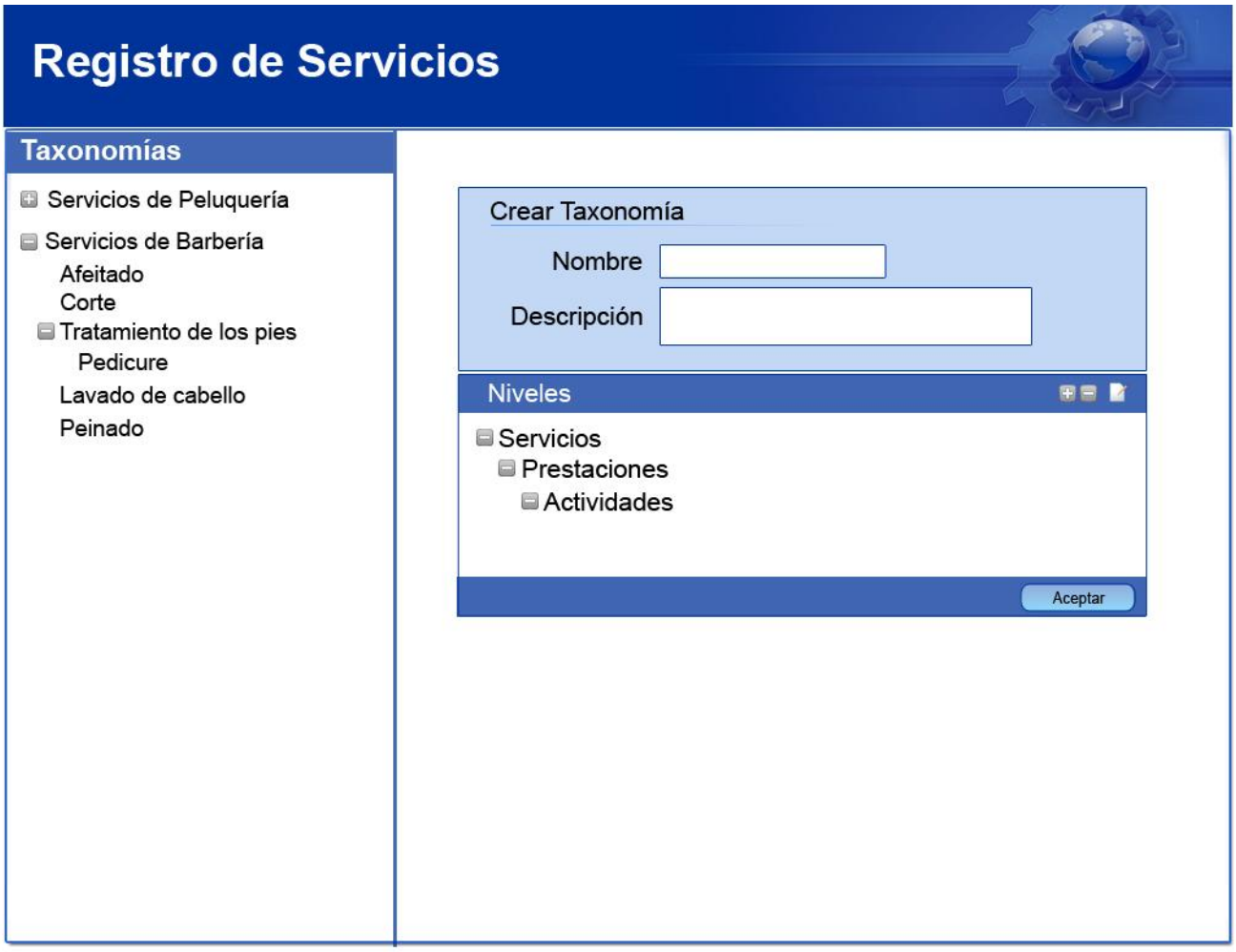


Fig.12 Pantalla *Crear Taxonomía.*

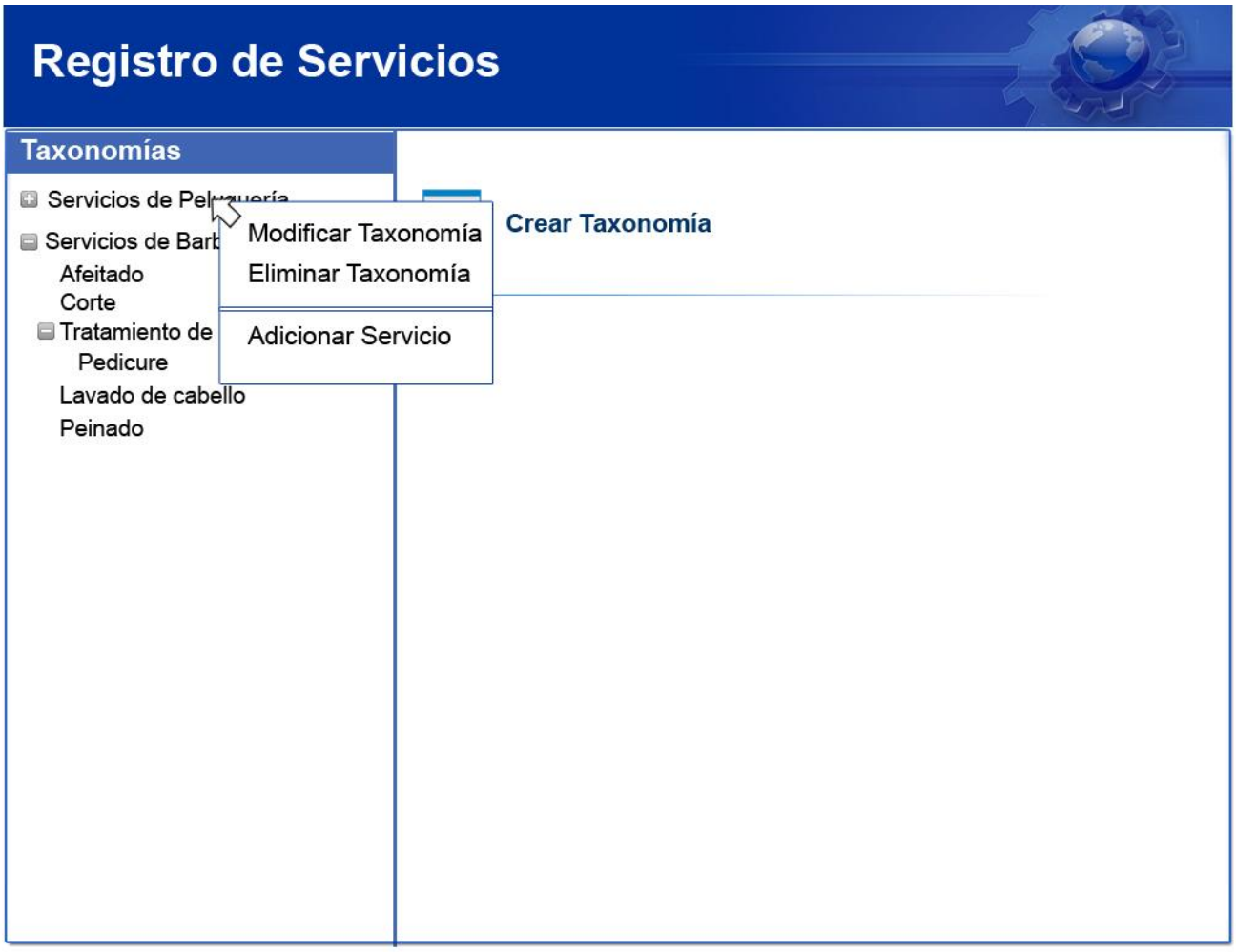


Fig.13 Pantalla para los eventos *Modificar Taxonomía, Eliminar Taxonomía y Adicionar Servicio.*

3.4.2 Descripción textual del caso de uso: Gestionar Servicios

Nombre del CU	Gestionar Servicio (CU-4)
Actores	Administrador.
Propósito	Permite insertar los servicios en la taxonomía definida, así como modificarlos, visualizarlos y/o eliminarlos.
Resumen	El caso de uso inicia cuando el administrador necesita insertar los servicios en la taxonomía definida. Tiene la posibilidad de modificar y/o eliminar un servicio cuando estime conveniente. El caso de uso finaliza cuando ha insertado un servicio.
Referencias	RF 4.1, RF 4.2, RF 4.3

Precondiciones	Existir la taxonomía para registrar los servicios.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador debe dirigirse al menú que aparece en la parte izquierda de la pantalla Taxonomía, donde se encuentran listadas las taxonomías creadas y pulsar clic derecho para adicionar los servicios. (Ver figura 13) 3. Llena los campos de la ventana y pulsa el botón aceptar. 6. Si decide modificar un servicio, ir a la sección "Modificar Servicio". 7. Si decide eliminar un servicio, ir a la sección "Eliminar Servicio"	2. Muestra una ventana nombrada "Adicionar Servicio" en la que el administrador debe insertar el nombre y la descripción. (Ver figura 14). 4. Muestra los datos introducidos por el administrador y muestra un mensaje indicando que la operación se realizó exitosamente.
Sección “Modificar Servicio”	
Acciones del Actor	**Respuesta del Sistema**
1 El administrador debe seleccionar el servicio a modificar en el menú de la parte izquierda y presionar clic derecho sobre el servicio. 3. Selecciona la opción "Modificar Servicio". 5. Realiza los cambios a la taxonomía y pulsa el botón Aceptar.	2. Despliega un menú indicando las operaciones que puede realizar sobre el servicio. 4. Muestra una ventana con la información relacionada al servicio, donde el administrador puede realizar los cambios deseados. 6. Guarda la información introducida por el administrador y muestra un mensaje indicando que la operación se realizó correctamente.
Sección “Eliminar Servicio”	
Acciones del Actor	**Respuesta del Sistema**

1 El administrador debe seleccionar el servicio a eliminar en el menú de la parte izquierda y presionar clic derecho sobre el servicio. 3. Selecciona la opción "Eliminar Taxonomía"	2. Despliega un menú indicando las operaciones que el administrador puede realizar sobre el servicio. 4. Elimina el servicio seleccionada con toda su información.
Prioridad: Crítica	

