Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema de Integración Bancaria (SIB), desarrollo del subsistema de gestión de solicitudes y pagos. Módulo administración, Módulo reportes y Módulo clientes.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor:

Yanalia Gómez Hernández

Tutor:

Johann Rodríguez Hernández

La Habana, Cuba.
Junio de 2011

Saber no es suficiente: tenemos que aplicarlo. Tener voluntad no es suficiente: tenemos que implementarla.

Goethe

DECLARACIÓN DE AUTORÍA

Declaración de Autoría

Yanalia Gómez Hernández declara que es la autora de la presente tesis y reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yanalia Gómez Hernández	Ing. Johann Rodríguez Hernández
_____	_____
Firma del Autor	Firma del Tutor

Agradecimientos:

***A mis padres**, por mostrarme el camino correcto y apoyarme en todas las decisiones que he tomado. Por hacerme sentir tan especial.*

***A mi hermano**, por ser la persona maravillosa que es y estar siempre presente en mi vida.*

***A mis amigos (Rache, Sahyli, Annie, Yuniel e Ivan),** por ser partícipes de los mejores momentos que he vivido estos 5 años y tener una de los valores que más admiro en las personas, la sinceridad.*

***A mis compañeros del laboratorio,** por estar a mi lado todos los días, en especial a **Rubén, Fernando e Isidro**.*

***A la profesora Alina,** por ser mi tutora postiza y la estrella que me guió al final del camino.*

***Al profesor Joe,** por su gran apoyo y solidaridad.*

A la Revolución, a Fidel y a la Universidad de las Ciencias Informáticas, por dejarme formar parte de este proyecto.

Gracias a todas las personas que me han ayudado a construir este sueño.

Dedicatoria

***A mi mamá y a mi papá**, porque siempre me han apoyado y dado mucho amor. Por ser lo más preciado que tengo en la vida.*

***A mi hermano**, porque es mi ejemplo a seguir desde pequeña y ser tan especial.*

***A mi abuela**, por darme tanto cariño a pesar de sus locuras, porque sin ella mi vida dejaría de tener una parte maravillosa.*

RESUMEN

Resumen

La informatización de las tareas de una empresa, es un elemento fundamental para la superación y desarrollo de la misma. Por eso, algunas organizaciones basan su crecimiento en la aplicación de soluciones informáticas que gestionen los diferentes procesos que se llevan a cabo para el correcto funcionamiento de la misma, marcando una diferencia muy grande en su éxito. Servicio Administrativo de Identificación, Migración y Extranjería (SAIME) es una empresa cuyo objetivo principal es brindar servicios relacionados a la identificación indubitable de las personas naturales que habitan en la República Bolivariana de Venezuela. En estos momentos, SAIME necesita de un sistema que logre la generación de reportes que reflejen la cantidad recaudada en el día por servicios y solicitudes liquidadas, así como de los servicios vendidos por su tipo o estado, la gestión de las actividades de administración y la comunicación online con sus clientes. Por tal motivo, el objetivo principal del presente trabajo de diploma se centra en la implementación de tres módulos que den respuesta a estas necesidades de la empresa. El documento recoge un estudio sobre otros sistemas de gestión empresarial en el ámbito nacional e internacional, se detallan las tecnologías, lenguajes y herramientas utilizadas, la descripción de la propuesta de solución, la información de las clases y la validación de la solución mediante las pruebas funcionales.

Palabras clave: administración, clientes, empresa, gestión, interacción, reportes.

Índice

Índice

Índice

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Introducción

La Universidad de las Ciencias Informáticas (UCI) tiene en sus manos una misión muy importante, desarrollar la industria de *software* cubano, para lograrlo, se realiza la integración con otros países; uno de ellos es la República Bolivariana de Venezuela (VNZ), con la cual se realizan varios proyectos.

La informatización de diferentes áreas en este país ha contribuido de forma significativa a que procesos vitales se lleven a cabo con mayor productividad. La dirección de gestión administrativa del Servicio Administrativo de Identificación, Migración y Extranjería (SAIME) ha sido pionero en revolucionar la imagen del sector público, gracias en gran parte a los sistemas informáticos.

SAIME, es un organismo adscrito al Ministerio del Interior para las Relaciones Interiores y de Justicia de la República Bolivariana de Venezuela, cuyo objetivo principal es brindar servicios relacionados a la identificación indubitable de las personas naturales que habitan en el territorio nacional. Para su funcionamiento cuenta con varias direcciones dentro de las cuales está: la Dirección de Gestión Administrativa, que tiene un grupo dedicado fundamentalmente a todo lo relacionado con el chequeo del estado de cuentas; los reportes de recaudación periódica de las oficinas y el control de los *voucher* (comprobante de pago bancario).

Actualmente, los ciudadanos que desean tener un servicio de SAIME deben obtener un *voucher* que se le entregará al hacer el pago en un banco recaudador y se le comprobará en las Oficinas de Migración y Extranjería donde se le prestará el servicio solicitado; todos estos procesos de recaudación de pagos de SAIME se realizan de forma manual, trayendo consigo inconvenientes. No hay una manera efectiva de comprobar la legitimidad del *voucher* presentado por el cliente, habiéndose detectado irregularidades en algunos casos, y el cliente necesita hacer un depósito bancario y obtener el *voucher* asociado, de forma personal, para cada trámite que va a realizar. Los procesos de la dirección de gestión administrativa no están informatizados, por lo que se hace difícil realizar la administración de los datos de la empresa. No existe ninguna forma de conocer cuáles son las actividades diarias que se realizan en la empresa, como por ejemplo, la cantidad de dinero recaudado y la cantidad de servicios vendidos.

INTRODUCCIÓN

Para darle solución a los problemas que genera esta comunicación física, se crea en el Centro de Identificación y Seguridad Digital (CISED) de la UCI, una solución informática de notificación de pago bancario entre el SAIME y sus bancos recaudadores o Sistema de Integración Bancaria (SIB), herramienta que facilitará los procesos de pago, recaudación y conciliación a todos los entes involucrados en dichos procesos. Como parte del SIB, se desarrollará un subsistema de gestión de solicitudes y pagos que necesita:

- Lograr la generación de reportes que reflejen la cantidad recaudada en el día por servicios y solicitudes liquidadas, así como de los servicios vendidos por su tipo o estado.
- Lograr la gestión de los servicios que brinda el proveedor SAIME, los bancos recaudadores, las cuentas recaudadoras, tipos de pagos, clientes, proveedores, avisos promocionales, mensajes de correo y las categorías de estos.
- Lograr la comunicación de los clientes con el proveedor de servicio SAIME.

Queda definido como **problema científico** a resolver: ¿Cómo garantizar la gestión de los procesos de reportes y administración del subsistema de gestión de solicitudes y pagos del SIB, y la comunicación de los clientes con SAIME?

Se tiene como **objeto de estudio:** el proceso de integración bancaria para la gestión de los procesos de administración, reportes e interacción con los clientes.

El **campo de acción** queda definido como: el proceso de integración de SAIME con sus bancos recaudadores para la gestión de los procesos de administración, reportes e interacción con los clientes.

Se plantea como **objetivo de la investigación**: Desarrollar los módulos de Administración, Clientes y Reportes como parte del subsistema de gestión de solicitudes y pagos del SIB.

Para lograr el objetivo planteado, se definen los siguientes **objetivos específicos**:

1. Caracterizar el proceso de recaudación de pagos de SAIME.
2. Estudiar sistemas que utilizan entornos web para la gestión de información y creación de reportes, además de las tecnologías y herramientas a utilizar.
3. Desarrollar los módulos de administración, clientes y reportes como parte del

subsistema de gestión de solicitudes y pagos del SIB.

4. Probar los módulos de administración, clientes y reportes como parte del subsistema de gestión de solicitudes y pagos del SIB.

Hipótesis:

Si se desarrollan los módulos de administración, clientes y reportes como parte del subsistema de gestión de solicitudes y pagos del SIB, se garantizará la gestión de los procesos de reportes y administración del subsistema de gestión de solicitudes y pagos del SIB, y la comunicación de los clientes con el proveedor del servicio SAIME.

Variables:

Independiente: Desarrollo de los módulos de administración, clientes y reportes.

Dependiente:

- Gestión de los procesos de reportes.
- Gestión de los procesos de administración.
- Comunicación de los clientes con el proveedor del servicio SAIME.

Variable	Dimensión	Indicadores
Desarrollo de los módulos de Administración, Clientes y Reportes.	Factibilidad	Tiempo de desarrollo: • Extenso – más de 7 meses. • Moderado – de 4 a 7 meses • Breve – menos de 4 meses
		Esfuerzo: • Alto • Moderado • Despreciable
Gestión de los procesos de reportes.	Chequeo de datos	Cantidad de no conformidades: • De 11 a 20 – Mal • De 6 a 10 – Regular • De 0 a 5 – Bien
Gestión de los procesos de administración.	Chequeo de datos	Cantidad de no conformidades: • De 11 a 20 – Mal • De 6 a 10 – Regular • De 0 a 5 – Bien
	Seguridad	Complejidad: • Alta

		• Media • Baja
		Control • Bueno • Malo
Comunicación de los clientes con el proveedor del servicio SAIME.	Chequeo de datos	Cantidad de no conformidades: • De 11 a 20 – Mal • De 6 a 10 – Regular • De 0 a 5 – Bien

Tabla 1: Operacionalización de las variables (Fuente: elaboración propia)

Para la realización de la aplicación se establecen las siguientes tareas de investigación:

- Describir el funcionamiento de la recaudación de pagos de SAIME.
- Analizar tipos de pagos realizados en bancos.
- Analizar otras entidades cubanas o extranjeras que tengan sistemas que gestionen las actividades de administración, que emitan reportes y que se comuniquen con los clientes.
- Realizar un estudio del estado del arte relacionado con la comunicación entre sistemas bancarios y otros sistemas informáticos.
- Estudiar los requisitos funcionales y no funcionales que responden a los módulos de administración, reportes y clientes del subsistema de gestión de solicitudes y pagos del SIB.
- Describir casos de uso a desarrollar.
- Realizar una investigación para establecer herramientas, metodología y lenguajes de programación que satisfaga las características que debe tener el sistema informático.
- Realizar análisis y diseño de los casos de uso para lograr una mejor comprensión del comportamiento de los mismos.
- Realizar la implementación de los casos de uso para lograr el desarrollo de los módulos administración, reportes y clientes.
- Realizar las pruebas al sistema informático para lograr una mayor calidad en su funcionamiento.
- Realizar la validación de las variables de la hipótesis.

Los métodos de la investigación quedan definidos como:

Métodos teóricos:

- Analítico - sintético: se consultarán y estudiarán varias bibliografías para definir una solución informática. Se realizará un resumen de los aspectos más importantes.
- Modelación: se realizarán modelos para definir cómo quedarán establecidos los procesos que se automatizarán.

Métodos empíricos:

- Observación: se realizará un estudio del proceso de recaudación de pagos de SAIME.
- Medición: se realizará la medición de las variables y se registrarán los resultados.

El presente trabajo se organiza de la siguiente forma:

Capítulo 1 Fundamentación teórica: se exponen los conceptos fundamentales que sustentan la investigación. Incluye el análisis de la información existente acerca de las tendencias y tecnologías que existen en la actualidad. Se seleccionan las herramientas y tecnologías para desarrollar la solución.

Capítulo 2 Propuesta de solución: se identifican los requerimientos funcionales y no funcionales, así como los casos de uso. Se presentan las fases de Análisis y Diseño definidas por la metodología que se establezca para dar solución al problema científico.

Capítulo 3 Implementación: se da cumplimiento a los planes trazados a través de las fases análisis y diseño, y se desarrolla la solución diseñada.

Capítulo 4 Pruebas a la solución: se realizan las pruebas al sistema informático y se hacen las mediciones de las variables.

Capítulo 1
Fundamentación Teórica

1.1 Introducción

En la actualidad, gran cantidad de empresas emplean nuevas formas para facilitar y mejorar su modo de operación a través de Internet. Un ejemplo de esto es la utilización de aplicaciones web, las cuales brindan múltiples beneficios, tales como: la disponibilidad de la información de las empresas desde cualquier parte con acceso a Internet, brindar mejor servicio a los clientes, la mejora de la comunicación interna, el acceso seguro de la información, accesibilidad permanente, eliminación de las barreras de tiempo y reducción de costos de operaciones, todo esto marca una gran diferencia en el éxito de las empresas.

Estas aplicaciones web también gestionan información logrando mejorar la administración de los datos de las mismas y ayudando en el desarrollo de las empresas. Uno de los problemas que solucionan estos sistemas de gestión es la transcripción de datos en forma manual, pues se pueden cometer errores accidentales que le quitan credibilidad a la información de gestión.

1.2 Conceptos asociados al dominio del problema

Sistema de integración

Los sistemas de integración, son sistemas informáticos que integran varias organizaciones permitiendo una comunicación entre ellas. Mediante los sistemas de integración se consigue una mejora de los servicios, se provee de nuevos canales de comunicación y se logra un ahorro en las operaciones.

Módulo

Un módulo es una parte de un programa o sistema informático donde se realizan algunas de sus funcionalidades. Es independiente, por lo que no afecta a todo el sistema si se realiza un cambio en su programación por un error o por rediseño del algoritmo, también esta característica permite que sea más fácil trabajar en él.

Sistema de gestión

Un sistema de gestión es una aplicación informática que se usa para la gestión, y mejora continua de las políticas, los procedimientos y procesos de la organización. Éste ayuda a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos y el enfoque centrado en la gestión.

Los sistemas informáticos de gestión son capaces de resolver problemas, realizar procesos administrativos y contables, así como la gestión de catálogos de productos, incluyendo cualquier proceso de negocio que requiera de una sistematización y organización.

1.3 Aplicación web

Una aplicación web (*web-based applications*) es un tipo especial cliente/servidor donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (http) están estandarizados y no han de ser creador por el programador de aplicaciones. (Mora, 2002)

Estas aplicaciones tienen como ventaja que poseen elementos que permiten una comunicación activa entre el usuario y la información, admitiendo el acceso recíproco a la misma, ya que la página responderá a cada una de sus acciones.

Las aplicaciones web crean páginas en el formato HTML o XHTML que son ejecutadas por navegadores web, usan lenguajes que son interpretados en el cliente y en ocasiones *plugins* para agregar elementos dinámicos a la interfaz de usuario. Estas aplicaciones tienen una estructura de tres capas: el navegador que la ejecuta es la primera; la segunda, es la que gestiona toda la información, usando una tecnología web dinámica, como por ejemplo, PHP o ASP.NET; y la tercera capa de acceso a la base de datos donde se almacena toda la información. El navegador se comunica con la segunda capa, quien atiende las peticiones usando consultas que ejecuta en la base de datos y mostrando la respuesta en una interfaz de usuario. En la actualidad existen compañías Proveedores de Aplicaciones de Servicio (SAP) que desarrollan *software* como servicio; éstas proveen acceso al *software* por la web; en el caso de las aplicaciones de escritorio, brinda la posibilidad de desarrollarla totalmente nueva o adaptarla para ser usada con una interfaz web. También existen aplicaciones web que necesitan la instalación de un *software* para

poder ejecutarse, pues este enriquece la aplicación proveyendo funcionalidades como el uso de webcam y la captura de videos e imágenes. Para el desarrollo de aplicaciones web en el servidor, se utilizan varios lenguajes de programación como PHP, Java, JavaScript, Perl, Ruby, Python, HTML y XML.

Algunas de las ventajas que ofrecen las aplicaciones web sobre las aplicaciones de escritorio por lo que se recomienda su uso son:

- Pueden ejecutarse en cualquier navegador eliminado los problemas de compatibilidad.
- No es necesario descargar o instalar programas para realizar tareas sencillas, lo cual ahorra tiempo.
- No ocupan espacio en las computadoras personales de los usuarios ni consumen muchos recursos y siempre se usa la última actualización que se haya lanzado gracias a sus actualizaciones inmediatas.
- Las aplicaciones web son multiplataforma.
- Se pueden usar desde cualquier sistema operativo gracias a que solo se necesita un navegador web para ejecutarlas y acceso a Internet.
- Pueden estar siempre disponibles eliminando las barreras geográficas.
- Ofrecen una gran colaboración entre las organizaciones debido su sencillo acceso y compartición de los datos.

1.3.1 Sistemas similares

Las aplicaciones web pueden ser utilizadas para la gestión de la información o del negocio de una empresa, ofreciendo soluciones para optimizar el rendimiento y la relación con los clientes. Permiten aumentar la cantidad y la calidad de los servicios ofertados a los clientes debido a que son ágiles, eficaces y accesibles desde cualquier parte del mundo. Brindan un acceso seguro y rápido a los datos desde cualquier ubicación, posibilitando realizar servicios online que sustituyan muchas de las tareas administrativas.

OpenGestión: Empresa española que se dedica a crear soluciones de *software* de gestión utilizando entornos web. Un ejemplo de su trabajo es MyGestión, que ofrece una plataforma de gestión empresarial que cuenta con cuatro módulos que permiten gestionar la administración, los clientes, la contabilidad y las tiendas *online* de las empresas. (OPEN GESTIÓN S.L) Para utilizar esta plataforma no se requiere de ninguna instalación, ya que

se puede trabajar utilizando Internet a través de un navegador. Uno de los módulos que brinda es:

Módulo de gestión de clientes: en este módulo se realiza un registro de los datos de todos los clientes, pudiendo agruparlos en categorías y almacenar información de contacto (dirección, correo electrónico, teléfono, etc.) y económica (descuentos, formas de pago, etc.); también se pueden agregar, editar, eliminar e incluso buscar datos de los clientes de forma rápida.