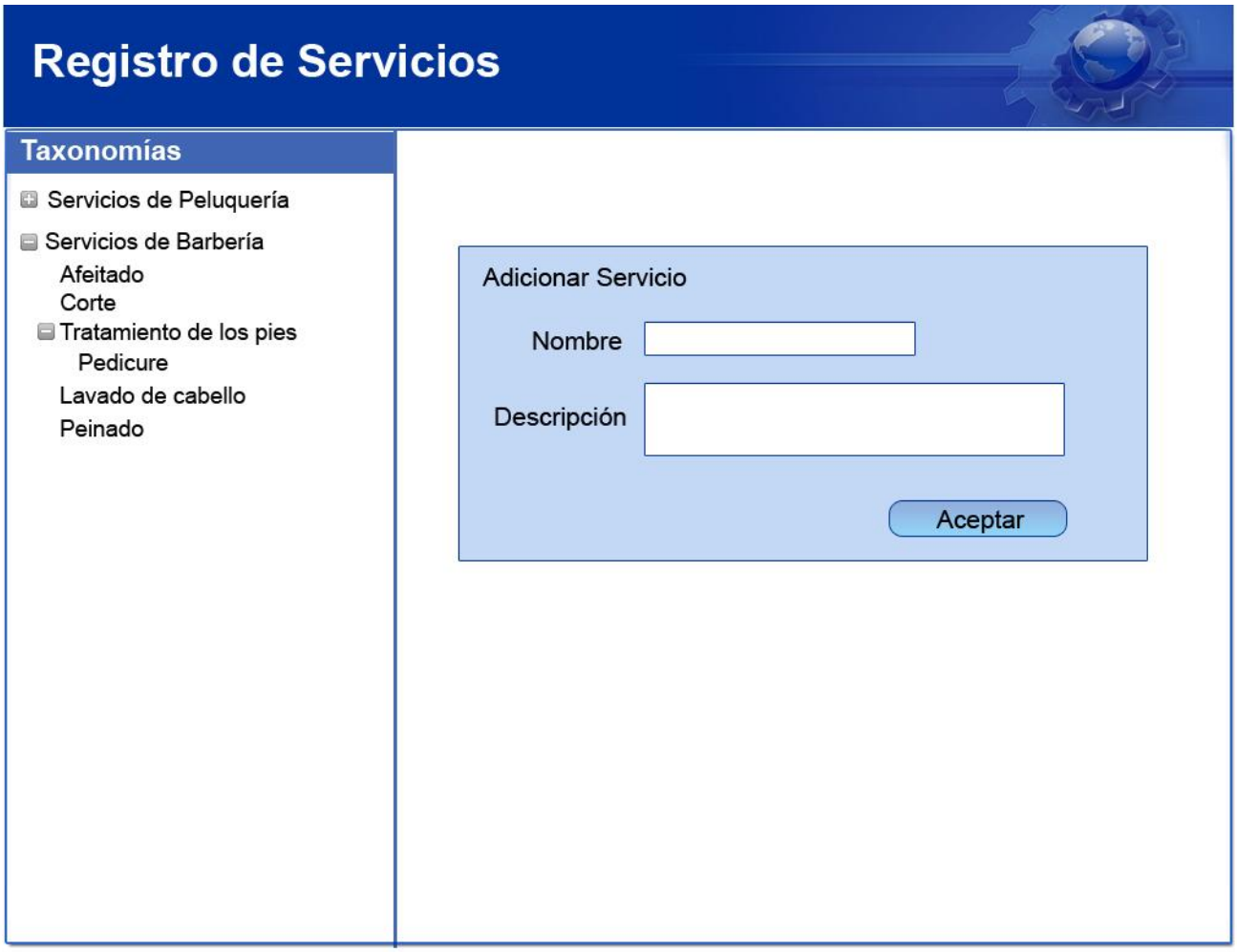
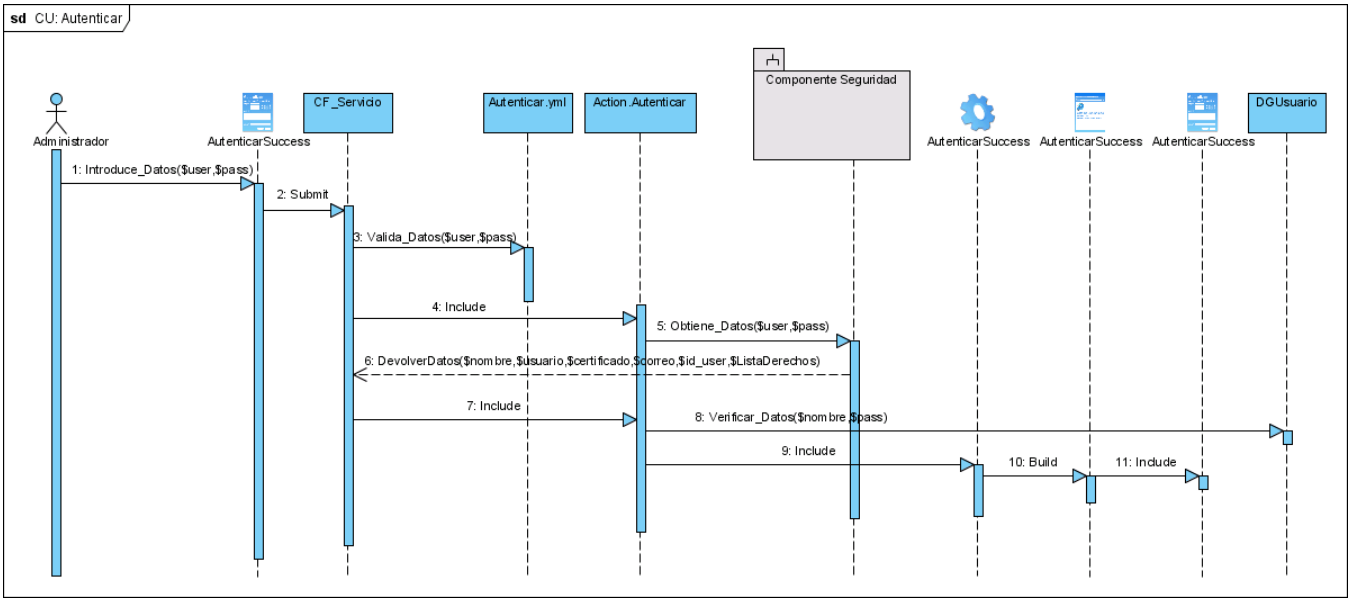


Fig.14 Pantalla *Adicionar Servicio.*

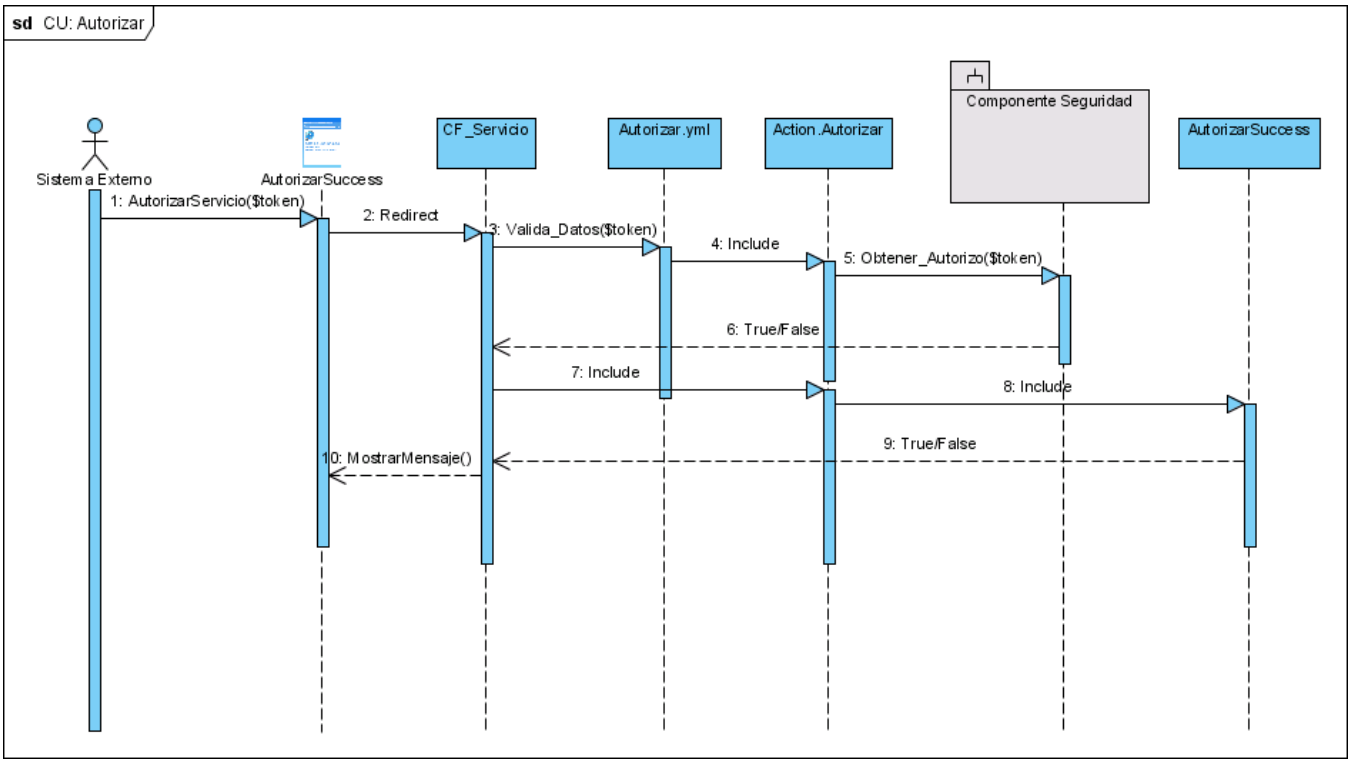
3.5 Diagramas de secuencia

El diagrama de secuencia muestra para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas. Todos los sistemas se tratan como cajas negras; los diagramas destacan los eventos que cruzan los límites del sistema desde los actores a los sistemas.

3.5.1 Diagrama de secuencia del Caso de Uso: Autenticar

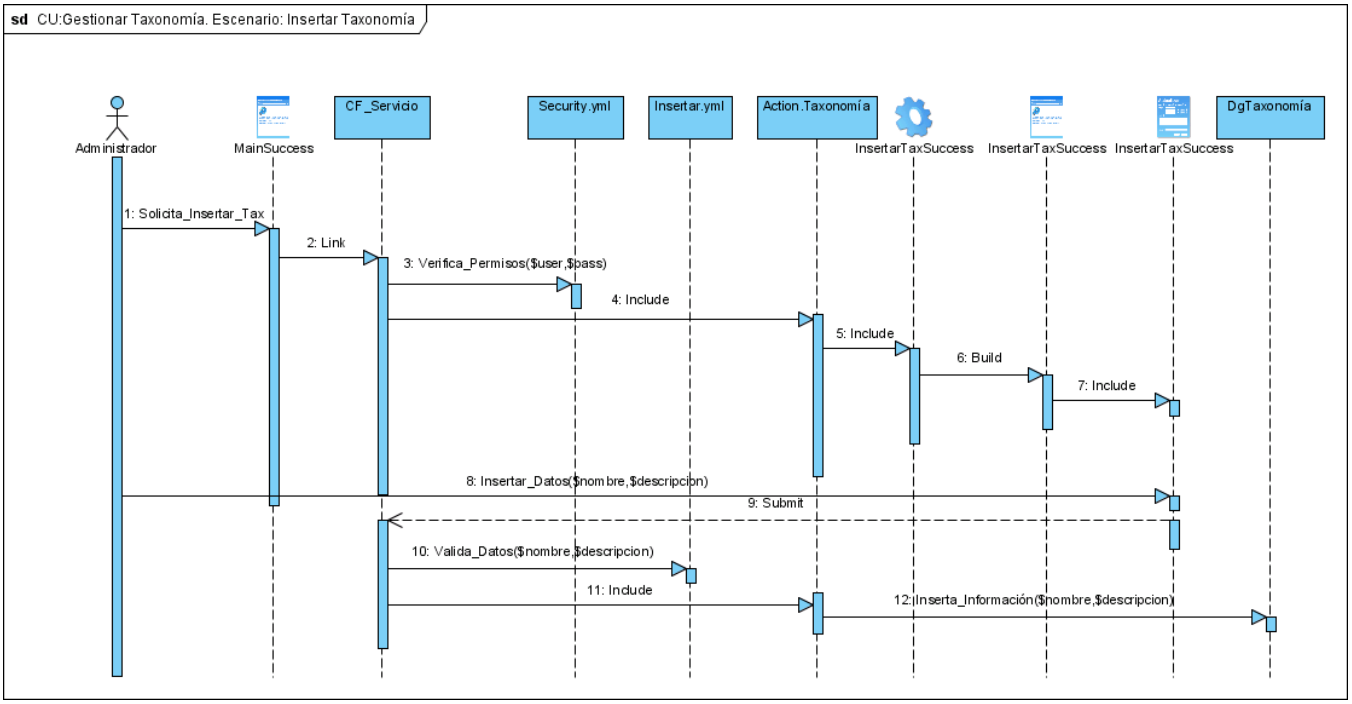


3.5.2 Diagrama de secuencia del Caso de Uso: Autorizar

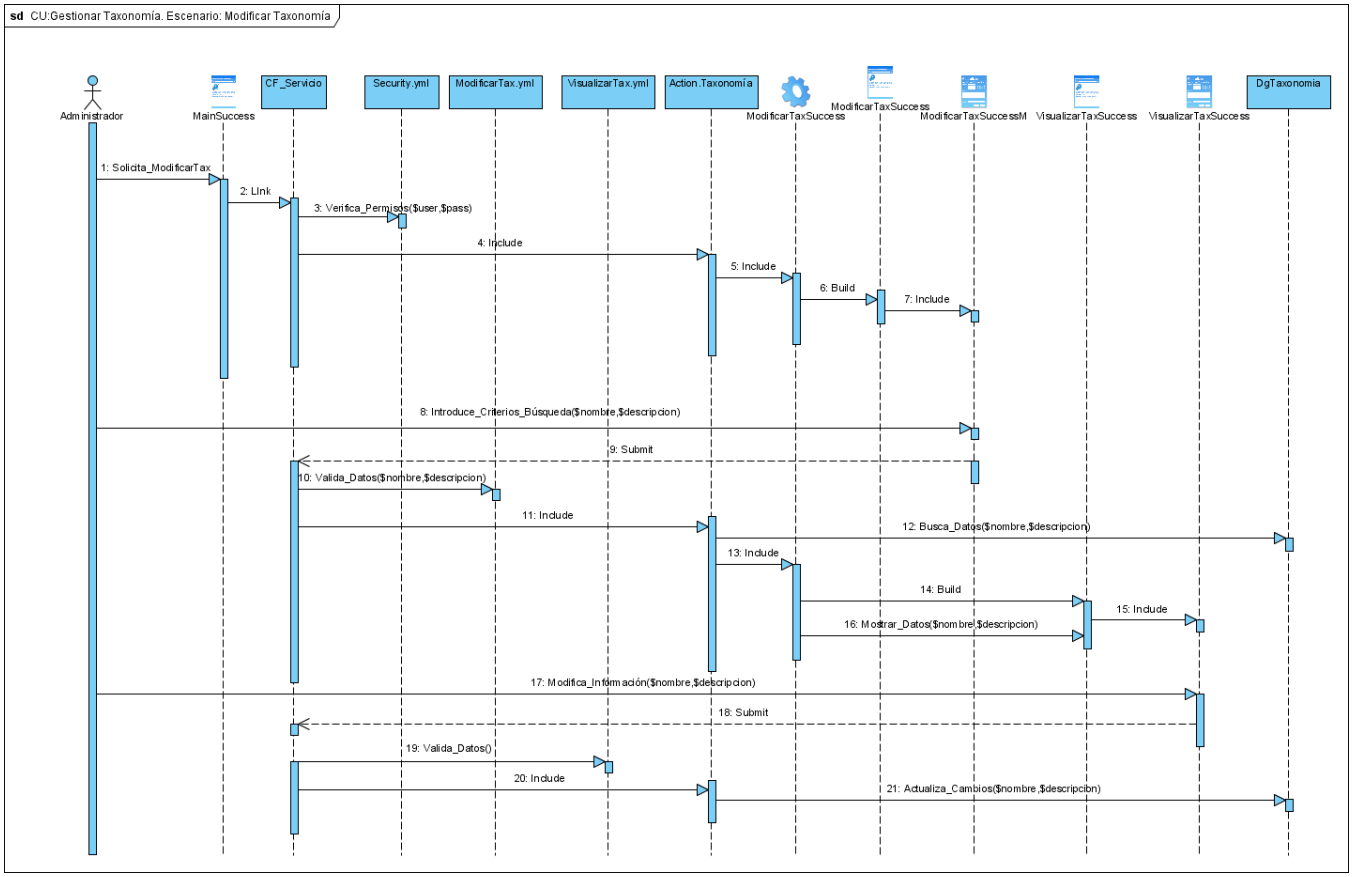


3.5.3 Diagrama de secuencia del Caso de Uso: Gestionar Taxonomía

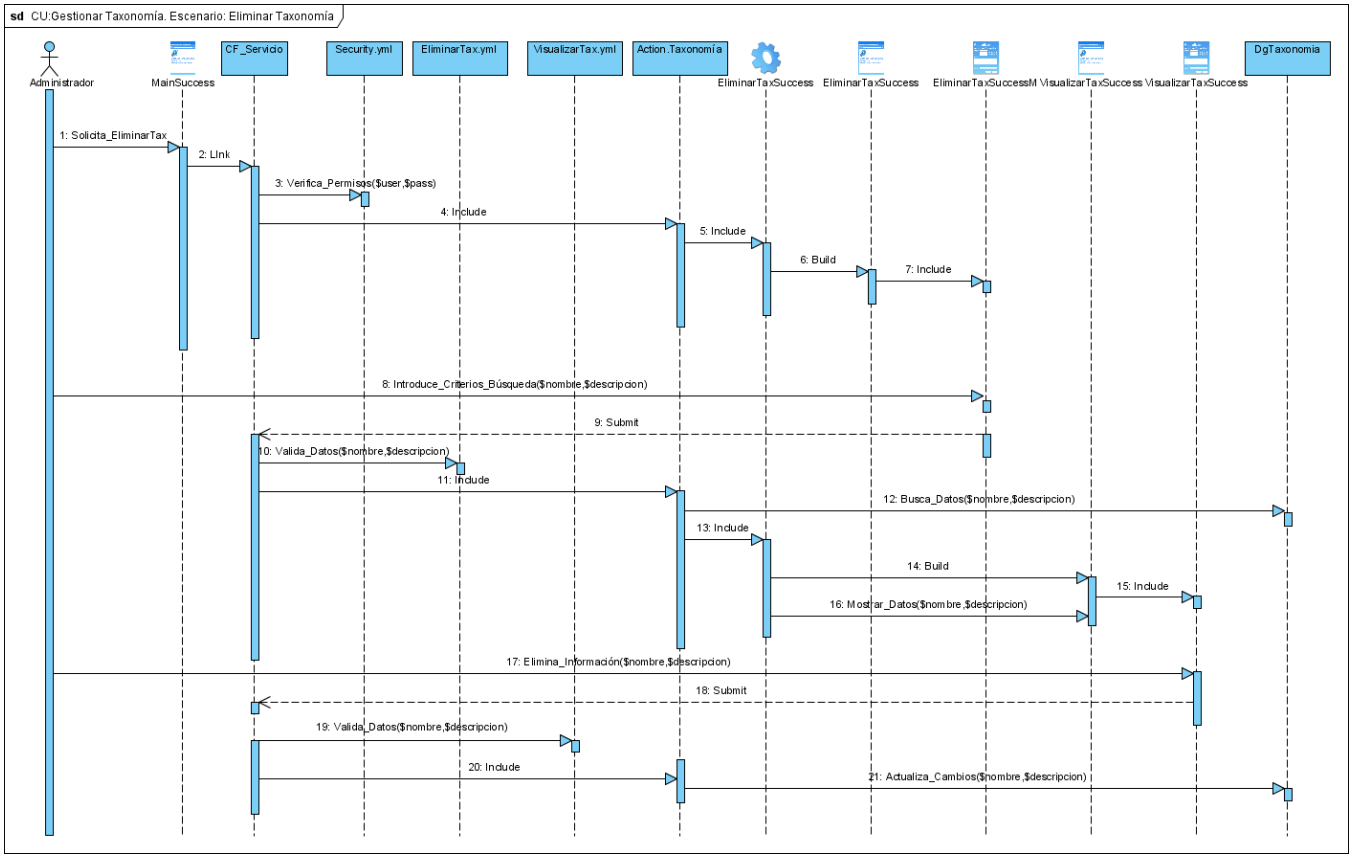