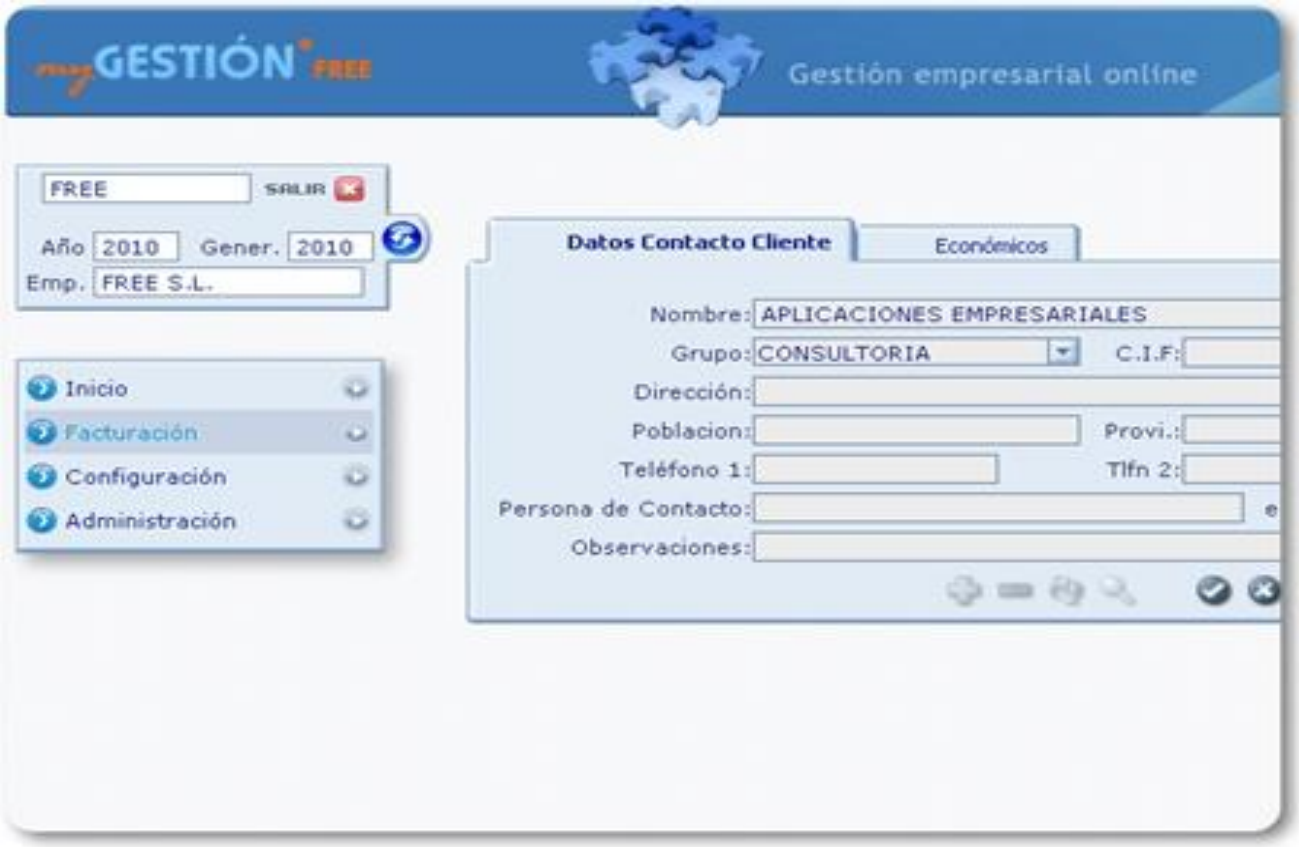


Figura 1.1: Módulo de gestión de cliente de MyGestión (Fuente: OPEN GESTIÓN S.L)

Módulo de presupuestos: este módulo permite generar reportes de presupuestos e imprimirlos para su posterior envío a los clientes.

Módulo de facturación: se relaciona con el módulo de gestión de clientes y permite elaborar reportes de factura. Las cuales pueden ser modificadas desde la función de administración del programa, permitiendo imprimirla o enviarla vía correo electrónico según sea necesario.

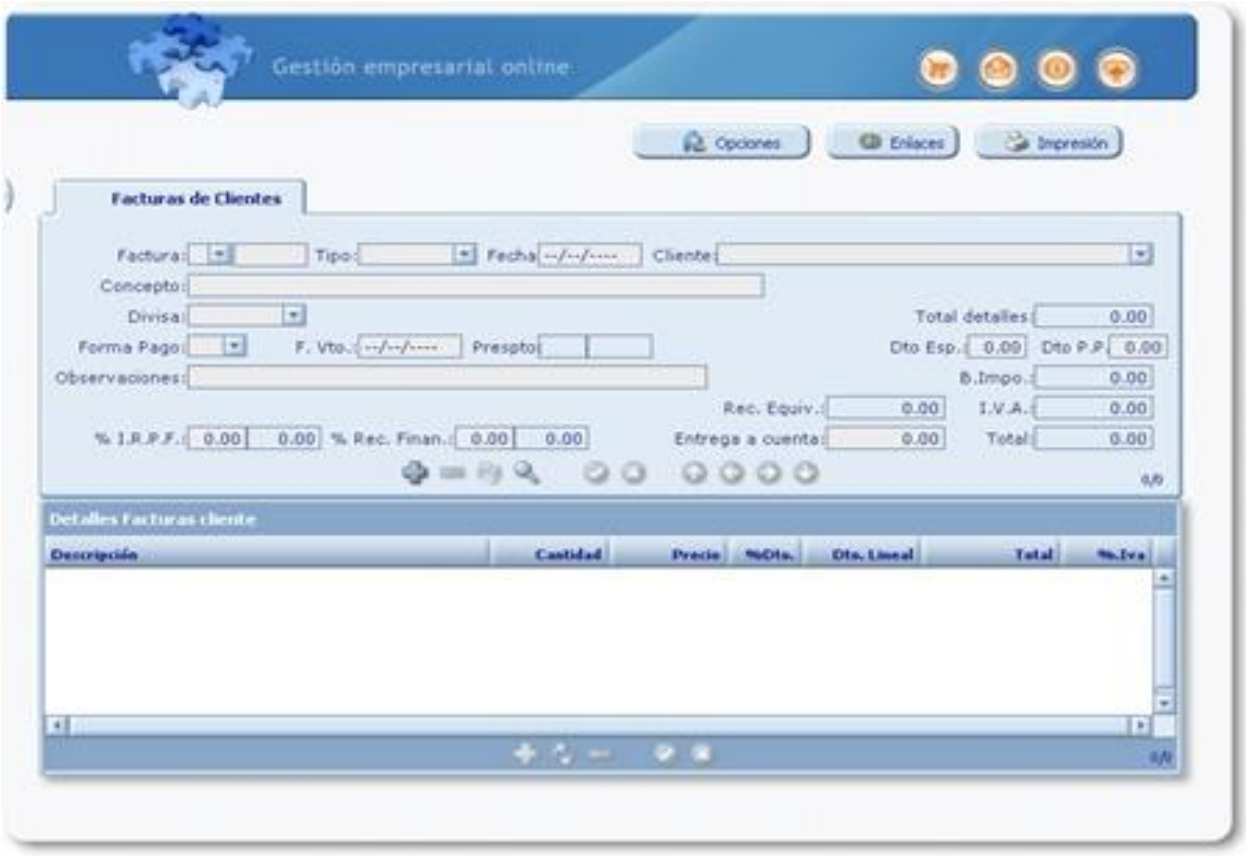


Figura 1.2: Módulo de facturación de MyGestión (Fuente: OPEN GESTIÓN S.L)

A pesar de sus múltiples ventajas para la gestión empresarial, la utilización de MyGestión, no es una vía factible para el desarrollo de los módulos clientes y reportes, ya que este *software* pertenece a una entidad propietaria. Para su utilización se tendría que hacer una inversión inicial de 20 mil euros.

Websuitepro: Sistema que permite la gestión de una empresa por la web, posibilitando el análisis y reportes de gastos y la gestión de los clientes. (Web Suite Pro) Este sistema elimina las barreras geográficas ya que permite la interacción con clientes y empleados desde cualquier lugar que tenga acceso a Internet. Gestiona las compras que se realizan y ejecuta el control de inventarios, permitiendo saber exactamente en qué condiciones se encuentra la empresa. Aunque este *software* ofrece varios beneficios, no se puede utilizar pues la versión gratuita es bastante limitada, y no está disponible en español.

SiGestCTC: Aplicación web que gestiona la información en el Sindicato Nacional de la Administración Pública en Cuba y permite tener acceso mediante la red a los servicios que brinda esta institución. El sistema posibilita que los directivos de las diferentes estructuras sindicales del país cuenten con la información actualizada de forma tal que puedan tomar decisiones eficaces ahorrando tiempo y coste. (Meriño Menadier, y otros) Para su desarrollo se utilizó un servidor central y herramientas de *software* libre como PHP, MySQL como lenguajes de programación, RUP como metodología de desarrollo y UML como lenguaje de modelado.

Este sistema posee un módulo de administración que gestiona todo lo relacionado con los permisos de usuarios y los cambios en la configuración del sistema asegurando que cada actor del sistema pueda acceder y gestionar la información a la que tiene permiso.

Figura 1.3: Módulo de administración de SiGestCTC (Fuente: Meriño Menadier, y otros)

Esta aplicación también genera los modelos A1, éstos son generados en cada una de las estructuras del sindicato, desde el municipio hasta la máxima instancia que es la Central de Trabajadores de Cuba Nacional (CTC). Este modelo es la vía para conocer toda la información de los trabajadores del país, donde se reflejan el total de centros de trabajo, secciones sindicales y otros datos referidos a la organización. Constituye el modelo fundamental de datos del movimiento obrero, y su elaboración data desde casi la propia creación de la CTC en Cuba, aunque no en la forma que se realiza actualmente.

Figura 1.4: Modelo A1 Nacional generado por el sistema (Fuente: Meriño Menadier, y otros)

Este sistema, a pesar de tener algunas similitudes con la aplicación que se quiere desarrollar, no se puede utilizar, pues no posee varias de las características con las cuales ésta debe cumplir, como por ejemplo, el módulo de administración solo se encarga de los permisos de los usuarios y no de la administración de los datos de la empresa.

1.4 Sistemas de Integración Bancaria

Sistema Integral de Recaudación del Servicio de Administración de Registros y Notarías

El Servicio de Administración de Registros y Notarías (SAREN) es un organismo de la República Bolivariana de Venezuela que gestiona todo lo relacionado con registrar y notarial documentos de ventas de bienes muebles e inmuebles, compañías anónimas, firmas personales y registro de títulos universitarios entre otros servicios. También da garantía de seguridad jurídica en el país. SAREN se divide en cuatro tipos de oficinas, los Registros Principales, los Registros Públicos, los Registros Mercantiles y las Notarías Públicas. (SAREN)

El proyecto de Automatización de los Registros y Notarías crea un Sistema Integral de Recaudación del SAREN para realizar la recaudación de pagos de este organismo. El

sistema integra los bancos recaudadores con SAREN, de forma tal que se pueda hacer un seguro cobro de los pagos. Este sistema permite que los clientes soliciten los servicios de SAREN a través de una aplicación web, también, mediante esta aplicación los administradores pueden gestionar los servicios que brindará SAREN así como los pagos, los bancos y las cuentas. El SIB utiliza el mismo procedimiento de comunicación que usa SAREN con sus bancos recaudadores, el envío de las cadenas de seguridad y las lecturas de pagos. Para la realización de los módulos no se puede utilizar este sistema ya que no usa el mismo proceso de gestión de la información.

Medios de pagos en línea

Para realizar los cobros y pagos en línea se requiere de una pasarela de pagos; ésta realiza una transferencia rápida y segura de información entre un portal de pago y los bancos asociados. Cuando un cliente solicita un servicio o un producto de un sitio web que tiene habilitado una pasarela de pago, se realizan una serie de tareas que procesa la transacción de manera transparente para el comprador.

Actualmente existen varios sitios como: Mercado Pago**,** DineroMail o PayPal, los cuales brindan pasarelas de pagos seguras que operan con tarjetas de crédito y transferencias bancarias, las cuales son viables a la hora de realizar la integración de un sistema web con bancos para realizar los pagos en estos. También se pueden implementar estas pasarelas utilizando un servidor seguro propio y realizando convenios con los bancos emisores de las tarjetas de crédito y otros sistemas de pago.

PayPal: empresa perteneciente al sector de comercio electrónico, propiedad de eBay. La empresa no se rige por las mismas leyes que las entidades bancarias, por lo que los usuarios no están protegidos frente a estas. PayPal cobra al vendedor una comisión por utilizarlo como plataforma de cobro y por retirar fondos a su cuenta corriente y al comprador por realizar la conversión de divisas al comprar en una moneda distinta. Esta plataforma no ofrece seguridad en cuanto a la recuperación de dinero de una transacción que se desee cancelar, ya que solamente esto ocurrirá si las partes involucradas están de acuerdo. Para realizar las transferencias seguras esta plataforma usa una tecnología de encriptación SSL de 128 bits para proteger toda la información confidencial. (eBay)

En la actualidad, en el sistema a desarrollar no se puede implantar esta solución porque los pagos se realizan de manera personal en el banco.

1.5 Estudio de las metodologías y estándares para el desarrollo del software

Existen dos tipos de metodologías para el desarrollo de *software*, las tradicionales y las ágiles. La metodología ágil está orientada para usarse en proyectos pequeños que tengan un reducido plazo de tiempo, que usen nuevas tecnologías y los requisitos sean cambiantes. La metodología tradicional se emplea cuando se quiere realizar un buen control del proceso mediante una rigurosa definición de roles, actividades y artefactos, también se realiza un detallado modelado y redacción de la documentación. Por las características antes expuestas, se puede decir que la metodología tradicional es mucho más factible cuando se requiere de una buena calidad del *software*, pues la generación de todos sus artefactos elimina considerablemente la aparición de errores y el mal levantamiento de los requisitos, permitiendo estar seguros de que la aplicación cumple con las necesidades del cliente y la satisfacción del mismo. Se propone, por política del proyecto al cual pertenece el sistema, utilizar el Proceso Unificado de Desarrollo (RUP) como una metodología tradicional para el desarrollo de la aplicación. Por todo lo mencionado anteriormente, sería una mejor solución utilizar una metodología ágil como XP pues el tamaño del grupo de desarrollo, en este caso, es de una persona y se hace necesario tener resultados tangibles a corto plazo. En la Figura 1.5 se ilustra las diferencias existentes entre estos dos tipos de metodologías.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Figura 1.5: Diferencias entre metodologías ágiles y no ágiles (Fuente: Letelier, y otros)

1.5.1 Proceso Unificado de Desarrollo (RUP)

Es una metodología que proporciona un conjunto de técnicas que soportan el ciclo completo del desarrollo de *software*, permitiendo realizar el análisis, la documentación y la implementación de sistemas orientados a objetos. Se adapta al contexto de cada organización, siendo así una metodología que se ajusta a las necesidades del cliente, ya que es muy importante interactuar con él y a las características de la organización. Tiene como objetivo asegurar que el *software* que se desarrolle posea una alta calidad y que cumpla con las especificaciones del cliente dentro del calendario y presupuesto establecido. RUP posee tres características fundamentales: estar centrado en la arquitectura, ser iterativo e incremental y dirigido por casos de uso, posee cuatro fases: inicio, elaboración, construcción y transición.

Centrado en la arquitectura: la arquitectura es la vista completa del sistema donde resaltan los elementos más importantes del modelo, diseñándola de tal forma que sea clara, flexible para futuros cambios y reutilizable, estando de acuerdo el equipo de proyecto y los clientes con su diseño.

Iterativo e incremental: cada fase se desarrolla en iteraciones, dividiendo el proyecto en mini-proyectos que terminen en un incremento, esto permite un mejor desarrollo así como una mejor comprensión. La iteración se refiere a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto.

Dirigido por casos de uso: los casos de uso responden las necesidades de los clientes finales y se definen a partir de los requisitos funcionales, los casos de uso dirigen el diseño implementación y prueba de estos, guiando así el proceso de desarrollo del *software*.

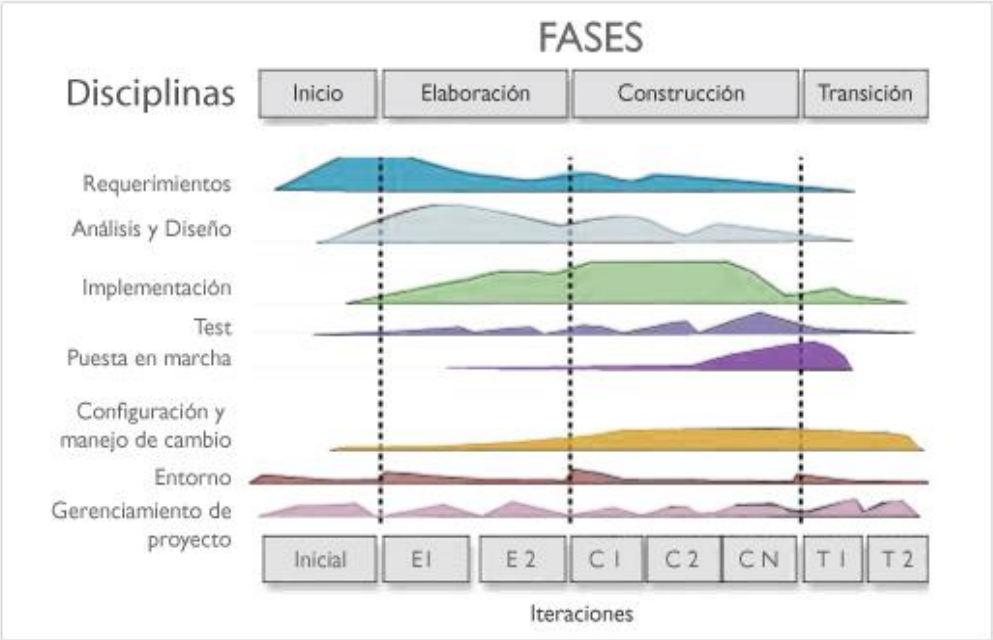


Figura 1.6: RUP en dos dimensiones (Fuente: Guerra Roberto, 2010)

Flujos

Modelado del negocio: se asegura que los usuarios finales y desarrolladores tengan un entendimiento común de la organización. Se describen los procesos del negocio, se identifican quiénes participan y las actividades que se quiere automatizar asegurándonos que el producto sea útil.

Requisitos: en este flujo se describen los requisitos definiendo las funcionalidades y restricciones. Se establece qué es lo que tiene que hacer el sistema exactamente.

Análisis y diseño: en el análisis se define qué es lo que va a hacer el sistema y en el diseño se obtiene un refinamiento del análisis indicando cómo debe quedar implementado el sistema.

Implementación: se realiza la implementación de la arquitectura y del sistema como un todo quedando como resultado un sistema ejecutable.

Prueba: en este flujo se evalúa la calidad del producto desarrollado a todo lo largo del ciclo de vida proporcionando una retroalimentación.