Escenario: Insertar Taxonomía



Escenario: Modificar Taxonomía

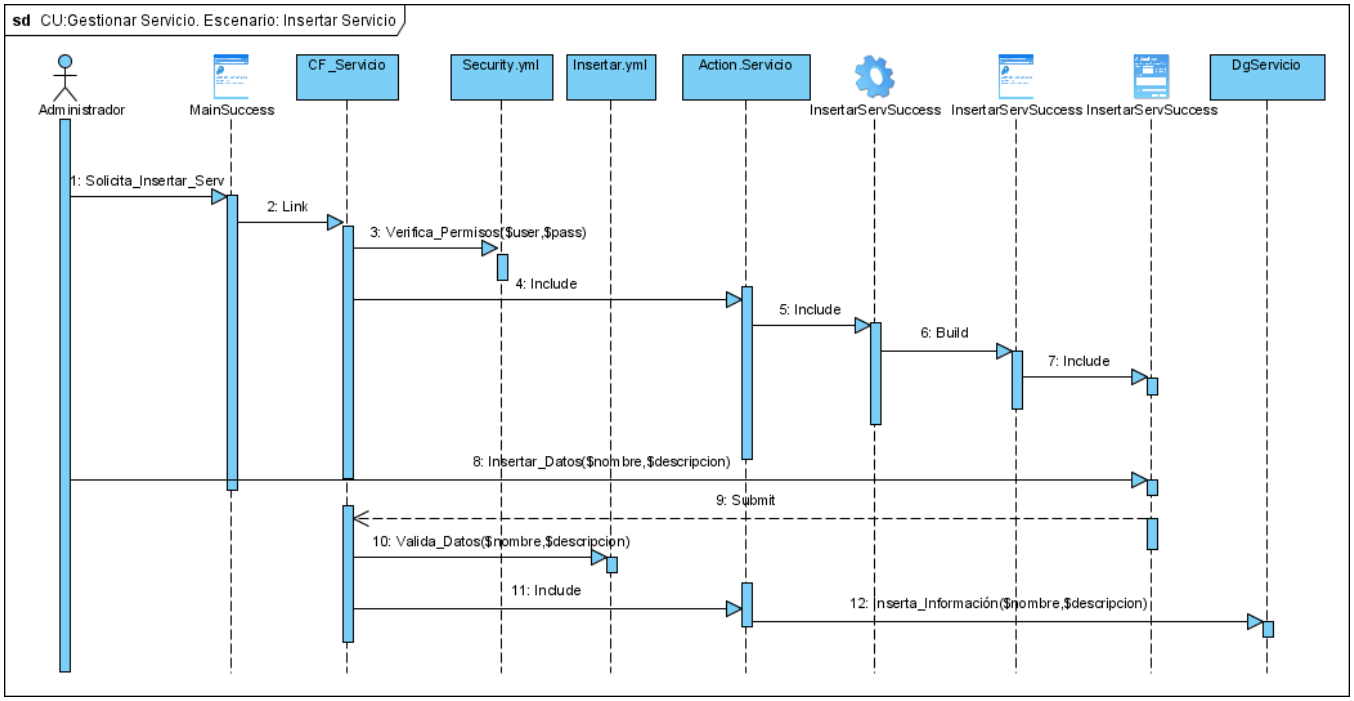


Escenario: Eliminar Taxonomía

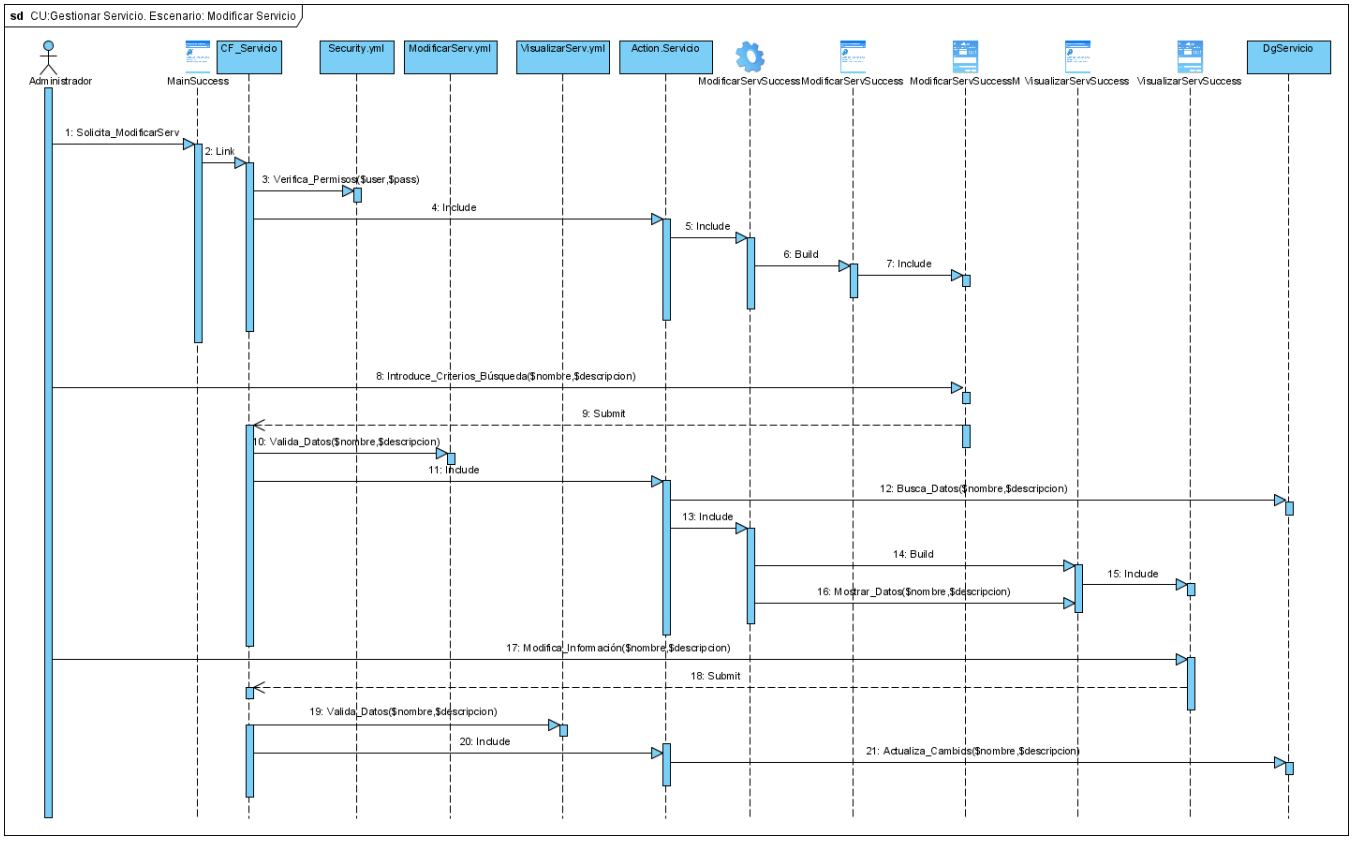


3.5.4 Diagrama de secuencia del Caso de Uso: Gestionar Servicios

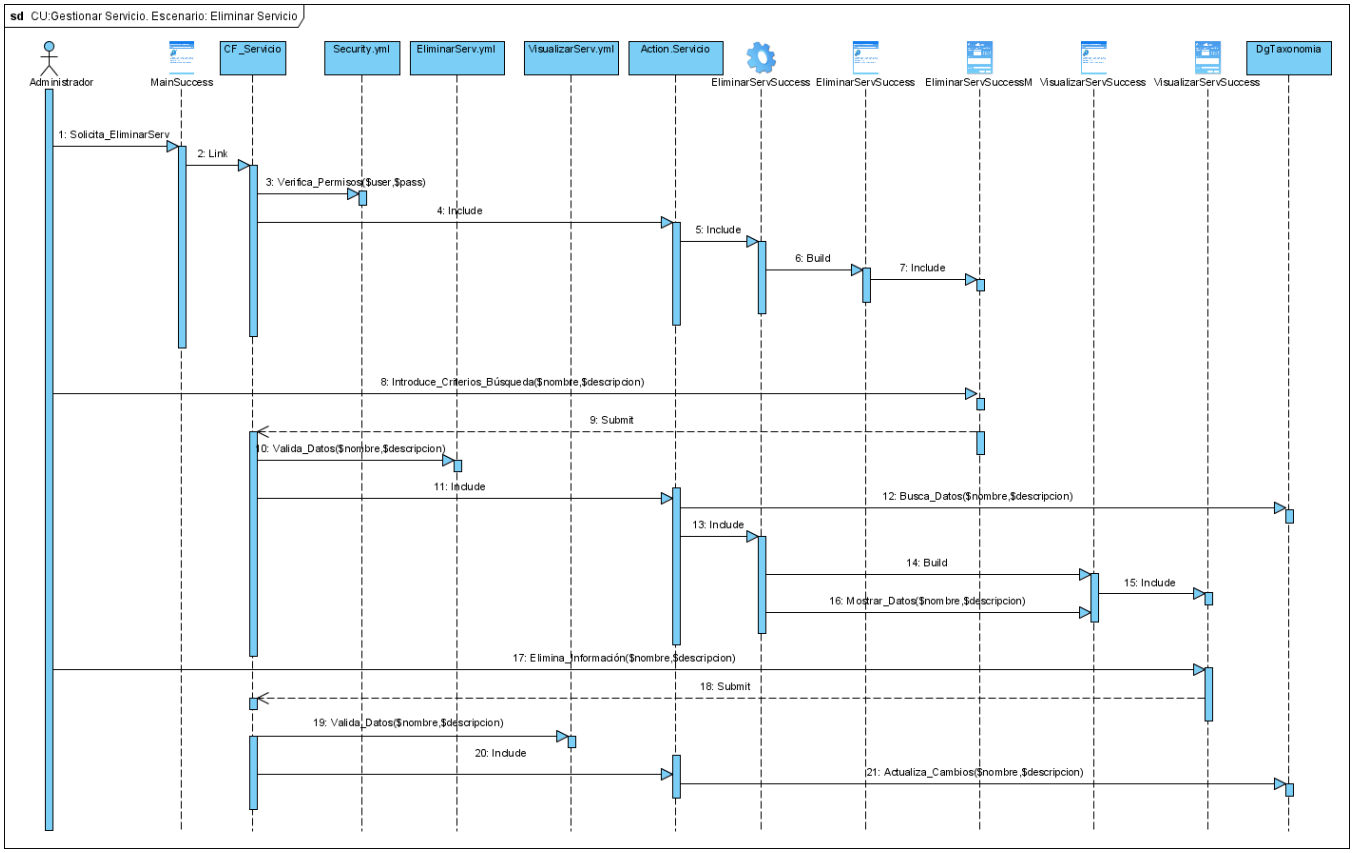
Escenario: Insertar Servicio