Despliegue: en este flujo se asegura la aceptación del producto por parte de los usuarios finales, distribuyendo e instalando el *software* y brindando asesoramiento.

Flujos de trabajo de apoyo

Gestión del proyecto: se realizan actividades para dar seguimiento al *software* y que este cumpla con las necesidades de los usuarios.

Gestión de configuración y cambios: se asegura la integridad y la correcta documentación de todos los cambios que se generan en los artefactos.

Ambiente: en este flujo de trabajo se realizan actividades para definir las herramientas y procesos que se necesitarán para darle soporte al *software.* (Guerra Roberto, 2010)

1.5.2 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta.

UML es un lenguaje para modelar sistemas de *software* donde se visualizan y conceptúan los procesos, funciones y esquemas de base de datos de los mismos. Permite documentar y construir los artefactos, describiendo el modelo del sistema. Visualiza un sistema gráficamente de forma tal que otro pueda entenderlo y define cuáles son las características de un sistema antes de su construcción. A partir de los modelos construidos se puede desarrollar el sistema y a su vez sirven de documentación.

El Lenguaje Unificado de Modelado es en la actualidad un esquema de representación gráfica ampliamente utilizado para modelar sistemas orientados a objetos. Unifica los distintos esquemas de notación que existían a finales de la década de los 80. Aquellos que diseñan sistemas utilizan el lenguaje (en la forma de diagramas) para modelar sus sistemas. Una de las características más atractivas de UML es su flexibilidad. Los modeladores en UML pueden desarrollar sistemas mediante el uso de distintos procesos, pero todos los desarrolladores pueden expresar dichos sistemas con un conjunto estándar de notaciones. (Deitel, y otros, 2003)

1.6 Selección de las herramientas y lenguajes de desarrollo

A continuación se describen las herramientas y lenguajes de programación utilizadas.

1.6.1 Lenguajes de programación empleados

XHTML

XHTML (Lenguaje de Marcado de Hipertexto Extensible) es un lenguaje de marcado que surge a partir de HTML con el objetivo de sustituirlo, eliminando así los problemas que existían producto a la limitación del uso de las herramientas XML, para esto combina la sintaxis de HTML, diseñado para mostrar datos y la sintaxis de XML, diseñada para describir datos. (Lancker, 2007)

Los documentos XHTML están diseñados para trabajar en conjunto con aplicaciones de usuarios basadas en XML, también son validados, visualizados y editados con las herramientas propias de este lenguaje. Gracias a su estricto etiquetado, permite una correcta interpretación de la información desde cualquier dispositivo que se desee acceder. XHTML posee un correcto etiquetado, pues el documento donde se encuentran los elementos correctamente anidados está estructurado coherentemente, donde se incluye etiquetas en minúsculas y valores de los atributos entrecomillados. La versión más estable de este lenguaje es XHTML 1.0, donde se pueden escribir documentos que funcionen tanto en las aplicaciones de usuarios conforme a HTML 4.0 como en las nuevas aplicaciones XHTML 1.0. Este lenguaje permite una separación entre el contenido y el diseño de manera que si se quiere modificar uno no afectaría al otro, admitiendo una independencia entre ellos.

CSS

CSS (Hojas de Estilos en Cascadas) es un lenguaje que permite separar la estructura de un documento de su presentación, ya que estas hojas de estilos se usan para definir cómo quedará la presentación de un documento XHTML. (Lancker, 2007)

Las etiquetas precisan cómo quedará estructurado el documento HTML, la información del estilo con las que serán visualizadas, se plasmará en las hojas de estilos en cascadas, como color, fuente, alineación del texto, tamaño y otras características. El uso de este lenguaje permite que los usuarios puedan crear su propia hoja de estilo en dependencia de sus preferencias aumentando la accesibilidad del sitio web. Una de las ventajas que proporciona este lenguaje de programación es que una página web dispone de varias hojas de estilo y se muestra según el dispositivo que se use o la elección del usuario.

PHP

PHP es un lenguaje libre e interpretado del lado del servidor, que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos en PHP son embebidos directamente en el código HTML y ejecutado por el servidor web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de código HTML puro. (Cobo, y otros, 2005)

Se utilizará PHP como lenguaje de programación ya que es gratuito e independiente de plataforma, por lo que está incluido en la soberanía tecnológica de VNZ con la ley de software libre como parte del uso prioritario de este tipo de software en este país. En la actualidad PHP puede ser usado en una interfaz de línea de comando y en la creación de aplicaciones con interfaces gráficas. Este lenguaje fue creado inicialmente para la elaboración de páginas web dinámicas, resultando ser muy útil a la hora de diseñarlas de forma eficiente. Permite a los programadores crear aplicaciones rápidas ya que es un potente lenguaje de secuencia de comando, y su similitud con lenguajes de programación estructurada como Perl o C le permite crear aplicaciones complejas con una curva de aprendizaje pequeña. PHP puede interactuar con la mayoría de los servidores web como Apache y sistemas operativos, sin costo alguno; también permite la conexión con varios motores de bases de datos como MySQL, Postgres SQL y Oracle.

Existen varios *frameworks* que utilizan PHP como Zend Framework, Kohana y Symfony, que es el propuesto para desarrollar la aplicación, también existen varios entornos de desarrollo para PHP como Dreamweaver, Zend Studio, PHPEclipse y NetBeans, que utiliza Symfony y también se propone para realizar el desarrollo del sistema.

1.6.2 *Framework*

Los *frameworks* o marcos de trabajo se basan en la reutilización de componente de *software* integrándolos para el desarrollo ágil de aplicaciones. Facilitan el trabajo de los desarrolladores ya que poseen varios componentes como máquinas virtuales, compiladores, bibliotecas de administración de recursos y especificaciones del lenguaje liberándolos de escribir código de bajo nivel. En resumen, un *framework* se define como componentes reutilizables para realizar el desarrollo de sistemas de una forma más rápida y cómoda. Existen varios tipos de *frameworks* web orientados a la interfaz de usuario,

como Java Server Faces, orientados a aplicaciones de publicación de documentos, como Coocon, orientados a la parte de control de eventos, como Struts.

Se propone el *framework* Symfony para el desarrollo del sistema, el cual simplifica el desarrollo de aplicaciones web. Incluye el patrón de diseño Modelo Vista Controlador (MVC) y está desarrollado en PHP5. Tiene como ventajas que es compatible con varios gestores de base de datos (MySQL, PostgreSQL, Oracle y Microsoft SQL Server) y es compatible con las plataformas Linux y Windows. Este *framework* ha sido probado en proyectos reales y es muy utilizado por empresas como es CETA-CIEMAT que se dedica principalmente a la investigación, desarrollo y servicio en tecnologías de la información y de las comunicaciones. (Francois Zaninotto, 2010)

Dentro de las principales características que posee este *framework* y que hace su uso más viable están: la facilidad de su instalación y configuración, utiliza la programación orientada a objetos y sigue la mayoría de patrones para el diseño de la web. Es adaptable a la arquitectura y políticas de cada empresa, facilita la generación de código gracias a una potente línea de comando lo cual ayuda a ahorrar tiempo de desarrollo. Facilita el trabajo con los formularios pues estos soportan validaciones automáticas asegurando una mejor calidad en los datos de la base de datos; también separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web facilitando el trabajo si se decide crear una nueva la aplicación web. Otra de las ventajas que ofrece es que automatiza las tareas comunes permitiendo que el desarrollador pueda dedicarse a las funcionalidades más específicas de la aplicación.

1.6.3 Mapeo de objeto-relacional

En la actualidad, casi todas las aplicaciones están diseñadas para usar la Programación Orientada a Objetos (POO), y las bases de datos más utilizadas son del tipo relacional, es decir, que solo permiten guardar tipos de datos primitivos. Este problema trae consigo que no se pueda guardar directamente en las tablas los objetos que genera la aplicación, por lo que se hace necesario realizar una conversión de los objetos en registro y para recuperar los datos se realiza el proceso inverso, todo esto trae consigo algunos inconvenientes. Para evitar esta serie de problemas, se crea el mapeo de objeto-relacional (ORM), que es el encargado de realizar estas conversiones y simula que la base de datos es orientada a objeto.

Un ORM (mapeo de objeto-relacional) es una técnica de programación que ayuda a realizar la conversión de tipos entre sistemas que usan la programación orientada a objetos y las bases de datos relacionales, posibilitando el uso de características de la POO como la herencia y el polimorfismo, logrando una rapidez en el desarrollo. (Wage, y otros) Algunos ejemplos de ORM que existen en la actualidad son Doctrine e Hibernate.

Mientras que Hibernate es un ORM para el lenguaje de programación Java, Doctrine es una librería para PHP, por lo que se propone utilizar este último. Doctrine permite realizar ingeniería inversa y crear el modelo a partir de la base de datos, generando de forma automática el modelo de clases basándose en el modelo relacional de tablas. También permite crear la base de datos a partir del modelo que se realiza utilizando la sintaxis propia de él, para esto se utiliza PHP que es un lenguaje de marcado ligero. Doctrine posee buscadores que pueden encontrar registros basándose en cualquier campo de una tabla. La principal ventaja de este ORM es su propio lenguaje DQL (Lenguaje de Consultas de Doctrine) surgido de HQL (Lenguaje de Consulta de Hibernate) de Hibernate ya que Doctrine surgió a partir de ese ORM, que es muy potente y brinda una abstracción de la base de datos muy completa. En ocasiones este lenguaje puede ser complejo para realizar consultas muy avanzadas, en este caso Doctrine da la posibilidad de usar SQL nativo.

DQL es un lenguaje que permite obtener los objetos de la base de datos, siendo esta una de sus principales ventajas. No es necesario escribir los *joins* a mano, pues DQL entiende las relaciones, haciendo las consultas menos complejas, también es portable con diferentes bases de datos. Es más factible el uso de DQL cuando se quiere obtener la información a cargar, que el uso de la forma automática de Doctrine, ya que esto puede causar que se ejecuten consultas SQL que no son necesarias, siendo esto una pérdida de rendimiento, con DQL se puede cargar toda la información de una sola vez.

1.6.4 Sistema gestor de bases de datos

Un SGBD no es más que un grupo de herramientas que permite la gestión de los archivos de las bases de datos. Existen varios sistemas de gestión de bases de datos como son: Apache Derby, MySQL, PostgreSQL y Oracle.

La implementación de una base de datos integrada es posible mediante el sistema gestor de bases de datos: un complejo conjunto de programas diseñado para facilitar el establecimiento y utilización de una base de datos. (Saffady, 1986)

El SGBD será PostgreSQL, es un sistema libre y orientado a objetos que brinda la opción de instalar una base de datos no privativa, permitiendo ahorrar dinero, mejorar el rendimiento e incrementar la productividad. Ofrece una alta congruencia, pues posee una herramienta llamada Control de Concurrencia Multi-Versión (MVCC) que permite que se puedan realizar diferentes procesos en la misma tabla (escritura y acceso) sin provocar bloqueos innecesarios, siendo capaz de manejar los registros sin necesidad de que los usuarios tengan que esperar a que estos estén disponibles.

1.6.5 Entorno de desarrollo integrado seleccionado

Un entorno de desarrollo integrado (IDE) es un paquete de *software* diseñado para la creación y ejecución de un programa. Son varias las compañas de *software* importantes las que cuentan con estos entornos de desarrollo, los cuales combinan las funciones de editor, compilado, enlazador y visor de *applets*. Un entorno de desarrollo integrado ofrece menús y botones para facilitar tanto como sea posible la creación de un programa. (Bell, 2003)

Por sus características, se utilizará el NetBeans 6.9 como el entorno de desarrollo para el sistema que se desarrollará, esta versión es muy recomendable para desarrollar aplicaciones web complejas con PHP 5.3 y da soporte a Symfony. NetBeans es un proyecto de código abierto que permite que las aplicaciones sean desarrolladas a partir de módulos, los cuales contienen las clases Java ya que está escrito en este lenguaje de programación. Este IDE posee una plataforma de aplicación que permite la administración de las interfaces de usuario, de las configuraciones del usuario, del almacenamiento, de ventanas y la utilización de *framework* basado en asistentes que permiten a los desarrolladores crear con rapidez aplicaciones web, de escritorio, empresariales y móviles.

1.6.6 Herramienta CASE para el modelado UML

Las herramientas CASE (Ingeniería de *Software* Asistida por Computadoras) son un conjunto de programas que ayudan a aumentar la productividad en el desarrollo de *software* reduciendo coste y tiempo. Se pueden definir como diversas aplicaciones

informáticas que dan apoyo a los analistas, desarrolladores e ingenieros de *software* durante la realización de tareas del ciclo de vida del desarrollo del mismo, como: la realización del diseño del sistema y la implementación automática de partes de este, cálculo de costes, compilación automática, documentación y detección de errores. (Alonso Amo, y otros, 2005)

Para el sistema que se va desarrollar, se utilizará el Visual Paradigm como herramienta CASE, ya que es una herramienta UML con la cual se puede modelar todo el ciclo de vida del *software*. Permite realizar todos los tipos de diagramas de clases y generar códigos desde estos, también permite generar la documentación. Con Visual Paradigm se puede realizar la ingeniería inversa, llevando de código a modelo y de código a diagrama, así como la generación de código a partir de modelos y diagramas. Con el uso de Visual Paradigm se puede generar la base de datos ya que realiza la transformación de diagramas de Entidad Relación en tablas de la base de datos. Soporta el mapeo de objeto relacional (ORM) y está ampliamente integrada con NetBeans.

1.6.7 Servidor web

Un servidor web es un ordenador en el que se ejecuta un programa servidor HTTP (*Hyper-Text Transfer Protocol*), por lo que puede denominarse servidor HTTP. En un ordenador que ejerce la función de servidor se puede instalar más de un tipo de *software* servidor. En caso de un servidor web, es muy frecuente instalar tanto *software* servidor HTTP como *software* servidor FTP (*File Transfer Protocol*). (Brochard, 2006)

El servidor Apache es un servidor web HTTP, multiplataforma, de tecnología *Open Source* sólido y de código abierto. La arquitectura de Apache es modular, y estos módulos aportan funcionalidades que pueden ser básicas para un servidor web. Podemos usar Apache cuando deseamos poner a disposición contenido de una forma segura y confiable, como por ejemplo compartir información desde una computadora personal hacia internet. Este servidor le brinda comodidad a los desarrolladores, ya que pueden utilizar una versión local de Apache y probar el código mientras se esté desarrollando.

El servidor web Apache es el servidor web más implantado entre los distintos servidores que ofertan servicios web en Internet. Entre las características más significativas destacamos: es modular, capacidad para crear servidores virtuales, capacidad para crear servidores seguros HTTPS, capacidad para crear sitios privados. (Caballero, 2003)

1.7 Descripción de la arquitectura

La aplicación se realizara mediante la implementación del patrón de arquitectura Modelo Vista Controlador (MVC) el cual separa la lógica de control, la vista y los datos en tres componentes. Este patrón es comúnmente usado en aplicaciones web donde la vista es la interfaz de usuario mediante el cual éste interactúa con el sistema, el modelo es la lógica del negocio y el sistema que gestiona la base de datos y el controlador es el que invoca peticiones al modelo para responder a eventos que son usualmente acciones del usuario sobre el sistema. Symfony realiza una implementación de este patrón con algunas variantes pues subdivide éstas tres capas en otras capas.

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos; ésta última es completamente transparente para el programador. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas.

La capa de la vista se separa en un *layout*[1] y en una plantilla, pues las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación cambiando solo el interior de la página. Normalmente, el *layout* es global en toda la aplicación o al menos en un grupo de páginas. La plantilla sólo se encarga de visualizar las variables definidas en el controlador.

La capa controlador normalmente se divide en un controlador frontal, que es único para cada aplicación y ofrece un punto de entrada único para toda la aplicación, facilitando el trabajo a la hora de hacer modificaciones, pues no se tendría que modificar todas las controladoras. Otra capa son las acciones, que incluyen el código específico del controlador de cada página. Quedando el patrón MVC como se muestra en la Figura 1.7

La capa del Modelo

- Abstracción de la base de datos
- Acceso a los datos

La capa de la Vista

[1] *Layout*: plantilla web que posee los elementos que se muestran de forma idéntica en una aplicación.

- Vista
- Plantilla
- Layout

La capa del Controlador

- Controlador frontal
- Acción

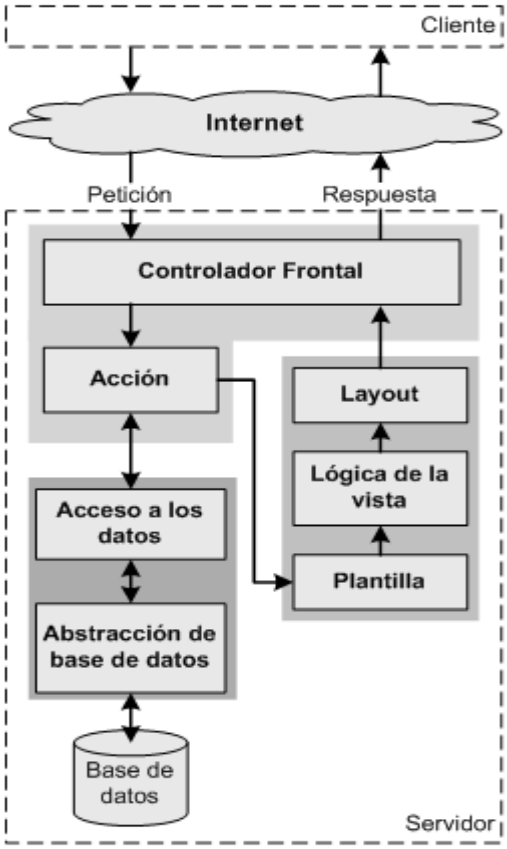


Figura 1.7: Patrón MVC en Symfony (Fuente: Potencier)

1.8 Patrones de diseño

Los patrones de diseño son arquitecturas comprobadas para construir *software* orientado a objetos que sea flexible y pueda mantenerse. Al utilizar patrones de diseño se puede reducir considerablemente la complejidad del proceso del diseño. Los patrones de diseño benefician a los desarrolladores del sistema:

- Ayudar a crear *software* confiable, utilizando arquitecturas comprobadas y la experiencia acumulada en la industria.