Escenario: Modificar Servicio

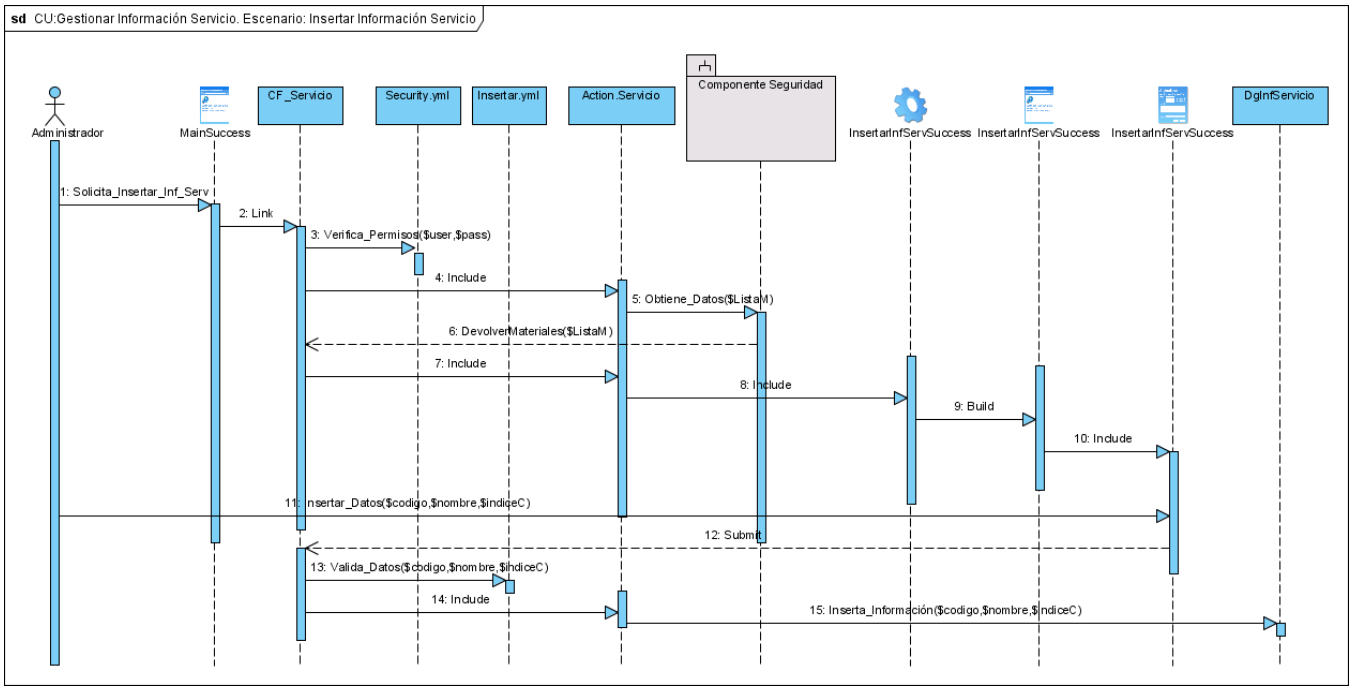


Escenario: Eliminar Servicio

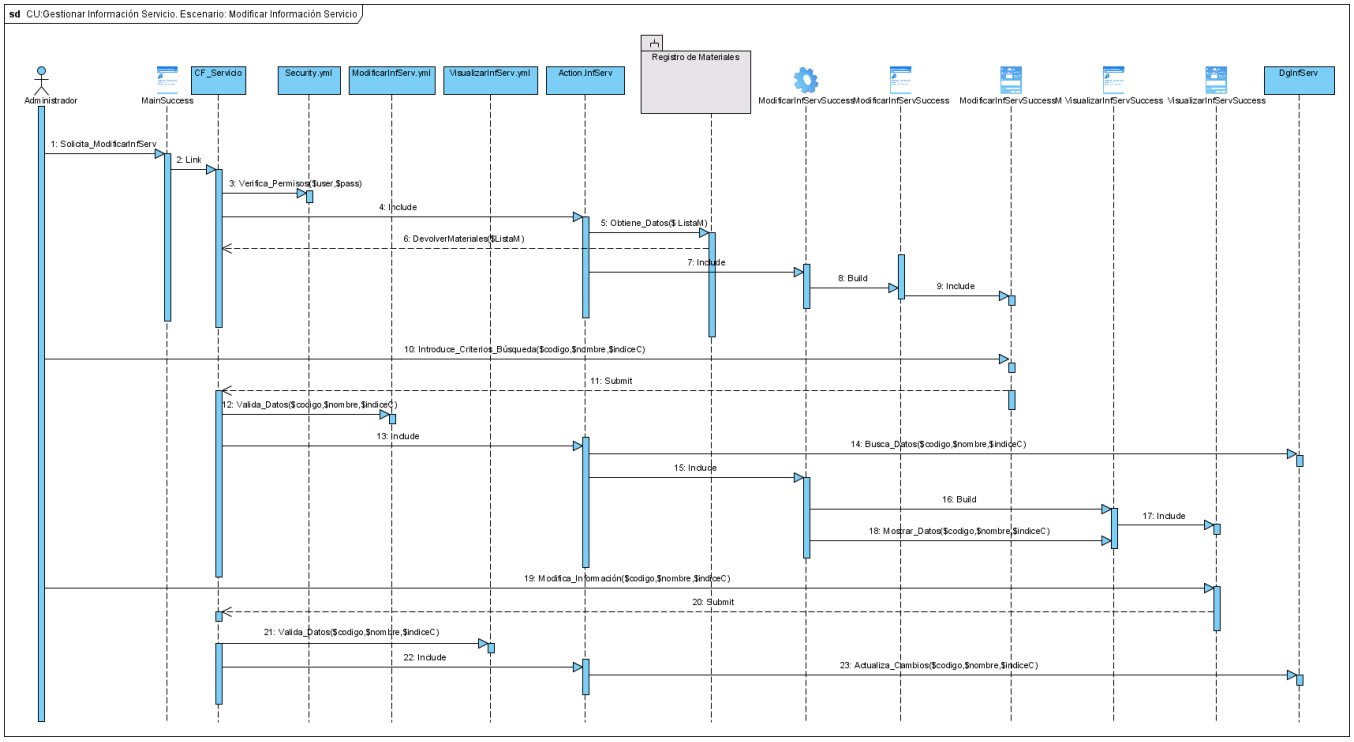


3.5.5 Diagrama de secuencia del Caso de Uso: Gestionar Información Servicios

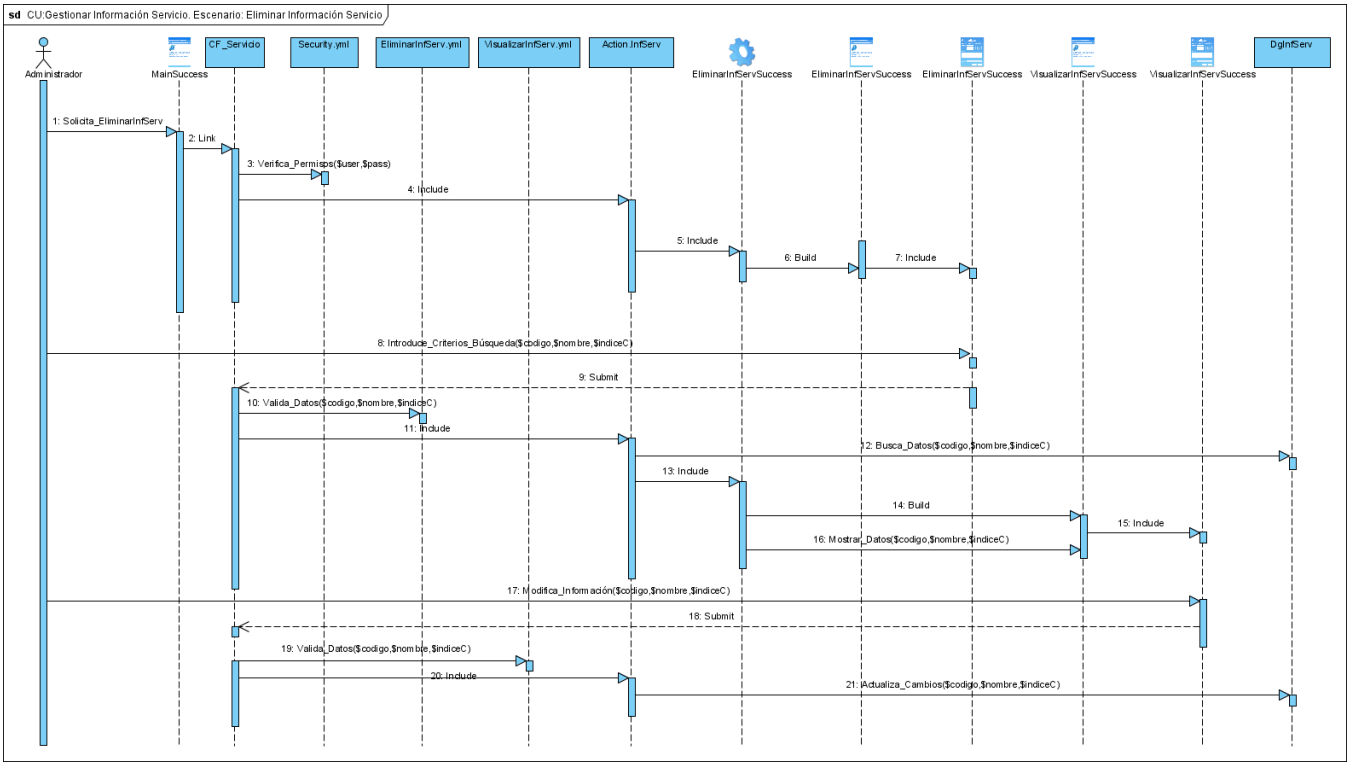
Escenario: Insertar Información Servicio



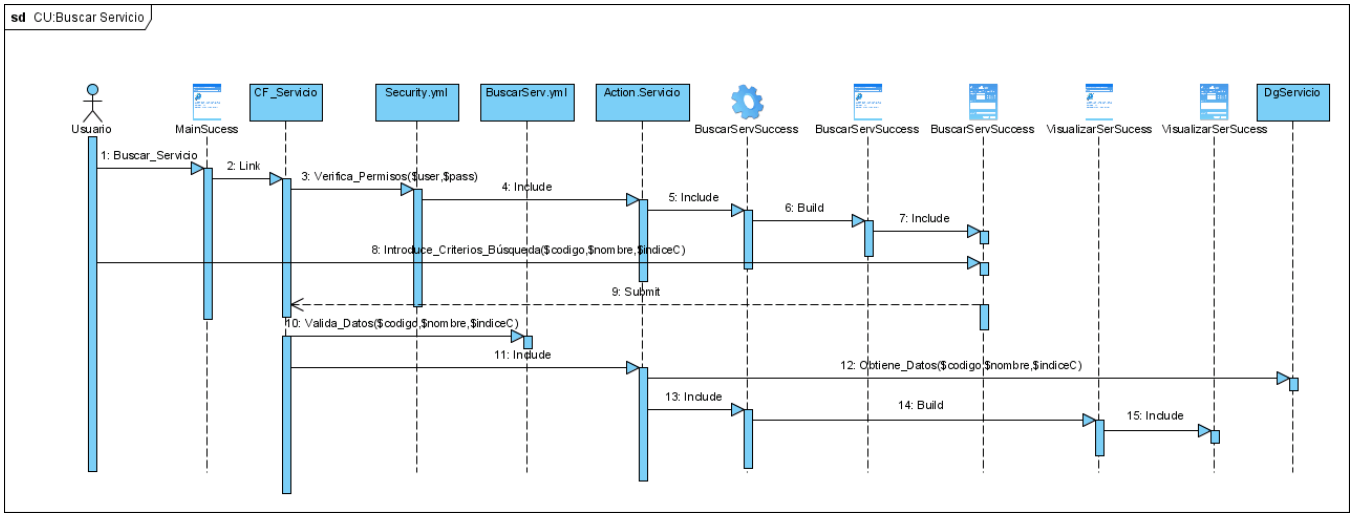
Escenario: Modificar Información Servicio



Escenario: Eliminar Información Servicio



3.5.6 Diagrama de secuencia del Caso de Uso: Buscar Servicio



3.6 Diagrama de clases persistentes

Las clases persistentes son las clases en una aplicación que implementan las entidades del problema del negocio. Son aquellas que están relacionadas con tablas en una base de datos. Representan almacenamientos de datos que persistirán más allá de la ejecución del software. Se propone como diagrama de clases persistentes el que se muestra en la siguiente figura:

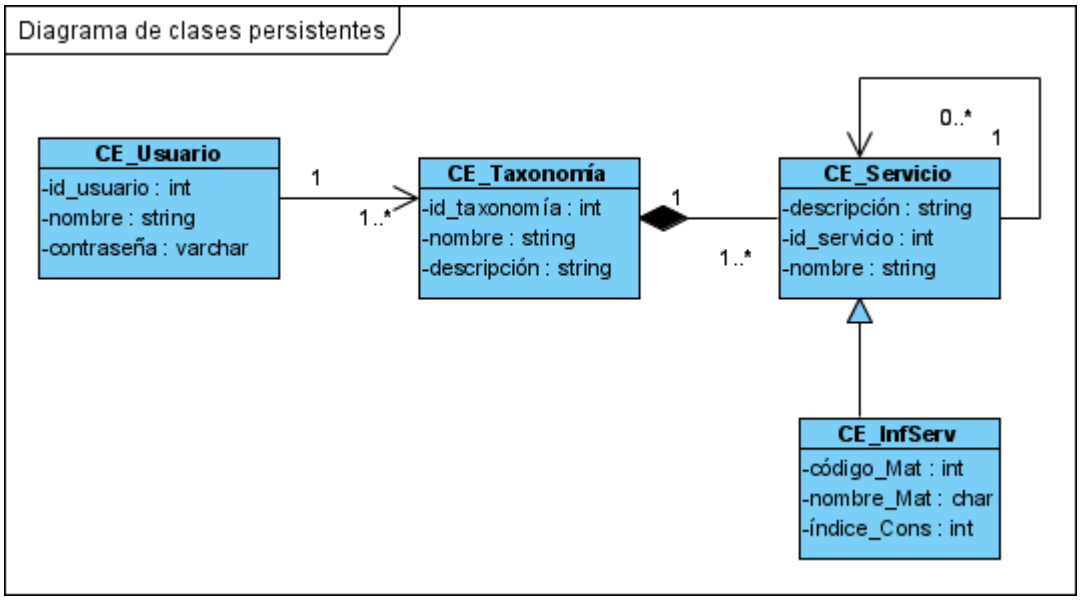


Fig.15 Diagrama de clases persistentes.

3.7 Modelo de datos

El concepto de modelo de datos es necesario para describir la estructura de una base de datos. El modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir, a distintos niveles de abstracción, los datos del universo de discurso o la estructura de una base de datos, que permiten construir una representación organizada de un sistema real. (46)

La figura que se muestra a continuación, representa el modelo de datos propuesto para el componente:

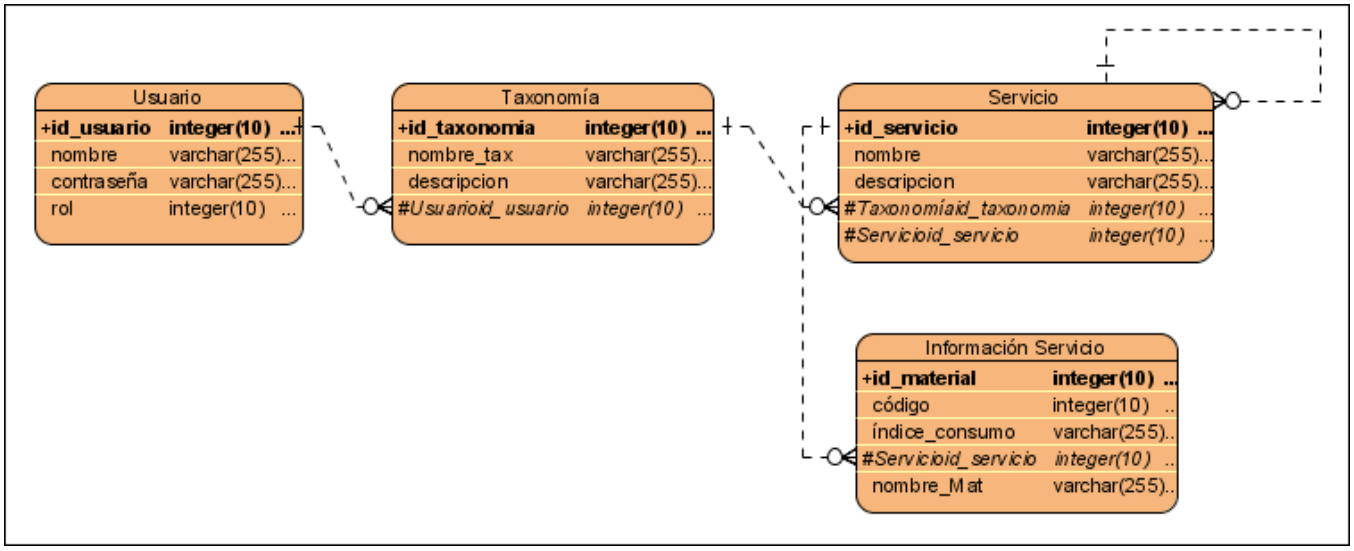


Fig.16 Modelo de datos.

3.8 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos y procesos.

Los diagramas de despliegue muestran la configuración en funcionamiento del sistema, incluyendo su hardware y software. (47)

En la siguiente figura se propone el diseño para el futuro despliegue de la aplicación:

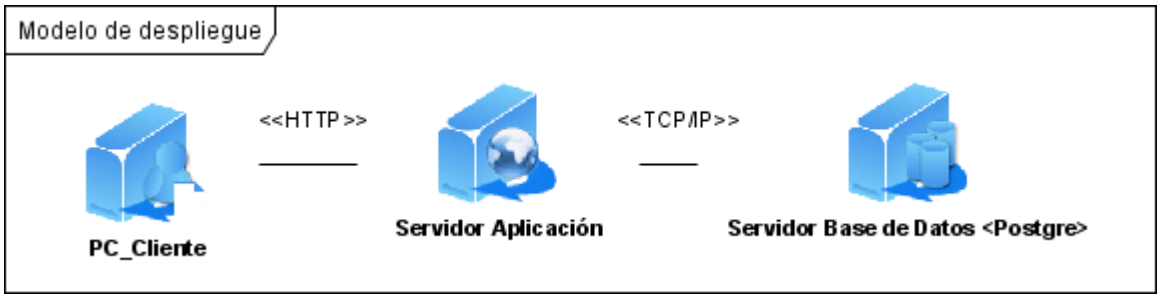


Fig.17 Diagrama de despliegue.