- Promover la utilización del diseño en experiencias posteriores.
- Ayudar a identificar los errores y obstáculos comunes que ocurren al crear el sistema.
- Ayudar a diseñar sistemas independientemente del lenguaje que se va a utilizar.
- Acortar la fase de diseño en un proceso de desarrollo de *software.* (Deite, y otros, 2004)

Symfony se ha construido teniendo en cuenta muchos de los patrones de diseño comunes como los Patrones de *Software* para la Asignación General de Responsabilidad (GRASP), pues es un patrón para diseño orientado a objeto que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

GRASP posee 5 patrones principales que son:

Experto:

Asigna toda la responsabilidad a la clase que posee toda la información necesaria para crear un objeto o implementar un método. Symfony incluye varias clases en las cuales se evidencian el uso de este patrón:

- *sfController*: Clase perteneciente al controlador frontal que se encarga de decodificar la petición y transferirla a la acción correspondiente.
- *sfRequest*: Almacena los elementos que forman la petición.
- *sfResponse*: Contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.

Creador:

Asigna quién debe ser el responsable de la creación o instanciación de nuevos objetos. Las acciones definidas para cada módulo se encuentran en la clase *nombremoduloActions* aquí se crean los objetos de las clases que representan las entidades y de los objetos de otras clases que intervienen en la lógica del módulo.

Controlador:

Ayuda a que la lógica del negocio esté separada de la capa de presentación; es el que recibe los datos de los usuarios y los envía a distintas clases en dependencia del método que fue llamado. La clase "*sfActions"* maneja todas las peticiones *web*, ésta es el único punto de entrada de toda la aplicación en un determinado entorno. Cuando recibe una

petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Bajo acoplamiento:

Ayuda a aumentar la reutilización y disminuye la dependencia entre las clases. En Symfony cada clase se le asigna una responsabilidad de forma tal que exista poca dependencia entre las mismas.

Alta cohesión:

Ayuda a que la información que almacenan las clases sea coherente y esté, siempre que se pueda, relacionada con la clase. En cada clase "*Actions"* se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un *software* flexible ante los cambios. Symfony agrupa las clases por funcionalidades que son fácilmente reutilizables, bien por su uso directo o por herencia.

Patrones de diseño *Gang-of-Four* (GOF) como son:

Creacionales:

- *Singleton* (Instancia única): en el controlador frontal de Symfony hay una llamada a *sfContext: getInstance (),* el método *getContext(),* es un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony garantizando la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

- *Abstract Factory* (Fábrica abstracta): cuando el *framework* necesita crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea permitiendo trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí.

Estructurales:

- *Decorator* (Envoltorio): este patrón añade funcionalidad a una clase dinámicamente. En Symfony existen plantillas que heredan de otras plantillas combinando la herencia para rehusar código entre ellas. El archivo "*layout.php"*, que es plantilla global, es común a todas las páginas de la aplicación, para no tener que repetirlo.

- *Composite* (Objeto compuesto): Symfony también utiliza este patrón en su código fuente, como por ejemplo en la clase "*composite.class.php"* de "*sfLog"* permitiendo tratar de la misma forma objetos simples y objetos compuestos.

Comportamiento:

- *Command* (Acción): encapsula una operación en un objeto, ejecutando dicha operación sin necesidad de conocer el contenido de la misma, permitiendo que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación.

También se utilizará el patrón Agente Remoto y Agente, los cuales pertenecen a los patrones GOF (Pandilla de los Cuatro), ya que el sistema debe comunicarse con un servicio externo (servidor).

El patrón Agente Remoto sugiere crear una clase de *software* local que represente el servicio externo y asignarle la responsabilidad de contactar el servicio real. El Agente Remoto es un caso especial del patrón general Agente (Proxy) de la Pandilla de los Cuatro y sugiere utilizar un sustituto del componente (objeto, servidor, controlador de dispositivo, DLL, etc.) en algunos contextos. (Larman)

1.9 Conclusiones parciales

Con la realización de un análisis profundo del proceso de recaudación de pagos de SAIME, se logró una mejor comprensión del funcionamiento de las actividades relacionadas con la administración y los reportes. Esto permitió mejorar considerablemente el flujo de las mismas.

Para desarrollar los módulos administración, clientes y reportes no se pueden utilizar ninguno de los sistemas que se investigaron, algunos pertenecen a entidades propietarias necesitando de una inversión inicial para poder utilizarlos. La versión gratuita de WebSuitePro que existe son limitadas y SiGestCTC no puede realizar las actividades específicas de los módulos.

Se recomienda la utilización de pasarelas de pago en el sistema que se quiere desarrollar ya que podría ser un medio viable y seguro para realizar los pagos, y en un futuro hacer que el sistema tenga un mejor funcionamiento.

La propuesta de utilizar el *framework* Symfony, PostgreSQL, Doctrine y NetBeans como herramientas para crear los módulos, agilizará el proceso de desarrollo. Éstas simplifican la creación de aplicaciones web y el acceso a los datos, permitiendo al mismo tiempo obtener una aplicación robusta y la utilización de varios lenguajes de programación.

Capítulo 2
Propuesta de solución

2.1 Introducción

Para darle solución al problema que se presenta es necesario realizar un análisis del comportamiento del negocio y así saber de qué forma se pueden efectuar las distintas acciones que el cliente desea para la optimización de sus procesos. Para ello se debe tener una amplia comprensión de todos los elementos relacionados con las actividades que conformarán dichos procesos e identificar los requisitos funcionales y no funcionales, pues ofrece una visión exacta de lo que se quiere realizar, así como la especificación de los mismos, esta actividad es muy importante y se debe realizar correctamente. La especificación de los requisitos y el diseño de la aplicación transforman los requerimientos en un diseño de lo que será el sistema, y brindará una propuesta de solución que le dará respuesta a todas las necesidades del cliente.

2.2 Procesos a informatizar

Se deberán informatizar los procesos de la Dirección de Gestión Administrativa de SAIME asociados al chequeo del estado de cuentas y los reportes de recaudación periódica de las oficinas.

Se administrará la información relacionada con los bancos recaudadores, cuentas bancarias, servicios que brindan, tipos de pago, proveedores, clientes, notificaciones y avisos promocionales. La informatización de estos procesos permitirá la creación, modificación y eliminación de esta información, así como la deshabilitación de clientes y servicios.

Se generarán reportes personalizados y se dará la opción de emitir planillas acerca de la cantidad diaria recaudada por servicios y solicitudes, así como los servicios vendidos por su tipo o estado, dando la posibilidad de imprimirlos. El resumen diario es el proceso que ocurre después del cuadre que se realiza al final del día por parte del banco.

Los clientes tendrán la posibilidad de registrarse en el sistema, cambiar su contraseña y modificar sus datos. Toda la información que se maneje en el sistema tiene un carácter confidencial, garantizando la integridad de la información de los ciudadanos.

2.3 Modelo de dominio

Un modelo de dominio es un diagrama de clases que contiene los conceptos u objetos del contexto del sistema, es decir, del mundo real y no de los componentes del *software*, estos representan los eventos que suceden en el mismo. El modelo de dominio se utiliza para lograr un mejor entendimiento del negocio al cual el sistema va a servir, puede ser tomado como el punto de partida para su diseño. En la Figura 2.8 se representa el modelo de dominio.

2.3.1 Identificación de los conceptos del dominio

- Operador de administración: es la persona que provee al proyecto de los datos y verifica que los objetivos se cumplan de la mejor forma posible.
- Reportes: es la información agrupada en tablas acerca de la operaciones que se realizan en las cuentas recaudadoras.
- Cuentas recaudadoras: se le denomina así a las cuentas de los bancos que recaudan dinero para SAIME.
- Notificaciones de correo: son mensajes que se les envían al cliente según sus preferencias mediante el correo electrónico.
- Tipos de pago: se refiere a las vías de pago que se les brindan a los clientes.
- Bancos recaudadores: se le denomina así a los bancos que recaudan dinero para SAIME.
- Servicio: se refiere a los servicios que brinda SAIME.
- Cliente: es la persona que accede a la aplicación y obtiene un servicio de SAIME.
- Datos personales: se refiere a los datos de las personas que obtienen un servicio que puedan ser útiles para SAIME.
- Sistema: es un *software* que gestiona todas las actividades que tengan relación con los clientes y con la administración de la información de la recaudación de pagos de SAIME.
- Avisos: se refiere a los avisos que recibirán los clientes si ocurre alguna eventualidad.

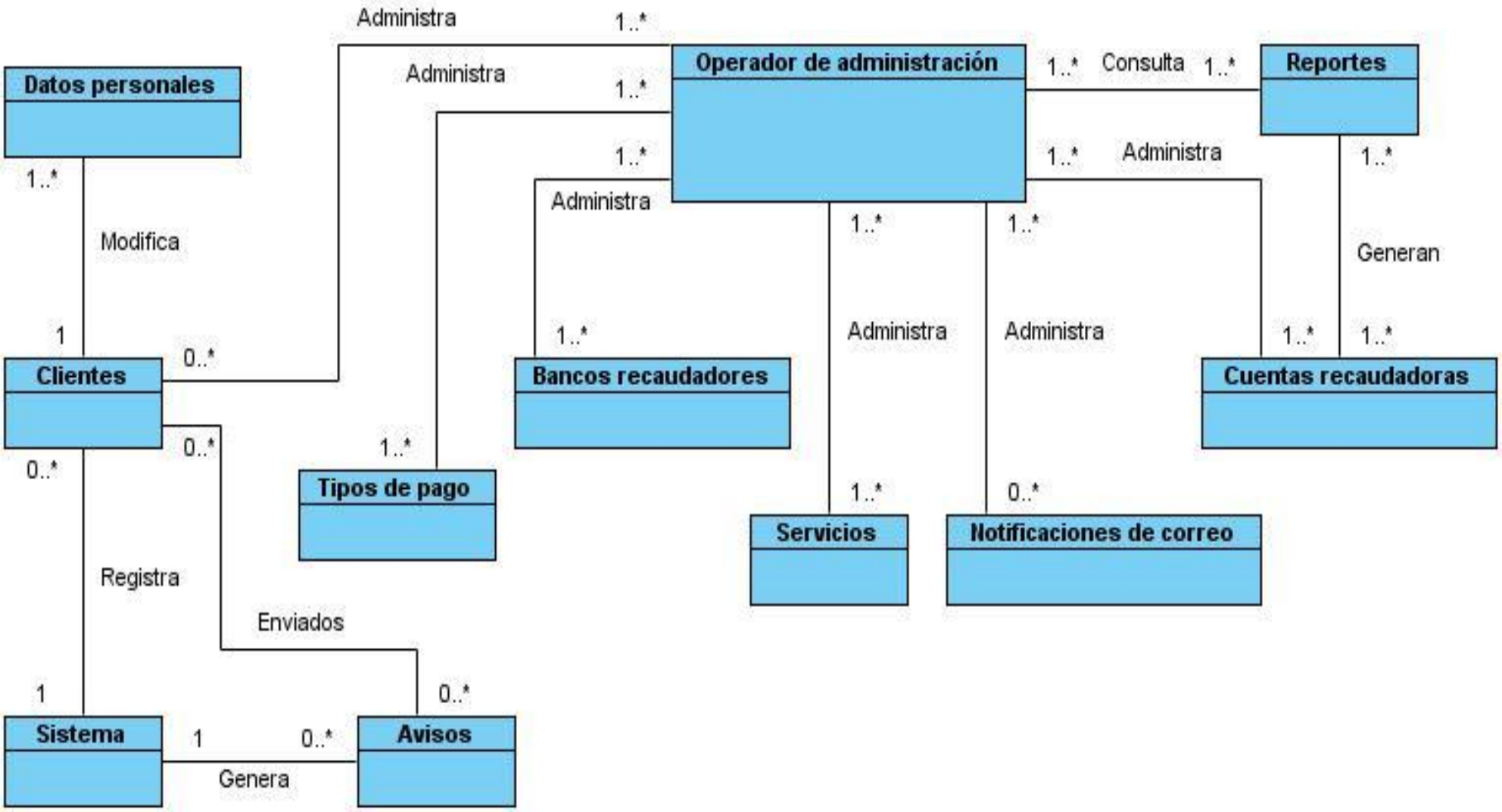


Figura 2.8: Modelo de dominio (Fuente: elaboración propia)

2.4 Solución propuesta

Para lograr darle solución a la problemática planteada, se propone realizar la implementación de un sistema informático que gestione la recaudación de pagos de SAIME, compuesto por tres módulos que se encarguen de la administración, los reportes y la interacción con los clientes. Para el desarrollo de estos tres módulos se utilizará la tecnología de Symfony con Doctrine como ORM y el lenguaje PHP, permitiendo la elaboración de un sistema web que brinde la posibilidad de realizar las actividades que se encuentran incluidas en estos módulos.

Módulo administración: este módulo se encargará de permitir la administración de los servicios que oferta SAIME, así como de los bancos y cuentas recaudadoras, clientes, proveedores y el envío de notificaciones mediante mensajes de correo electrónico. También permitirá la creación y envío de avisos promocionales para los clientes.

Módulo reportes: este módulo se encargará de realizar estadísticas y reportes de la cantidad diaria recaudada por servicios y solicitudes, así como los servicios vendidos por su tipo o estado, dando la posibilidad de imprimirlos.

Módulo clientes: este módulo se encargará de la comunicación de los clientes con SAIME, permitiendo que realicen actividades en el sistema como registrarse, cambiar la contraseña, modificar sus datos personales y marcar sus preferencias.

2.5 Especificación de los requisitos del sistema

La definición de los requisitos de los sistemas es una tarea sumamente importante que tiene una repercusión directa con el sistema finalmente implementado. Si esta definición no se realiza adecuada y minuciosamente, y si no se le dedican enormes esfuerzos desde el primer momento el sistema final denotará dolencias funcionales y de usabilidad y lo que suele ser peor, repercutirá en el aspecto negativo en el coste económico del sistema. (Granollers, y otros, 2005)

2.5.1 Requisitos funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de *software* que se desarrolle, de los posibles

usuarios del *software* y del enfoque general tomado por la organización al redactar requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo, los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etcétera. (Sommerville, 2005)

Los requisitos funcionales definidos para darle solución a la problemática se pueden ver en el Anexo 1.

2.5.2 Requisitos no funcionales

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como: la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa define las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. (Sommerville, 2005)

Requisitos de software:

- Sistemas operativos multiplataforma.
- Navegador web: Internet Explorer, Mozilla.
- Servidor web HTTP Apache 2.2.14.
- Servidor gestor de base de datos PostgreSQL 8.4.

Requisitos de hardware:

- Los servidores FTP, Web y de Base de Datos deben poseer 4 GB (*gigabyte*) de memoria RAM (*Random Access Memory*) como mínimo.

Estación de trabajo (PC cliente)

- Periféricos: mouse y teclado.
- Tarjeta de Red.
- 256 MB de Memoria RAM (Mínimo)
- Procesador Intel Celeron.

Impresora

- Conexión USB[2].
- Controladores multiplataforma.

Requisitos de apariencia o interfaz externa:

- La interfaz contará con teclas de función y menús que faciliten y aceleren su utilización.
- El diseño se realizará de manera similar a la página de Solicitud de Pasaporte de SAIME y responderá a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- El sistema permitirá al usuario distinguir visualmente entre los elementos en las ventanas, a través del uso de colores, tamaño de las fuentes, íconos, así como otras técnicas.
- La corrección de errores en la introducción de datos será clara y fácil de realizar. La entrada de datos incorrecta será detectada y notificada al usuario.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.
- La navegación será intuitiva, se utilizará un lenguaje simple, cercano al usuario y tratando de que sea sencillo y ameno, facilitando así la navegación por las opciones.

Restricciones en el diseño y la implementación:

- Las tecnologías de desarrollo de la aplicación serán: el *framework* Symfony v1.4.8, NetBeans v6.9, Visual Paradigm v6.4 y como mapeo de objeto relacional Doctrine.
- El lenguaje de programación será PHP 5.3.
- Se utilizará la Programación Orientada a Objetos.
- El protocolo de comunicación será HTTP.
- El protocolo de comunicación entre servidores será SFTP (*Simple File Transfer Protocol*).

Requisitos de usabilidad:

- La aplicación deberá poseer una interfaz y navegación asequibles y funcionales tanto para usuarios expertos como para los que no tienen conocimientos profundos de informática.

[2] USB: *Universal Serial Bus,* en español Conector Universal en Serie, es un puerto que sirve para conectar periféricos a un ordenador.

Requisitos de seguridad:

- Se realizarán validaciones que garanticen el sentido de los datos con respecto al negocio, se prohibirá la entrada de caracteres especiales.
- Disponibilidad: la información generada en el sistema por cada usuario, estará centralizada y en servidores seguros, donde el sistema podrá generar diferentes reportes en dependencia de los usuarios del sistema y sus respectivos roles o niveles de acceso.
- Integridad: garantiza el tratamiento adecuado de las excepciones y validación de las entradas del usuario para evitar entradas inadecuadas.

2.6 Modelo del sistema

Los diagramas de casos de uso del sistema son diagramas de comportamiento que muestran la relación entre los actores y los casos de uso en el sistema. Representa los distintos requisitos funcionales con los que debe cumplir la aplicación y cómo se relacionan con su entorno. Son muy fáciles de interpretar y muy útiles en la comunicación con los clientes. El diagrama de casos de uso del sistema que se quiere implementar se muestra en la Figura 2.9.

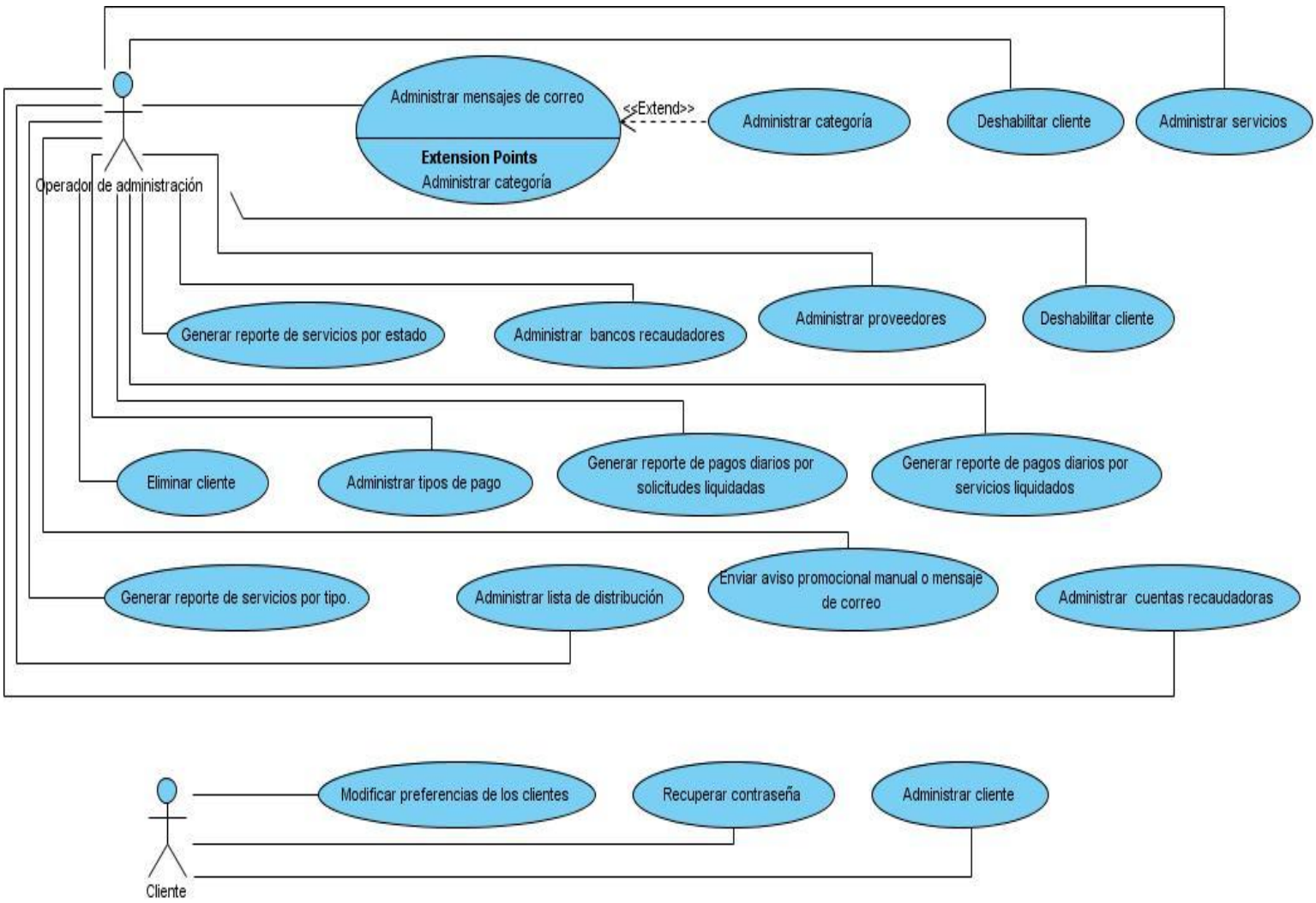


Figura 2.9: Diagrama de casos de uso (Fuente: elaboración propia)

Un actor del sistema es un rol que el usuario tiene en el mismo; puede ser una persona, una organización, otro sistema o una empresa. Los actores realizan una labor frente al sistema y pueden intercambiar información con éste. En la Tabla 2.2 se muestra los actores con sus descripciones.

Actor	Descripción
Cliente	Cualquier usuario que desee registrarse en el sistema, modificar sus preferencias y recuperar su contraseña.
Operador de administración	Persona encargada de realizar reportes, administrar servicios, bancos recaudadores, cuentas bancarias, tipos de pago, clientes, gestionar avisos del proveedor al cliente y el envío de

mensajes de correo electrónico y SMS.

Tabla 2.2: Descripción de los actores del sistema (Fuente: elaboración propia)

2.6.1 Especificación de los casos de uso del sistema

El diagrama de casos de uso no es suficiente para tener un buen entendimiento acerca de las funcionalidades que debe hacer el sistema, para ello se realiza una especificación de los casos de uso. Aquí es donde se describen paso por paso las acciones que se realizan en los casos de uso, detallando la acción del actor y la respuesta que le da el sistema, las precondiciones como estado inicial, las pos-condiciones como posibles estados finales y la prioridad del caso de uso. La descripción de los casos de uso del sistema de los módulos administración, reportes y clientes se pueden ver en el documento "Modelo de sistema (Módulos clientes, reportes y administración de SIB)", en la Tabla 2.3 se muestra la descripción del caso de uso "Administrar servicio"**.**

Caso de Uso:	**Administrar servicio**
Actores:	Operador de administración
Resumen:	El caso de uso inicia cuando el Operador de Administración ejecuta la opción **Administrar servicio**, donde se crean, modifican, habilitan, deshabilitan y eliminan los servicios, finalizando así el mismo.
Precondiciones:	El Operador de administración debe estar autenticado en el sistema.
Referencias	**RF1**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Operador de administración selecciona la opción **Administrar servicio.**	1.1- El sistema muestra el listado de los servicios, un menú desplegable con la opción: • **Nuevo** • **Habilitar** • **Deshabilitar** Por cada servicio un botón: • **Borrar** • **Editar**

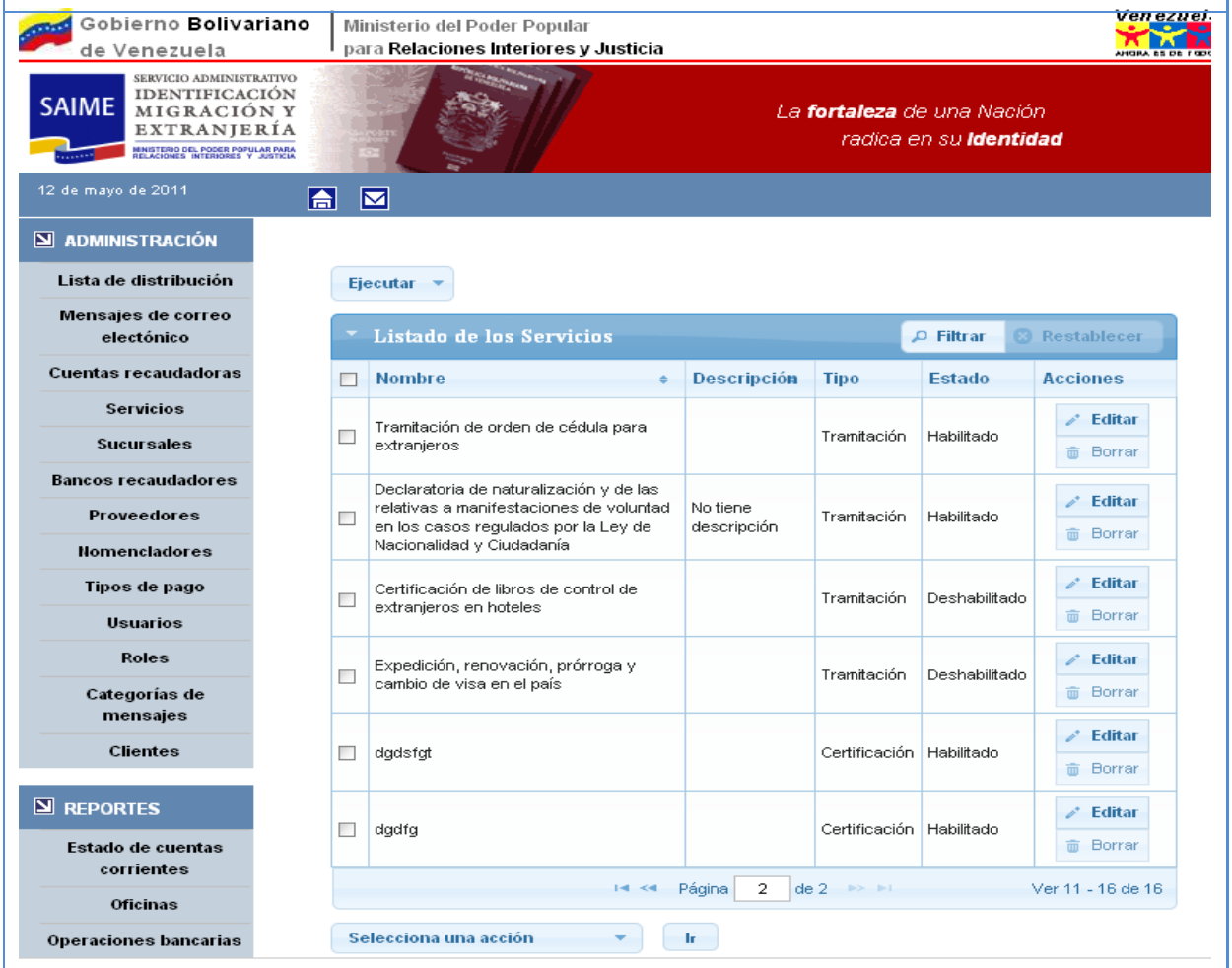
2- El Operador de administración selecciona: 2a) Nuevo, ver **"Sección 1 Crear servicio".** 2b) Editar, ver **"Sección 2 Modificar servicio".** 2c) Habilitar, ver **"Sección 3 Habilitar servicio".** 2d) Deshabilitar, ver **"Sección 4 Deshabilitar servicio".** 2e) Borrar, ver **"Sección 5 Eliminar servicio".**	
Sección 1 "Crear servicio"	
	1- El sistema muestra una interfaz con los siguientes campos a llenar. • **Nombre del servicio** • **Pago** • **Precio** • **Descripción** • **Tipo** • **Estado** • **Proveedor** Y las opciones: • **Guardar** • **Guardar y crear otro** • **Cancelar**
2- El Operador de administración introduce los datos y selecciona la opción: 2a) **Guardar**, continua en la **Sección 1 "Crear servicio"** acción 2.1. 2b) **Cancelar,** ver **Flujo alterno 1 "Cancelar Acción".** 2c) **Guardar y crear otro,** continúa en la **Sección 1 "Crear servicio"** de la acción 2.1 a 2.3 y regresa a la acción 1.	2.1- El sistema valida que no haya campos vacíos. En caso contrario, ver **Flujo alterno 2 "Campos vacíos de crear".**
	2.2- El sistema valida que los datos estén correctos. En caso contrario, ver **Flujo alterno 3**

	"Datos incorrectos de crear".
	2.3- El sistema muestra un mensaje notificando el éxito de la acción y guarda los cambios efectuados.
Sección 2 "Modificar servicio"	
	1- El sistema muestra una interfaz con los servicios filtrados por: • **Nombre** • **Estado** • **Proveedor** • **Creado**
2- El Operador de administración selecciona el servicio que desea modificar.	2.1- El sistema muestra una interfaz con los campos del servicio para ser modificados. • **Nombre del servicio** • **Pago** • **Precio** • **Descripción** • **Tipo** • **Estado** • **Proveedor** y las opciones: • **Guardar** • **Cancelar**
3- El Operador de administración modifica los datos deseados y selecciona la opción: 3a) **Guardar**, continúa en la **Sección 2 "Modificar servicio"** acción 3.1. 3b) **Cancelar,** ver **Flujo alterno 1 "Cancelar acción".**	3.1- El sistema valida que no haya campos vacíos. En caso contrario, ver **Flujo Alterno 4 "Campos vacíos de modificar".**
	3.2- El sistema valida que los datos estén correctos. En caso contrario, ver **Flujo Alterno 5 "Datos incorrectos de modificar".**
	3.3- En caso de que ocurran cambios en el precio de dicho servicio, el sistema busca los clientes afectados por tener el servicio

	solicitado.
	3.4- El sistema notifica del cambio a los clientes afectados.
	3.5- El sistema renombra el servicio en todas las solicitudes activas que contengan dicho servicio, en caso de que el cambio sea para el nombre.
	3.6- El sistema muestra un mensaje notificando el éxito de la acción.
	3.7 El sistema guarda los cambios efectuados.
Sección 3 "Habilitar servicio"	
	1- El sistema muestra una interfaz con los servicios filtrados por: • **Servicios deshabilitados.**
2- El Operador de administración habilita el servicio que desea.	2.1- El sistema muestra un mensaje notificando el cambio.
	2.2- El sistema muestra un mensaje notificando el éxito de la acción.
	2.3 El sistema guarda los cambios efectuados.
Sección 4 "Deshabilitar servicio"	
	1- El sistema muestra una interfaz con los servicios filtrados por: • **Servicios habilitados.**
2- El Operador de administración deshabilita el servicio que desea.	2.1- El sistema muestra un mensaje notificando el cambio.
	2.2- El sistema muestra un mensaje notificando el éxito de la acción.
	2.3 El sistema guarda los cambios efectuados.
Sección 5 "Eliminar servicio"	
	1- El sistema muestra una interfaz con los servicios filtrados por: • **Nombre** • **Estado** • **Proveedor** • **Creado**
2- El Operador de administración borra el	2.1- El sistema muestra un mensaje preguntando si

o los servicios que desee eliminar.	está seguro de realizar esa acción y las opciones: • **Aceptar** • **Cancelar**
3- El Operador de administración selecciona la opción: 2a) **Aceptar**, continúa en la **Sección 5 "Eliminar servicio"** parte 3.1. 2b) **Cancelar**, ver **Flujo alterno 1 "Cancelar acción".**	3.1- El sistema verifica que los servicios seleccionados no estén solicitados en ninguna solicitud activa. En caso contrario, ver **Flujo Alterno 6 "Cancelar eliminación"**
	3.2- El sistema muestra un mensaje notificando el éxito de la acción.
	3.3 El sistema guarda los cambios efectuados.

Prototipo de Interfaz



Flujos alternos

Flujo alterno 1 "Cancelar acción"

Acción del Actor	Respuesta del Sistema

	1- El sistema reinicia las variables y regresa a la acción 1.1 del **Flujo Normal de Eventos**.
Flujo alterno 2 "Campos vacíos de crear"	
	2- El sistema muestra un mensaje notificando al usuario que ha dejado campos vacíos y regresa a la **Sección 1 "Crear Servicio"** acción 2.
Flujo alterno 3 "Datos incorrectos de crear"	
	3- El sistema muestra un mensaje notificando al usuario que no ha llenado los campos correctamente y regresa a la **Sección 1 "Crear servicio"** acción 2.
Flujo alterno 4 "Campos vacíos de modificar"	
	4- El sistema muestra un mensaje notificando al usuario que ha dejado campos vacíos y regresa a la **Sección 2 "Modificar servicio"** acción 3.
Flujo alterno 5 "Datos incorrectos de modificar"	
	5- El sistema muestra un mensaje notificando al usuario que no ha llenado los campos correctamente y regresa a la **Sección 2 "Modificar servicio"** acción 3.
Flujo alterno 6 "Cancelar eliminación"	
	6- El sistema muestra un mensaje que no se puede efectuar la eliminación del servicio ya que está solicitado en una solicitud activa y regresa a la **Sección 5 "Eliminar servicio"** acción 2.
Prototipo de Interfaz	
Pos-condiciones	Quedan administrados los servicios.

Tabla 2.3: Descripción del caso de uso "Administrar servicio" (Fuente: elaboración propia)

2.7 Análisis

El objetivo de este flujo es analizar los requisitos descritos en los requerimientos, mediante su refinamiento y estructuración, a fin de: lograr una compresión más precisa de

los requisitos y obtener una descripción de éstos, que sea fácil de mantener y que ayude a estructurar el sistema en su conjunto.

2.7.1 Diagrama de clases del análisis

El diagrama de clases del análisis ayuda a refinar los requerimientos y permite a los desarrolladores describir la estructura interna del sistema. Ofrece mayor expresividad y formalización. Se realiza usando el lenguaje del desarrollador y representa la vista interna del sistema. Este diagrama no debe contener redundancias ni inconsistencias en la interpretación de los requerimientos, ya que proyecta como realizar la funcionalidad dentro del sistema. Se usa para que los desarrolladores comprendan como el sistema debe ser diseñado e implementado. El diagrama de clases del análisis del caso de uso "Administrar servicio" se muestra en la Figura 2.10, los restantes diagramas referentes a los otros casos de uso de los módulos se pueden ver en el Anexo 2.

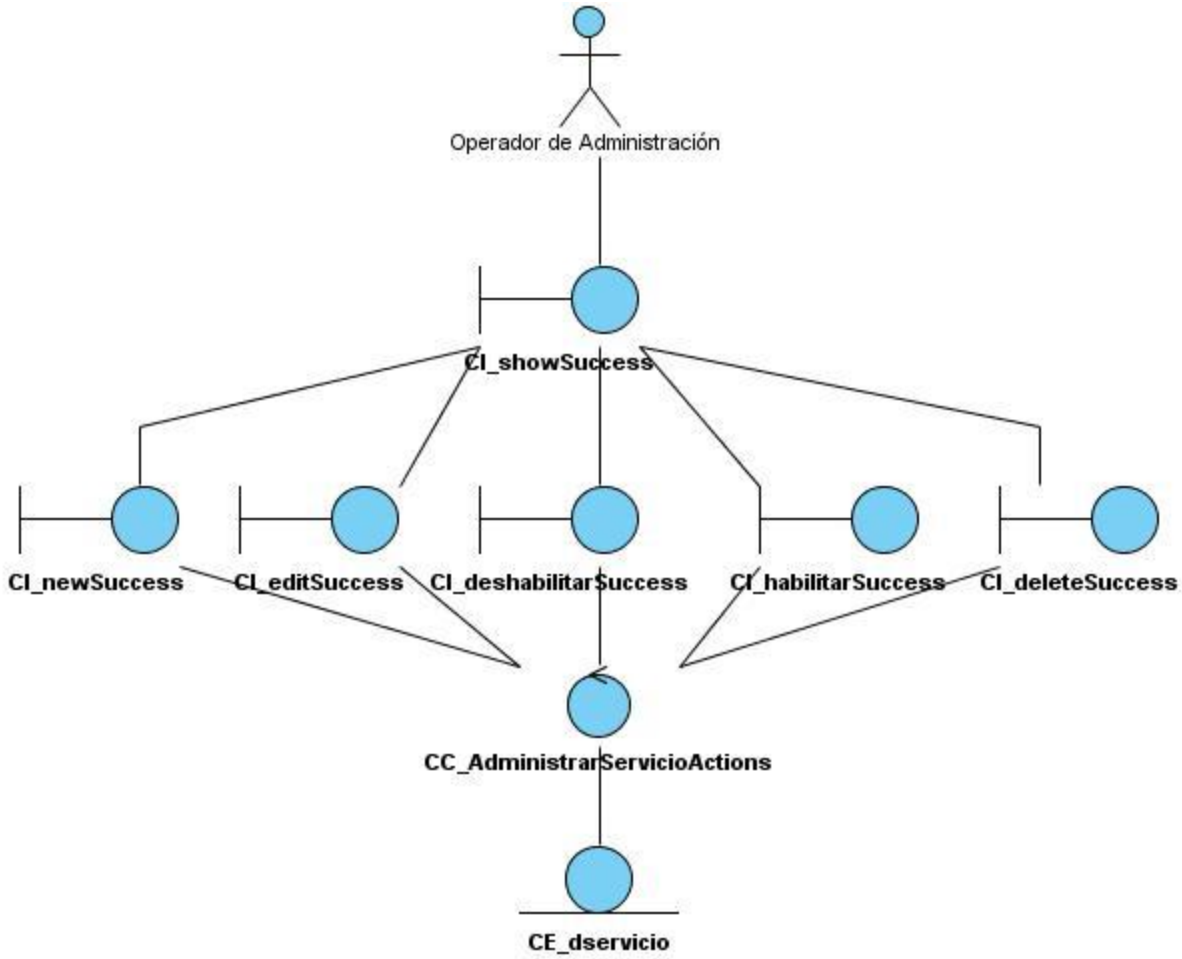


Figura 2.10: Diagrama de clases del análisis del caso de uso "Administrar servicio" (Fuente: elaboración propia)

2.8 Diseño

El diseño de un sistema se realiza con el objetivo de refinar el análisis y tiene en cuenta los requisitos no funcionales, es decir, cómo cumple el sistema sus objetivos. En el diseño se modela el sistema, incluyendo la arquitectura que soportará a los requisitos. Cuando se realiza un buen diseño el sistema puede ser implementado sin imprecisiones. En ocasiones la implementación puede ser hecha por un generador automático de código.