3.9 Especificación de los Servicios Web

Un Servicio Web es un artefacto clave en una arquitectura SOA. Ofrece una vista del proveedor de servicio y del cliente de servicio. Estas vistas se corresponden con dos mitades de un contrato que permiten la separación clara de la especificación de la implementación. La siguiente tabla describe cómo los distintos aspectos de una especificación de servicio afectan al proveedor y al cliente de la especificación. (48)

Rol	Especificación de interfaz	Especificación de comportamiento	Especificación de política
Proveedor	Informa sobre el conjunto de operaciones y mensajes al que el servicio debe responder. Todas las operaciones deben responder con los mensajes correctos.	Informa sobre el comportamiento que este servicio debe soportar. Si tal especificación es formal y completa, se puede probar una implementación para el cumplimiento de la especificación.	Informa sobre un conjunto de restricciones que se tienen en cuenta para la implementación, así como el conjunto de calidades de servicio que deben ejecutarse.
Cliente	Informa sobre el conjunto de operaciones que se pueden invocar.	Informa sobre los requisitos de protocolo que el cliente debe ejecutar (ordenación de operaciones, flujos de datos, etc.). También indica las operaciones que el cliente debe implementar para dar soporte a colaboraciones.	Informa las restricciones sobre las que el cliente debe estar consciente y los requisitos de seguridad. También identifica el servicio que un cliente puede obtener de un determinado proveedor.

3.10 Representación de los Servicios Web

Los Servicios Web permiten que las aplicaciones compartan información invocando funciones de otras aplicaciones, independientemente de cómo se hayan creado estas, cuál sea el sistema operativo o la plataforma en que se ejecutan y dispositivos utilizados para obtener acceso a ellas. Para ilustrar cómo se realiza la integración de los componentes Registro de Servicios, Registro de Materiales, Componente de Seguridad y los sistemas externos que deben consumir información del Registro de Servicio, se crea el Modelo de Servicios Web que se muestra a continuación:

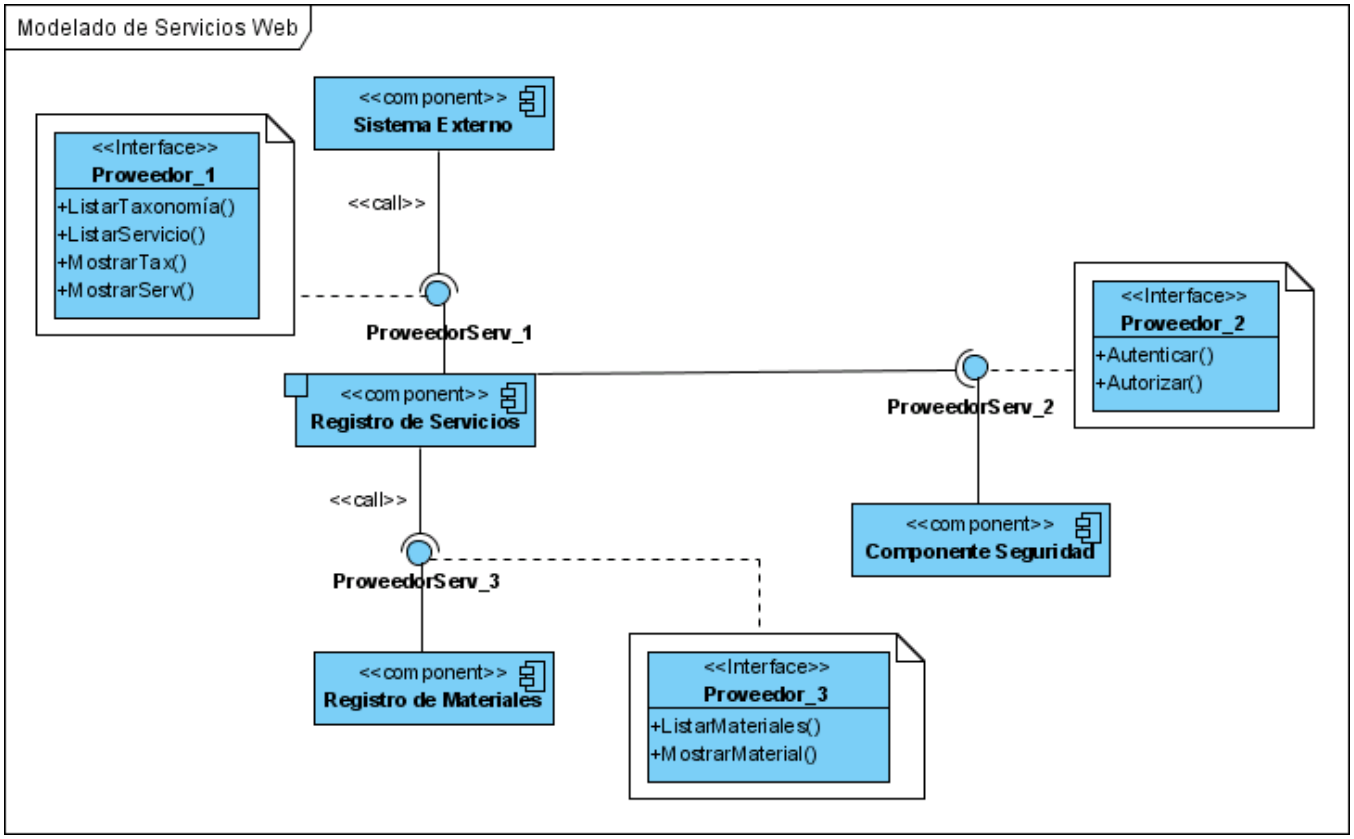


Fig.18 Representación de los Servicios Web.

Según el diagrama anterior, los componentes Registro de Servicios, Componente de Seguridad y Registro de Materiales; proveen interfaces Web Services para ofrecer su información a través de la red a los sistemas que invoquen una petición determinada. El Componente de Seguridad tiene publicado el método *Autenticar(),* necesario para que un usuario del Registro de Servicio pueda autenticarse con los derechos y privilegios establecidos para él. Para proporcionar al Sistema Externo cualquier

información que este necesite, el Registro de Servicios previamente debe acceder al Componente de Seguridad consumiendo el método *Autorizar(),* con el objetivo de verificar que la petición es válida.

El Registro de Materiales provee una interfaz Web Service con los métodos *ListarMateriales()* y *MostrarMaterial(),* que el Registro de Servicios consume para gestionar la información de sus servicios. Estos métodos permiten realizar la búsqueda de los materiales necesarios y mostrar la información correspondiente.

Cuando un usuario accede desde un Sistema Externo al Registro de Servicios, una vez autorizado a consumir sus servicios, éste puede realizar las operaciones *ListarTaxonomía(), ListarServicio(), MostrarTaxonomia()* y *MostrarServicio();* con la finalidad de listar las taxonomías y servicios y la información correspondiente a estos, según los privilegios que el Componente de Seguridad tenga registrado.

Para que este grado de interoperabilidad sea posible, es necesario que tanto el cliente como el servidor acuerden un protocolo mediante el cual se define cómo se invoca el servicio y los parámetros que se pasan, SOAP es usado con estos fines. Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja.

3.11 Conclusiones

En el desarrollo de este capítulo quedó definido el sistema a implementar, a través del modelo de clases de análisis y diseño, separando los diagramas en casos de uso para una mejor comprensión, representando las relaciones entre las clases interfaz, control y entidades. Además, se conformó el diagrama que representa la integración de los componentes que se proponen, utilizando Servicios Web.

CONCLUSIONES

Al dar culminación al trabajo se han cumplido el objetivo general propuesto, así como a las tareas de la investigación, ya que se diseñó un Componente basado en patrones de arquitectura y diseño, capaz de gestionar los servicios de cualquier negocio.

- Se realizó un estudio de las tendencias, técnicas y metodologías actuales para el modelado del componente. Se seleccionaron los patrones MVC y Composite para la arquitectura y diseño de la aplicación respectivamente.
- Se analizaron los aspectos teóricos conceptuales de los sistemas ERP, fundamentalmente del módulo que gestiona los servicios.
- Se obtuvo un conjunto de artefactos que son entradas fundamentales a tener en cuenta para los posteriores flujos de trabajo que propone la metodología RUP.
- Se propuso un diseño para el futuro despliegue de la aplicación.

RECOMENDACIONES

Se recomienda para el futuro desarrollo de este producto:

- Implementar el componente siguiendo el diseño propuesto.
- Concebir dentro de SAS la integración de este componente con otros sistemas.
- Continuar investigando funcionalidades y capacidades para agregar al componente.
- Proponer diseños que mejoren la calidad y el rendimiento de la aplicación en sus próximas versiones.

REFERENCIA BIBLIOGRÁFICA

1. **García, Alberto Otero.** Arquitecturas Empresariales. Orientación a Servicios (SOA) y Gestión de Procesos de Negocio (BPM) . [En línea] Febrero de 2007. http://www.aulesempresa.upc.edu/programes/everis.pdf.

2. *Igual a (1).*

3. *Igual a (1).*

4. **Río, Agustín Cernuda del.** Sistema de verificación de componentes software. [En línea] Febrero de 2002. http://tesis.uci.cu/document/tesisDoct/tesis.pdf.

5. **Szyperski, Clemens.** *Component Software – Beyond Object-Oriented Programming.* 1997.

6. **Mejia, Joel.** ERP (Entreprise Resource Planning)- Sistemas de Planeación de los Recursos de la Empresa Como el Nuevo Enfoque de Gestión. [En línea] http://www.monografias.com/trabajos14/enfoque-gestion/enfoque-gestion.shtml.