2.8.1 Diagrama de clases del diseño

Los diagramas de clases del diseño muestran una visión que describe las especificaciones de las clases del *software* y de las interfaces en una aplicación, se usan principalmente para modelar la vista del diseño estática de un sistema. Este diagrama incluye las clases con sus atributos y métodos, así como las asociaciones, dependencias, navegabilidad y las interfaces con sus operaciones y constantes. El diagrama de clases del diseño del caso de uso “Administrar servicio” se muestra en la Figura 2.10, los restantes diagramas referentes a los otros casos de uso de los módulos se pueden ver en el Anexo 3.

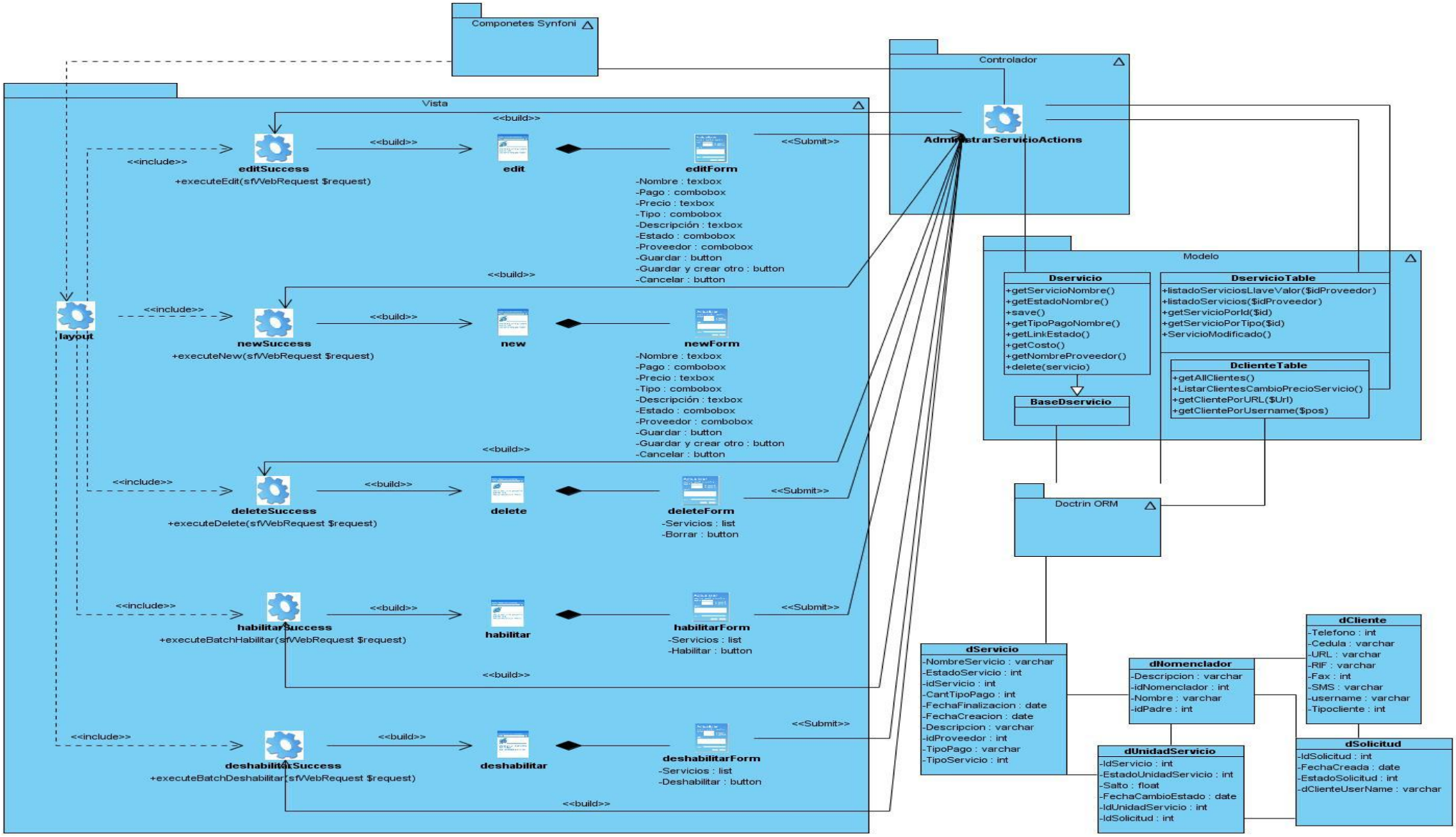


Figura 2.11: Diagrama de clases del diseño del caso de uso "Administrar servicio" (Fuente: elaboración propia)

2.8.2 Diagrama de interacción

El diagrama de secuencia describe la dinámica del sistema. A menos que se modele un sistema muy pequeño, resulta muy difícil representar toda la dinámica de un sistema en un único diagrama. Por lo tanto, la dinámica completa se representará mediante un grupo de diagramas de secuencia, cada uno de ellos vinculados generalmente a una sub-sunción del sistema. El diagrama de secuencia describe las interacciones de un grupo de objetos, mostrando de forma secuencial los envíos de mensajes entre objetos. El diagrama puede así mismo mostrar los flujos de datos intercambiados durante el envío de mensajes. (Debrauwer, y otros, 2005)

Los diagramas de secuencia del diseño de las diferentes secciones del caso de uso "Administrar servicio" se muestran en las Figura 2.12, Figura 2.13, Figura 2.14, Figura 2.15 y Figura 2.16.

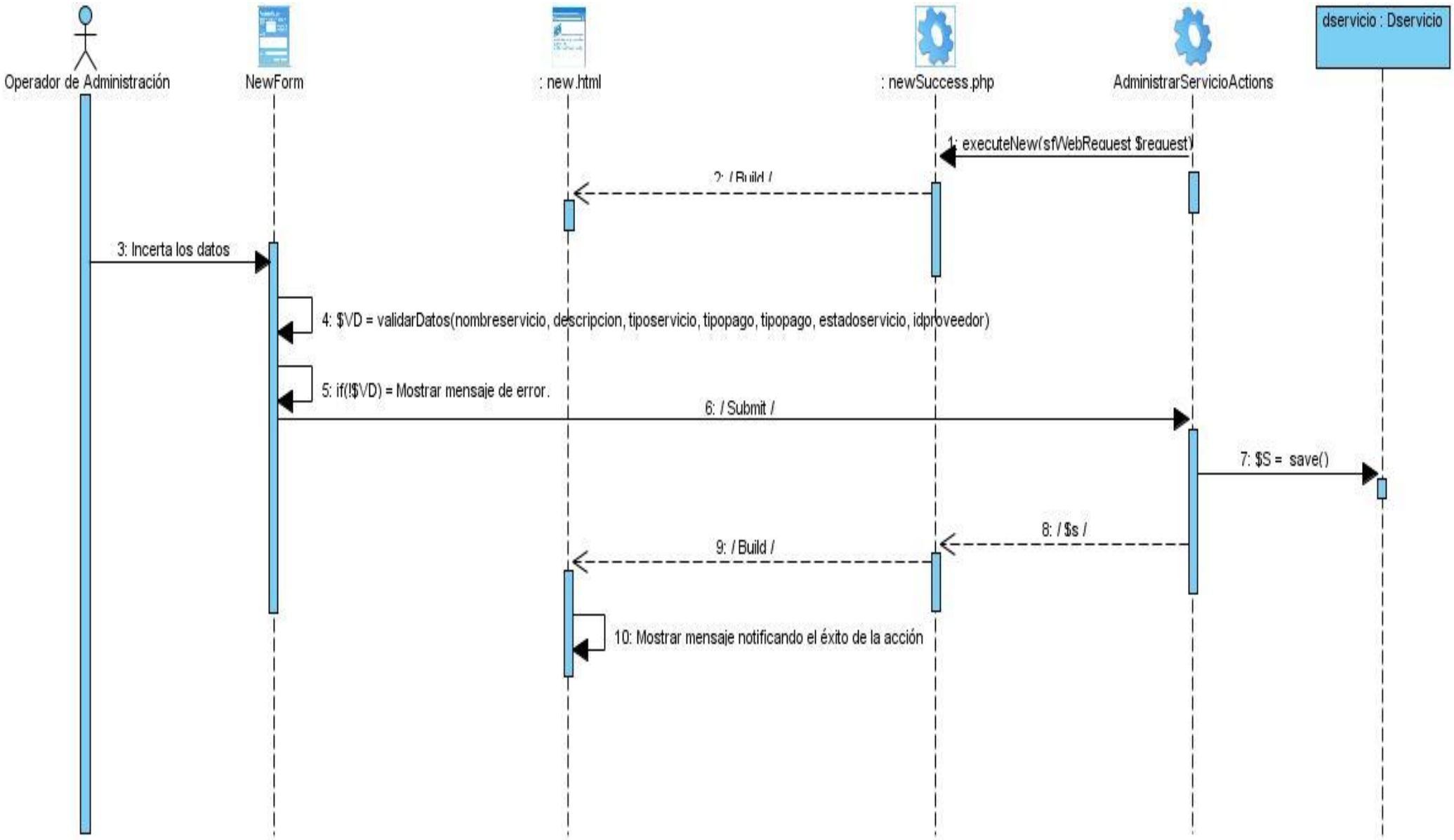


Figura 2.12: Diagrama de secuencia "Crear servicio" del caso de uso "Administrar servicio" (Fuente: elaboración propia)

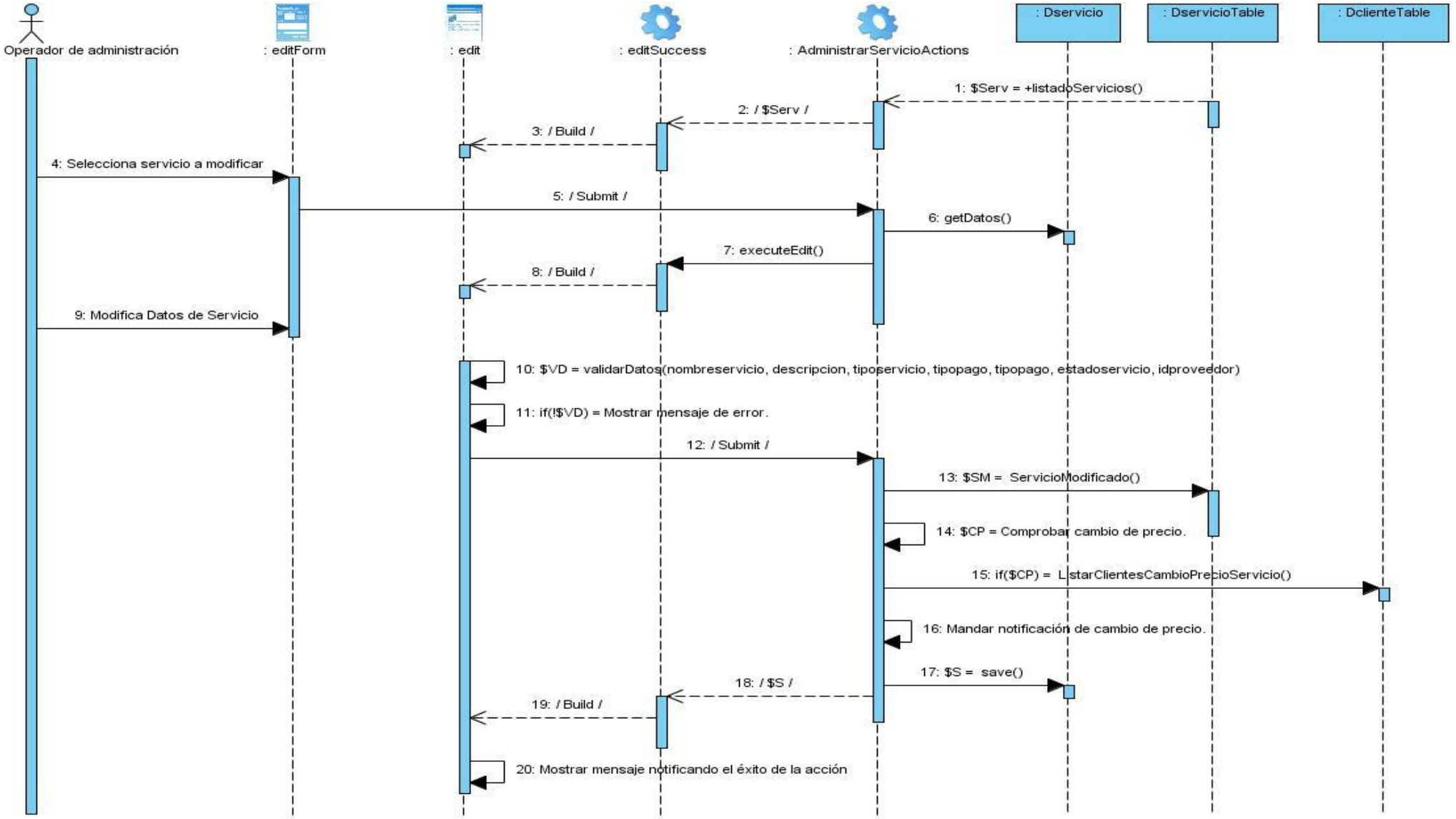


Figura 2.13: Diagrama de secuencia "Modificar servicio" del caso de uso "Administrar servicio" (Fuente: elaboración propia)

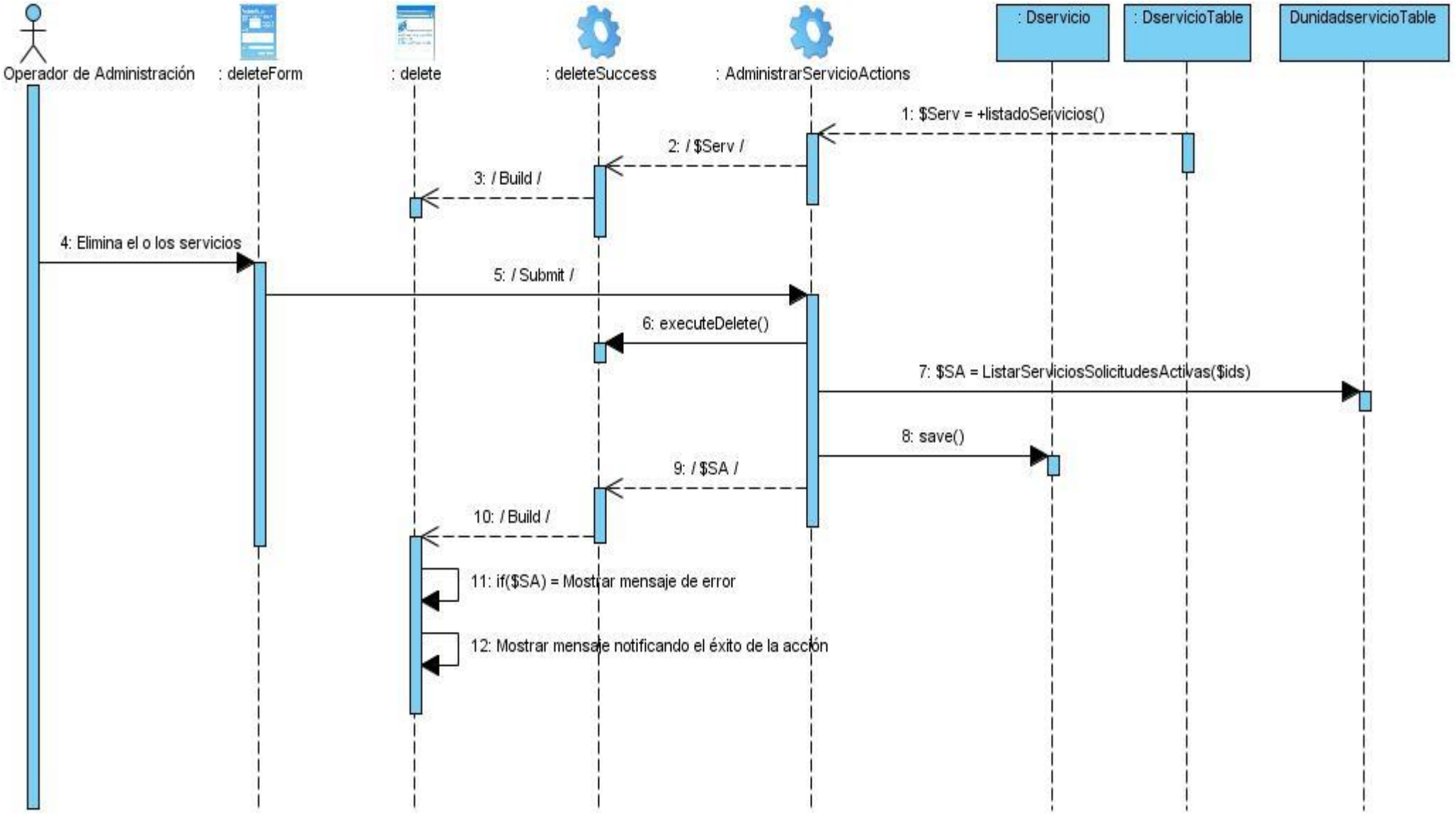


Figura 2.14: Diagrama de secuencia "Eliminar servicio" del caso de uso "Administrar servicio" (Fuente: elaboración propia)

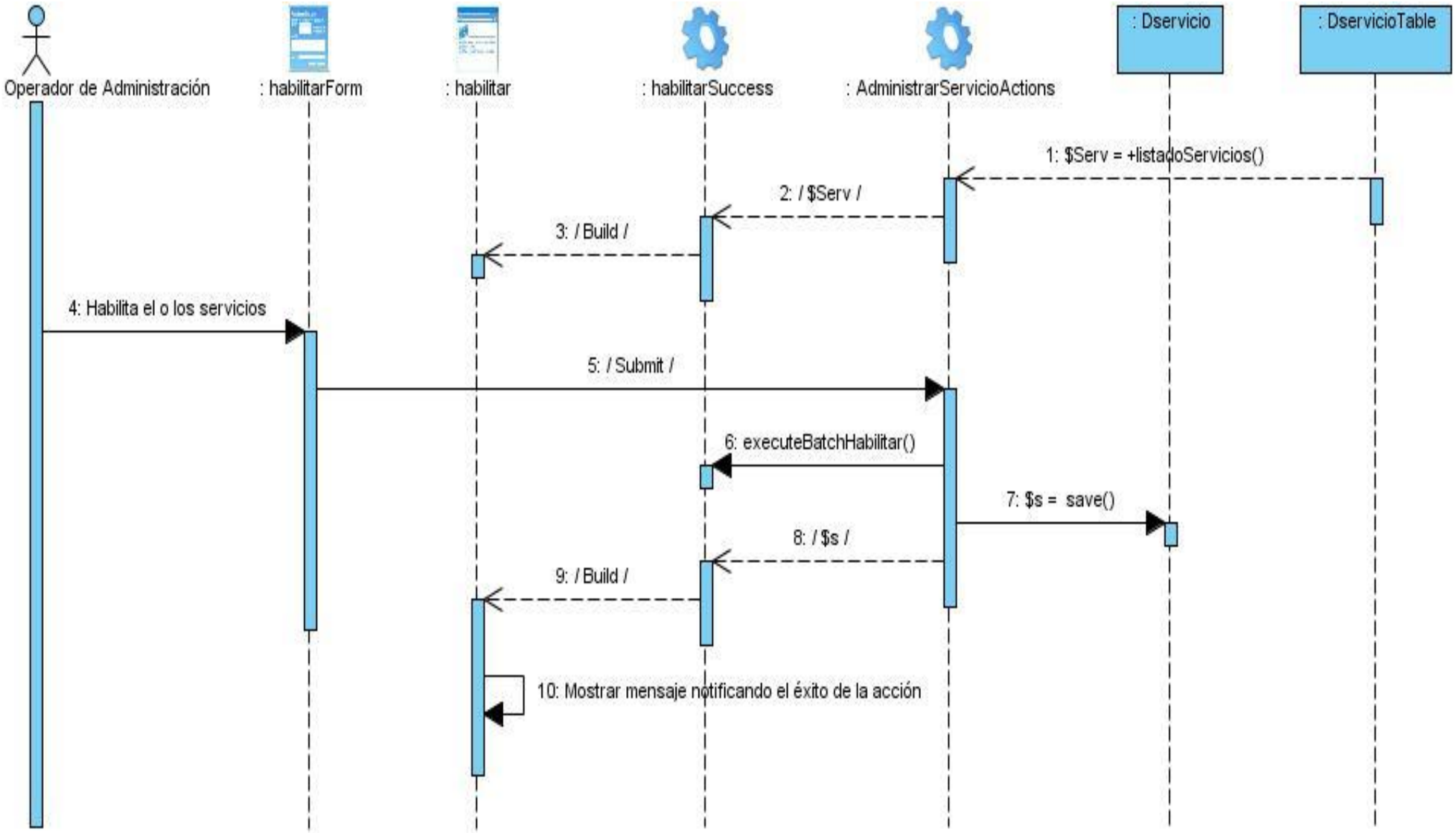


Figura 2.15: Diagrama de secuencia "Habilitar servicio" del caso de uso "Administrar servicio" (Fuente: elaboración propia)

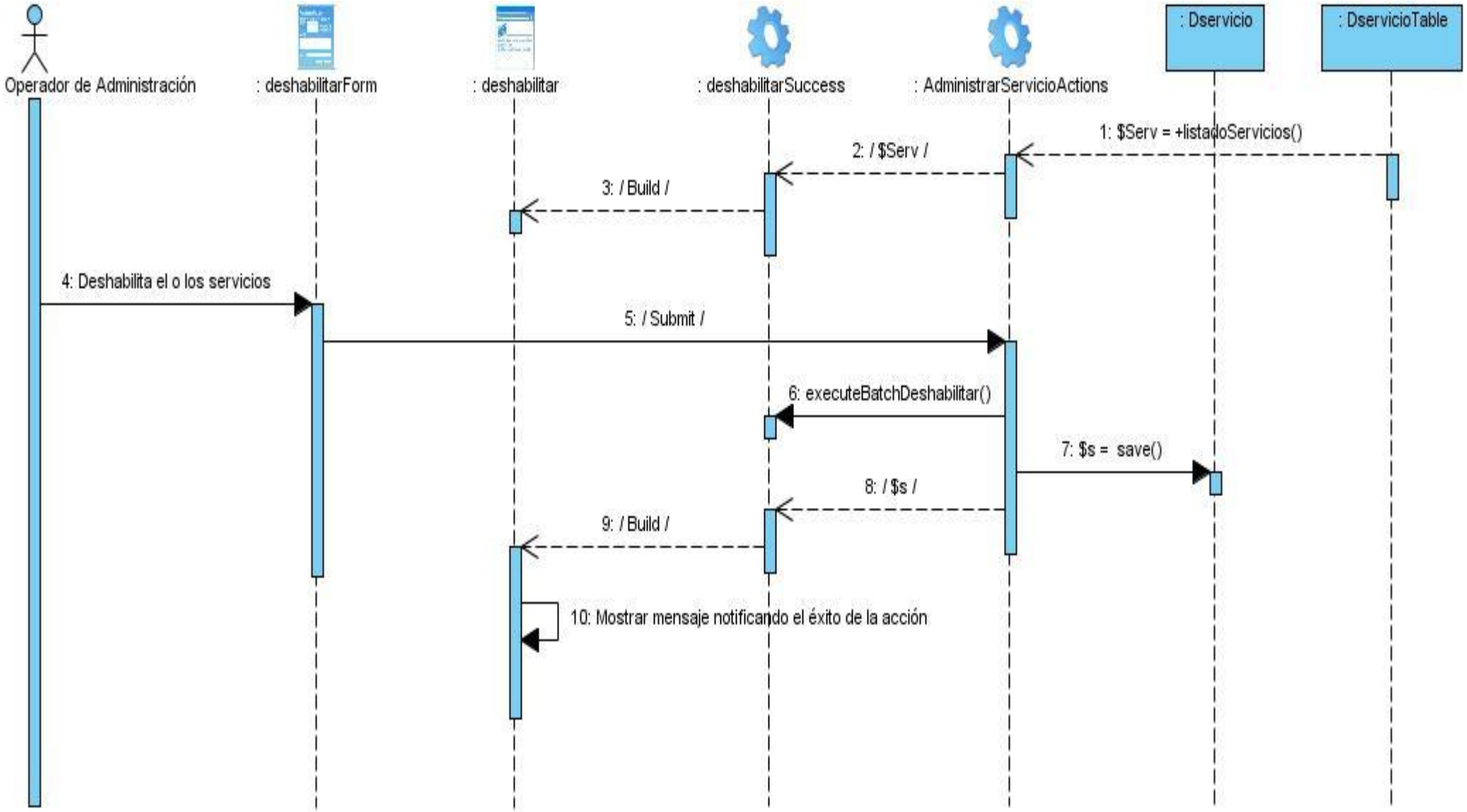


Figura 2.16: Diagrama de secuencia "Deshabilitar servicio" del caso de uso "Administrar servicio" (Fuente: elaboración propia)

2.8.3 Diseño de la base de datos

El diseño de la base de datos suele apoyarse en diagramas, a través de los cuales se trata de visualizar las diferentes entidades que intervienen, las relaciones entre ellas y el tipo de estas relaciones. Estos gráficos son de ayuda para decidir las distintas tablas que deben ser utilizadas en la base de datos. (Cobo, y otros 2005)

El diagrama entidad-relación de la base de datos del sistema representado en la Figura 2.17 se encuentra compuesto por 23 tablas. Éstas se componen de dos estructuras, los campos y los registros. Los campos, son los nombres de las columnas, deben ser únicos y tener un tipo de dato asociado, y los registros que corresponden a las filas que componen la base de datos, allí se forman los datos y los registros. Las tablas serán las encargadas de almacenar la información recogida desde el sistema. En la Tabla 2.4 se explica la función de cada tabla en la base de datos.

CAPITULO 2: PROPUESTA DE SOLUCIÓN

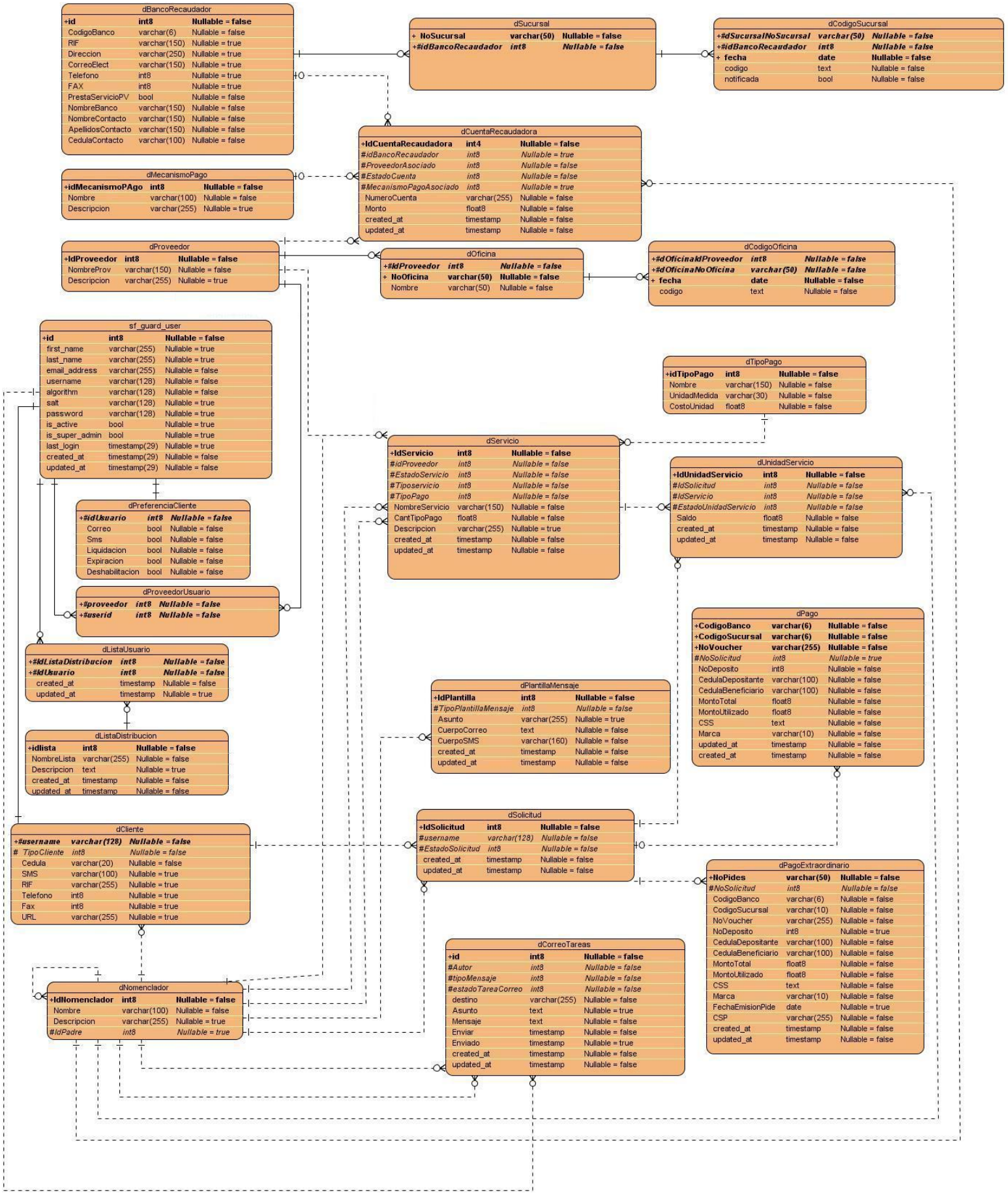


Figura 2.17: Diagrama entidad relación (Fuente: elaboración propia)

Descripción del modelo de datos

Tablas	Descripción
Sf_guard_user	Guarda todos los usuarios del sistema.
dListaUsuario	Guarda los indicadores de las tablas *Sf_guard_user* y *dListaDistribucion*, producto de la relación: muchos a muchos que existe entre ellas.
dListaDistribucion	Guarda las listas de distribución creadas para el envío de correos electrónicos.
dCorreoTareas	Guarda todos los correos o SMS (servicio de mensajes cortos) que serán enviados.
dPlantillaMensaje	Guarda el diseño de las plantillas existentes para el envío de mensajes.
dCliente	Guarda los datos de los clientes del sistema.
dNomenclador	Guarda todos los campos multievaluados del sistema.
dPreferenciaCliente	Guarda los datos de las preferencias de los clientes.
dTipoPago	Guarda los datos de los tipos de pago del sistema.
dServicio	Guarda los datos de los servicios que brinda el proveedor.
dUnidadServicio	Guarda las unidades de los servicios que están en una solicitud.
dProveedor	Guarda los datos de los proveedores.
dProveedorUsuario	Guarda los indicadores de las tablas *Sf_guard_user* y *dProveedo,* producto de la relación: muchos a muchos que existe entre ellas.
dCuentaRecaudadora	Guarda los datos de las cuentas recaudadoras de los bancos.
dBancoRecaudador	Guarda los datos de los bancos recaudadores.
dOficina	Guarda los datos de las oficinas de los proveedores.
dCodigoOficina	Guarda los códigos que se generan para las oficinas.
dSolicitud	Guarda los datos de las solicitudes realizadas por los clientes.
dPago	Guarda los datos de los pagos realizados por los clientes.
dPagoExtraordinario	Guarda los datos de los pagos extraordinarios que se crean.

dMecanismoPago	Guarda los datos de los mecanismos de pagos de las cuentas recaudadoras.
dSucursal	Guarda los datos de las sucursales de los bancos.
dCodigoSucursal	Guarda los códigos que se generan para las sucursales.

Tabla 2.4: Descripción del modelo de datos (Fuente: elaboración propia)

2.8.4 Definiciones del diseño a aplicar

Desarrollo de prototipos de interfaz de usuario

La interfaz ha de ser lo más uniforme posible, utilizando un mismo sistema de colores para realizar las interfaces de SAIME, de forma consistente y razonable, tratando de utilizar colores iguales o similares en todas las páginas, con textos concisos y claros, además sin mezclar muchos tipos de letra y tamaños en cada una. La navegación debe ser lo más rápida posible, por lo que la presencia de imágenes innecesarias y todo lo que atente contra una navegación rápida y eficiente debe ser eliminado.

Para obtener un mejor entendimiento y desarrollo del sistema se comienzan a definir y a diseñar las principales vistas que van a interactuar con el usuario final, en busca de aminorar la complejidad y abstracción, con el objetivo de elevar el nivel de satisfacción del cliente y garantizar el entendimiento con el mismo. En la Figura 2.18 se muestra el prototipo desarrollado para la interfaz principal.

Figura 2.18: Prototipo de interfaz principal (Fuente: elaboración propia)

Tratamiento de errores

Para que la aplicación funcione de manera segura, es importante comunicarle al usuario cuando es incorrecta la operación que se está efectuando. Para ello se realizará una validación y control a nivel de interfaz y el tratamiento de excepciones a nivel de código, detectándolas y mostrando la información sobre la excepción.

El sistema que se propone permitirá minimizar y tratar los posibles errores, así se podrá garantizar la integridad y confiabilidad de la información que se maneje. Los mensajes de error que se emitirán se mostrarán en un lenguaje de fácil comprensión para los usuarios, alertándolos de la introducción incorrecta de la información, así como de la falta de datos.

2.9 Conclusiones parciales

Con la correcta realización de los tres primeros flujos de la metodología “Proceso Unificado de Desarrollo” se logra una buena comprensión del negocio y la interpretación correcta de las necesidades del sistema. La descripción detallada de los procesos permitió obtener una mejor visión de los requisitos especificados por el cliente.

Se pudieron definir los casos de uso a automatizar, los trabajadores del sistema y una arquitectura robusta, sirviendo de base a los pasos posteriores de implementación y prueba. La realización del análisis y el diseño, mediante la confección de diferentes diagramas que describen cómo se realizarán los procesos, qué clases de *software* se utilizarán, y las interfaces mediante las cuales los usuarios interactuarán con el sistema, posibilita lograr un mejor entendimiento de las funcionalidades que se utilizarán en los módulos de administración, reportes y clientes, permitiendo la preparación necesaria para realizar la implementación de los mismos.

Capítulo 3
Implementación

3.1 Introducción

La mayor parte de la arquitectura del sistema es definida durante el diseño, siendo el propósito principal de la implementación desarrollar la arquitectura y el sistema como un todo. Se realiza el diagrama de despliegue y se describe cómo quedará el sistema una vez que haya sido desplegado y se elaboran los diagramas de componentes mostrando la forma en la que se desarrolló la aplicación.

3.2 Descripción de las principales clases

En las siguientes tablas se muestran las principales clases de la implementación que se generan por cada módulo, en este caso, para el módulo de "*GestionarClientes*" que da respuesta al caso de uso "Administrar cliente".