7. *Igual a (6).*

8. **Mejia, Joel.** ERP(ENTREPRISE RESOURCE PLANNING)- SISTEMAS DE PLANEACIÓN DE LOS RECURSOS DE LA EMPRESA COMO EL NUEVO ENFOQUE DE GESTIÓN. [En línea] 2004. http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/erpjoel.htm.

9. **Alexys Díaz, Juan Carlos González, Maria Elena Ruiz.** IMPLANTACIÓN DE UN SISTEMA ERP EN UNA ORGANIZACIÓN. [En línea] 2005. http://sisbib.unmsm.edu.pe/BibVirtualData/publicaciones/risi/n3_2005/a04.pdf.

10. *Igual a (9).*

11. *Igual a (9).*

12. Oracle ’s JD Edwards EnterpriseOne. [En línea] http://h71028.www7.hp.com/erc/downloads/4aa1-5107enw.pdf.

13. *Igual a (12) .*

14. JD Edwards EnterpriseOne. [En línea] 2008. http://www.oracle.com/global/es/products/applications/jdedwards-enterprise-one.html.

15. *Igual a (14) .*

16. **Ochoa, Emmanuel Sánchez.** VENTAJAS Y DESVENTAJAS DE ERP. [En línea] 2004. http://www.gestiopolis.com/canales2/gerencia/erpemma.htm.

17. Cuba lanza portal para el comercio electrónico nacional . [En línea] 2008. http://mensual.prensa.com/mensual/contenido/2001/03/06/hoyenlared.htm.

18. **Alcubilla, Julio César.** SOA un nuevo modelo de arquitectura en TI. [En línea] Enero de 2007. http://www.tecnologiahechapalabra.com/datos/soluciones/gerencia/articulo.asp?i=522.

19. **García, Pablo.** Introducción a la arquitectura orientada a servicios. [En línea] Noviembre de 2007. http://www.fidesol.org/portal//index.php?option=com_content&task=view&id=13&Itemid=1.

20. **Barco, Antonio.** Principios de la orientación a servicios. [En línea] 2007. http://arquitecturaorientadaaservicios.blogspot.com/.

21. Definición de SOA. [En línea] Mayo de 2008. http://www.zonatorrida.biz/?p=7.

22. *Igual a (21).*

23. Procesos de ingeniería de software. [En línea] http://www.liderdeproyecto.com/metodologias/index.html.

24. *Igual a (23).*

25. **Hernandis, José Alberto.** Visual Paradigm for UML. [En línea] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

26. **Caballero, Ismael.** Una herramienta CASE: Visual Paradigm. [En línea] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

27. **Gracia, Joaquin.** UML: Diagramas UML. ¿Qué es UML? [En línea] Mayo de 2005. http://www.ingenierosoftware.com/analisisydiseno/uml.php.

28. **Helman, Dean.** Model-View-Controller. [En línea] http://ootips.org/mvc-pattern.html.

29. **Dodero, Juan Manuel.** Patrones estructurales: Composite. [En línea] 2003. http://www.dei.inf.uc3m.es/docencia/p_s_ciclo/tdp/curso0203/apuntes/composite.pdf.

30. *Igual a (29).*

31. **García, Luis Fernádez.** Patrón Composite. [En línea] http://kybele.escet.urjc.es/documentos/SI/Patrones/07_Composite.pdf.

32. **Potiencer, Fabien.** Symfony, la guia definitiva. [En línea] Diciembre de 2007. http://www.librosweb.es .

33. *Igual a (32).*

34. Web Services Description Language. [En línea] 2007. http://atc.ugr.es/pedro/tutoriales/cursos/curso_soap/wsdl.htm.

35. *Igual a (34).*

36. **Rovira, Rubén.** ¿Qué es un Web Service? [En línea] http://www.mug.org.ar/FoxProGufa/ArticFox/239.aspx.

37. **Cantallops, Victor Manuel Avila.** SISTEMA PARA LA PLANIFICACIÓN DE MATERIALES GASTABLES DE USO MÉDICO (MÓDULO DE PLANIFICACIÓN) . 2007.

38. *Igual a (37).*

39. **Neyra, Calderon.** Servicio al cliente. [En línea] 2002. http://www.monografias.com/trabajos11/sercli/sercli.shtml.

40. **A.Ville., Claude.** Taxonomía. [En línea] 1993. http://www.monografias.com/trabajos5/taxo/taxo.shtml.

41. **Dapena., MSc. Martha D. Delgado.** Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos. [En línea] http://www.inf.udec.cl/revista/ediciones/edicion8/Rbc.pdf.

42. **T., José Camilo Daccach.** Requerimientos funcionales. [En línea] 2007. http://www.deltaasesores.com/prof/PRO449.html.

43. **21, Informática Siglo.** Documento de Especificación Requerimientos No funcionales. [En línea] http://www.minprotecconsocial.gov.co/VBContent/library/documents/DocNewsNo18DocumentNo1.PDF

44. **Annia Arencibia Morales, Karel Gómez Velázquez, Leonardo González González.** Centro de Control para el Sistema de Información para la Salud.

45. *Igual a (44).*

46. **Colina, Instituto Tecnológico de.** Definicion de modelo de datos. [En línea] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm.

47. Ingeniería de Software II. [En línea] http://teleformacion.uci.cu/mod/resource/view.php?id=21514.

48. Ayuda Rational 2007.

BIBLIOGRAFÍA

1. Ayuda Rational 2007.
2. Centro de Control para el Sistema de Información para la Salud [Tesis] / aut. Annia Arencibia Morales Karel Gómez Velázquez, Leonardo González, González.
3. Comercialización de Software [Tesis] / aut. Espinosa Yulkeidi Martínez. - 2006.
4. Component Software – Beyond Object-Oriented Programming. [Libro].
5. http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.
6. http://arquitecturaorientadaaservicios.blogspot.com/.
7. http://atc.ugr.es/pedro/tutoriales/cursos/curso_s .
8. http://atc.ugr.es/pedro/tutoriales/cursos/curso_soap/wsdl.htm.
9. http://biblioteca.itesm.mx/cgi-bin/nav/salta?cual=bases:17 .
10. http://desarrolloerp.blogspot.com/2005/09/erp-cubano-opensource.html .
11. http://h71028.www7.hp.com/erc/downloads/4aa1-5107enw.pdf.
12. http://iaaa.cps.unizar.es/docencia/SW.html.
13. http://iaaa.cps.unizar.es/docencia/SW.html.
14. http://informatica.uv.es/estguia/ATD/apuntes/teoria/presentaciones/ModeloRelacional.pdf
15. http://kybele.escet.urjc.es/documentos/SI/Patrones/07_Composite.pdf.
16. http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm .
17. http://latinamerica.infor.com/soluciones/erp/servicemanagement/.
18. http://mensual.prensa.com/mensual/contenido/2001/03/06/hoyenlared.htm.
19. http://ootips.org/mvc-pattern.html .
20. http://sisbib.unmsm.edu.pe/BibVirtualData/publicaciones/risi/n3_2005/a04.pdf.
21. http://sisbib.unmsm.edu.pe/BibVirtualData/Tesis/Human/Garcia_B_I/introduccion.pdf.
22. http://technet.microsoft.com/es-es/library/ms19 .
23. http://technet.microsoft.com/es-es/library/ms190983.aspx ..
24. http://teleformacion.uci.cu/mod/resource/view.php?id=21514 .
25. http://tesis.uci.cu/document/tesisDoct/tesis.pdf .
26. http://www.aulesempresa.upc.edu/programes/everis.pdf.
27. http://www.cibernetia.com/manuales/servicios_web/5_uddi.php.
28. http://www.cio.com/research/erp/edit/erpbasics.html.
29. http://www.dei.inf.uc3m.es/docencia/p_s_ciclo/tdp/curso0203/apuntes/composite.pdf.
30. http://www.deltaasesores.com/prof/PRO449.html.
31. http://www.deltaasesores.com/prof/PRO449.html .

32. http://www.desarrolloweb.com/articulos/1589.ph.
33. http://www.desarrolloweb.com/articulos/1589.php .
34. http://www.emb.cl/gerencia/articulo.mv?sec=12&num=150 .
35. http://www.estrategia.net/estrategia/cs21/ana.htm.
36. http://www.etsimo.uniovi.es/~feli/CursoMDT/Tema_1.pdf .
37. http://www.fidesol.org/portal//index.php?option=com_conte&task=view&id=13&Itemid=1 .
38. http://www.gestiopolis.com/canales2/gerencia/erpemma.htm.
39. http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/erpjoel.htm .
40. http://www.inf.udec.cl/revista/ediciones/edicion8/Rbc.pdf .
41. http://www.ingenierosoftware.com/analisisydiseno/uml.php .
42. http://www.librosweb.es.
43. http://www.liderdeproyecto.com/metodologias/index.html .
44. http://www.minproteccionsocial.gov.co/VBeContet/library/documents/DocNewsNo16758DocumentNo5401.PDF .
45. http://www.mitecnologico.com/Main/DefinicionModeloDeDatos.
46. http://www.monografias.com/trabajos11/sercli/sercli.shtml .
47. http://www.monografias.com/trabajos14/enfoque-gestion/enfoque-gestion.shtml.
48. http://www.monografias.com/trabajos14/impactosistema/impactosistema.shtml.
49. http://www.monografias.com/trabajos5/taxo/taxo.shtml.
50. http://www.mug.org.ar/FoxProGufa/ArticFox/239.
51. http://www.mug.org.ar/FoxProGufa/ArticFox/239.aspx.
52. http://www.oracle.com/global/es/products/applications/jdedwards-enterprise-one.html. http://www.ptesa.com/index.php?option=com_content&task=view&id=79&Itemid=68.
53. http://www.sap.com/argentina/company/press/press.epx?pressid=6349 [Libro].
54. http://www.tecnologiahechapalabra.com/datos/soluciones/gerencia/articulo.asp?i=522.
55. http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
56. http://www.zonatorrida.biz/?p=7.
57. http://www-306.ibm.com/software/uy/info/topic/openenvironment/soa/.