En la clase *"GestionarClienteActions"* se programan los métodos que ejecutan las plantillas, éstas almacenan todas las acciones del módulo. Ver en la Tabla 3.5 la descripción de esta clase.

Nombre: ***GestionarClienteActions***	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
autorizar($username)	Responsable de autorizar a los usuarios realizar la acción que deseen en dependencia de los permisos de acceso.
executeIndex()	Responsable de ejecutar su plantilla llamada *indexSuccess*, la cual redirige la acción a la página de bienvenida del módulo.
executeShow()	Responsable de ejecutar su plantilla llamada *showSuccess*, la cual redirige la acción a la página donde se muestran los datos del módulo.
executeNew()	Responsable de ejecutar su plantilla llamada *newSuccess*, la cual redirige la acción a la página donde se muestra el formulario para crear un cliente.

executeEdit()	Responsable de ejecutar su plantilla llamada *editSuccess, la cual* redirige la acción a la página donde se muestra el formulario para modificar los datos de un cliente.
processForm()	Responsable de validar los datos de los formularios y almacenarlos. Si se crea un cliente redirige al método enviar un correo.
enviarCorreoActivacion()	Responsable de enviar al cliente un correo con su URL (localizador uniforme de recurso) única para registrarse en el sitio.
executeActivarcuenta()	Responsable de activar la cuenta del cliente cuando este acceda a su URL única.

Tabla 3.5: Descripción de la clase GestionarClienteActions (Fuente: elaboración propia)

En la clase "*DclienteForm"* se configuran los campos del formulario. Ver en la Tabla 3.6 la descripción de esta clase.

Nombre: *DclienteForm*	
Tipo de clase: Formulario	
Para cada responsabilidad:	
Nombre:	**Descripción:**
configure()	Responsable de configurar todos los campos del formulario.
doSave()	Responsable de guardar los valores entrados a través del formulario.
valorPorDefecto()	Responsable de devolver el valor cuando es nuevo o cuando ya tiene valores.

Tabla 3.6: Descripción de la clase DclienteForm (Fuente: elaboración propia)

En la clase "*Dcliente"* se programan los métodos de acceso a datos. Ver en la Tabla 3.7 la descripción de esta clase.

Nombre: *Dcliente*
Tipo de clase: Acceso a datos.

Para cada responsabilidad:	
Nombre:	**Descripción:**
getTipoclienteComoTexto()	Devuelve el tipo de cliente.
getNombre()	Devuelve el nombre del cliente.
getApellidos()	Devuelve los apellidos del cliente.
getEmail()	Devuelve el correo electrónico del cliente.
activarCliente()	Responsable de activar la cuenta del cliente en el sistema.
desactivarCliente()	Responsable de desactivar la cuenta del cliente en el sistema.
getActive()	Devuelve el estado del cliente (activado o desactivado).
save()	Responsable de guardar los datos que se introduzcan en el formulario.

Tabla 3.7: Descripción de la clase Dcliente (Fuente: elaboración propia)

En la clase "*DclienteTable"* se programan los métodos de acceso a datos. Ver en la Tabla 3.8 la descripción de esta clase.

Nombre: *DclienteTable*	
Tipo de clase: Acceso a datos.	
Para cada responsabilidad:	
Nombre:	**Descripción:**
getAllClientes()	Devuelve un listado con todos los clientes existentes.
getClientePorURL()	Devuelve un listado con todas las URL de los clientes existentes.
getListaClientesActivos()	Responsable de listar los clientes que están activos en el sistema.
getClientePorUsername($pos)	Responsable de listar los clientes que tengan el usuario igual al pasado por parámetro.

Tabla 3.8: Descripción de la clase DclienteTable (Fuente: elaboración propia)

3.3 Diagrama de componentes

El diagrama de componentes representa la organización y dependencia de un conjunto de componentes. Pertenece a la vista estática de la construcción del sistema, ya que los componentes son mapeados en clases, interfaces y colaboraciones. (Ledesma)

En la Figura 3.19 se representa el diagrama de componentes general. El **Formulario** contiene el componente ¨*Form.class.php*¨, donde se realiza la configuración del mismo y utiliza los componentes del **Modelo**, este último contendrá tres clases por cada tabla de la base de datos, ¨*Clase.class.php*¨, ¨*ClaseTable.class.php*¨ y ¨*BaseClase.class.php*¨ en las cuales se implementaron las funcionalidades de acceso a datos, utilizando el *framework* Doctrine para tener acceso a los mismos. El componente ¨*Actions.class.php*¨ que se encuentra en el **Controlador** ejecuta las planillas que estarán en la **Vista**, la cual utilizará CSS que contendrá las hojas de estilo en cascada, el componente *jQuery* que permite enriquecer la aplicación web con listas, tablas o efectos y *jQuery UI* que proporciona abstracciones de nivel de baja interacción y animación, efectos avanzados y de alto nivel. Estos cuatro subsistemas de implementación utilizarán el *framework* Symfony para su funcionamiento, el cual contiene el componente *config* que se encargará de la configuración de todo el sistema y el componente *lib* donde se encontrarán las librerías utilizadas. Los diagramas de componentes de los módulos clientes, reportes y administración se pueden ver en el Anexo 4.

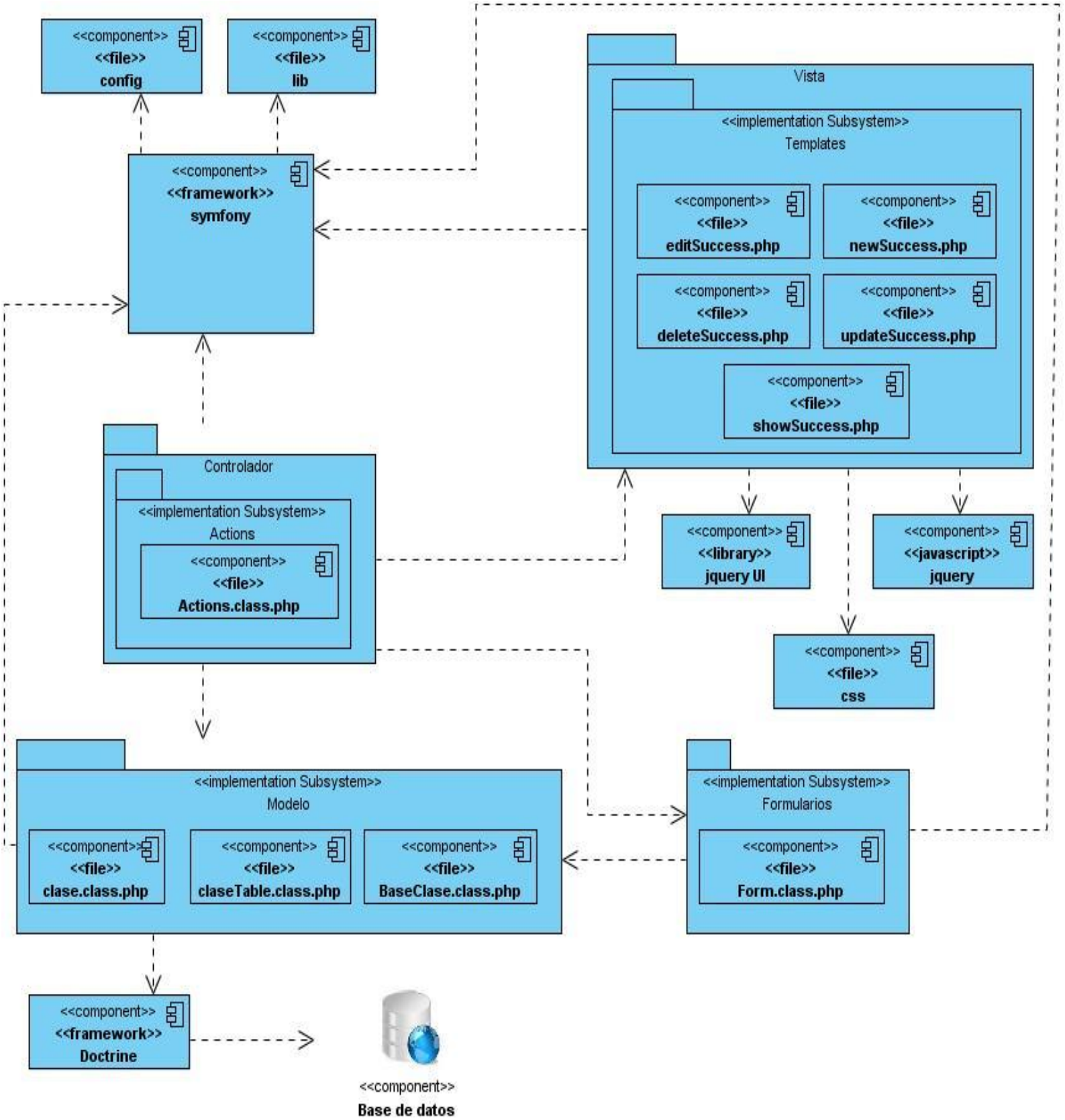


Figura 3.19: Diagrama de componentes general (Fuente: elaboración propia)

3.4 Diagrama de despliegue

El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto al *hardware* y *software.* Se utiliza en el diseño e implementación, donde se distinguen los componentes y nodos, así como las relaciones entre estos. (Falgueras, 2003)

El despliegue del sistema estará ubicado en el Centro Datos SAIME, ahí se hospedará la aplicación web. Existirán tres servidores de aplicaciones; uno recibirá las peticiones y las re direccionarán a los otros dos para que no ocurra sobrecargo.

Existirá comunicación con los bancos a través de servidores SFTP, donde se colocarán y leerán archivos XML indistintamente. Debido a restricciones en algunos bancos se ubicará un servidor en el Centro Datos SAIME y otro en el banco. El sistema notificará las eventualidades a sus clientes y a los bancos recaudadores registrados por medio de correo electrónico, haciendo uso del servidor de correo electrónico del SAIME. En el centro de datos se necesitarán dos servidores para la base datos (uno principal y uno de respaldo).

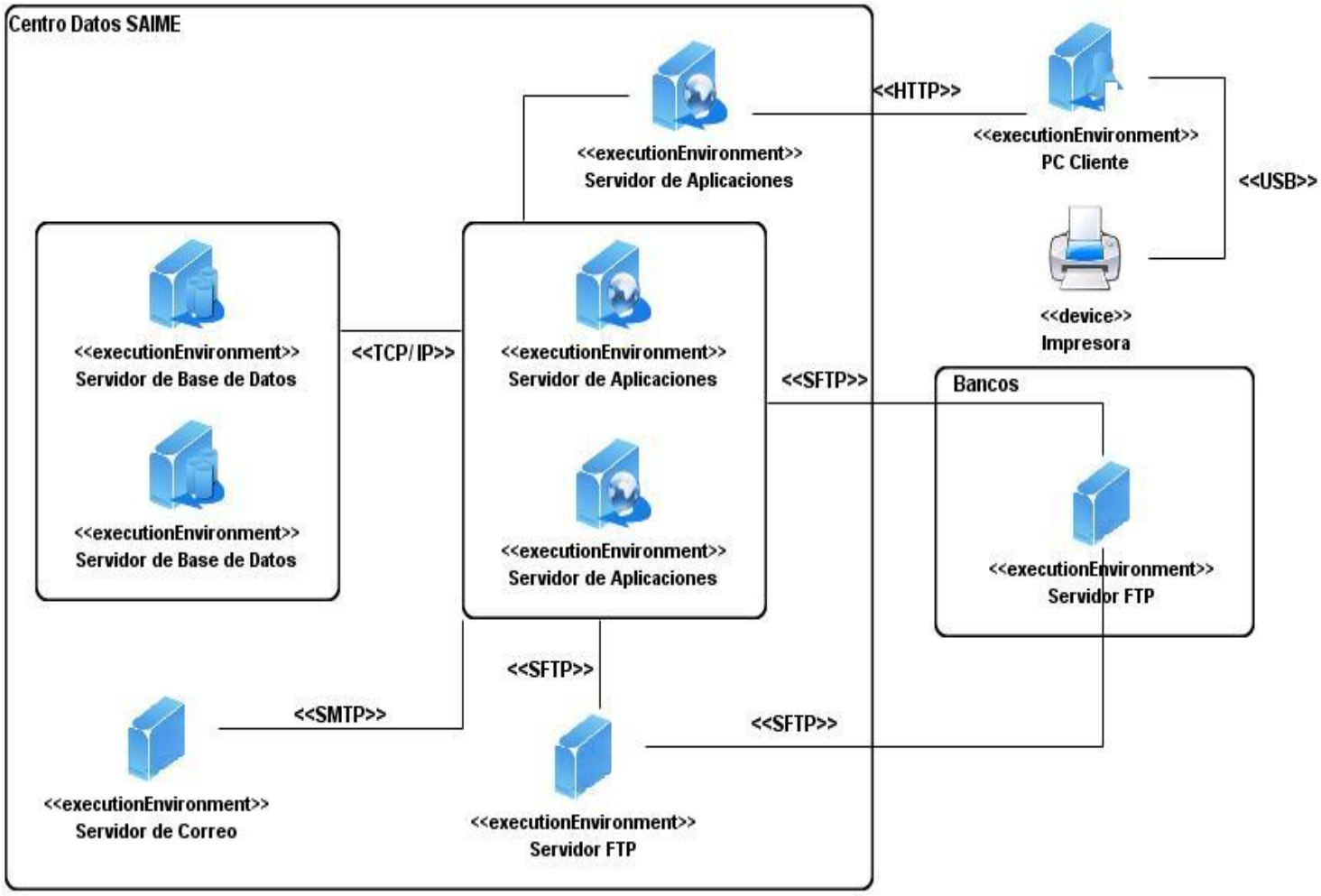


Figura 3.20: Diagrama de despliegue (Fuente: elaboración propia)

3.5 Estándares de código

Los estándares de codificación son pautas que indican cómo va a estar estructurada la programación, para facilitar la lectura, comprensión y mantenimiento del código. Estos establecerán patrones que permitan lograr un código más legible, reutilizable y que sirvan de guía durante el desarrollo y la implementación, desde el punto de vista arquitectónico, estandarizando el código a implementar. Para la implementación del sistema propuesto se seguirán los siguientes estándares de codificación:

Principios generales:

- Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.

- La mayoría de los elementos se deben nombrar usando sustantivos (posiblemente compuestos), o formas verbales en infinitivo.
- La forma de construir los nombres será colocando primero el verbo o el sustantivo, seguido de cada uno de sus complementos con la primera letra en mayúscula.

Nomenclatura de las clases

Los nombres de las clases comienzan con mayúscula, en caso de ser nombres compuestos, el segundo nombre comenzará del mismo modo. Se usaron palabras completas para evitar acrónimos y abreviaturas.

Nomenclatura de las funciones

El nombre a emplear para cada una de las funciones comienza con mayúscula, en caso de ser nombres compuestos, el segundo nombre comenzará del mismo modo y con sólo leerlo se reconoce el propósito de la misma. Para los métodos analizadores de atributos, se usa el estándar "*getAtributo ()".*

Nomenclatura de las variables

El nombre a emplear para las variables comienza con minúscula.

3.6 Conclusiones parciales

La implementación es un flujo muy importante dentro de la metodología de desarrollo, ya que permite obtener como resultado la aplicación que se desea, pues se implementan las clases y subsistemas definidos durante el diseño y se distribuye el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue, quedando dispuesta para la realización de las pruebas.

Se definieron los artefactos necesarios para la implementación del sistema, la cual se realizó de forma efectiva. La aplicación podrá facilitar la disponibilidad de la información manejada, la administración de los datos de la empresa y el acceso a los servicios que ofrece SAIME, permitiendo así una comunicación activa entre el usuario y la información.

Capítulo 4
Pruebas a la solución

4.1 Introducción

Para obtener un *software* con calidad se hace necesario realizar varias pruebas a la aplicación que se desarrolló, pues el desarrollo del *software* implica una serie de actividades de producción donde las posibilidades de que aparezcan fallos humanos son muy grandes. Pueden presentarse errores debido a una incorrecta especificación de los requisitos, el uso indebido de las estructuras de datos y una errónea integración de los módulos, por lo que el desarrollo del *software* ha de ir acompañado de una actividad que garantice la calidad.

La planificación y ejecución de las pruebas a todo lo largo del ciclo de desarrollo del *software* permite evaluar el sistema o un componente del mismo. De esta forma se obtienen resultados que servirán de retroalimentación para eliminar los errores que surjan. En este capítulo se realiza la validación de la solución propuesta, con el objetivo de comprobar la eficiencia y calidad de los módulos, asegurando que se le da respuesta a los distintos requisitos planteados por el cliente.

4.2 Pruebas de software

Las pruebas del *software* son un elemento crítico para la garantía de la calidad del mismo y representa una revisión final de las especificaciones, el diseño y de la codificación. Es necesario señalar que en cada fase del ciclo de vida del desarrollo del *software* se plantean un conjunto de pruebas que permiten verificar que el *software* desarrollado satisface las especificaciones de esa fase. (Alonso Amo, y otros, 2005)

La calidad de un sistema se comprueba, entre otras cosas, por la coincidencia entre el, la aplicación que se desarrolló y los requisitos establecidos en la primera fase. Para determinar el grado de cumplimiento de estos requisitos se usan las pruebas del sistema. El objetivo de las pruebas de un programa es detectar un posible mal funcionamiento del sistema, donde un error puede ser costoso de reparar mientras más avanza la etapa del ciclo de vida del *software*.

Para escoger los tipos de pruebas que se van a utilizar es necesario verificar cuáles son los que mejor se adapten al sistema, por lo que hay que tener en cuenta el lenguaje de

programación que se utilizó, el tipo de funcionalidad que se implementa, la plataforma en que se ejecutan los procesos, los errores más importantes, si la aplicación es de escritorio o web, si realiza conexiones a bases de datos y qué herramientas de desarrollo y metodología se utilizaron.

4.3 Pruebas funcionales

Las pruebas funcionales, son pruebas de *software* que tienen por objetivo verificar que el sistema desarrollado cumpla con las funciones específicas para lo cual ha sido creado. A este tipo de pruebas se les denomina pruebas de comportamiento o pruebas de caja negra, que se basan en el análisis de los datos de entrada y salida. Éstas quedan definidas en los casos de prueba preparados antes del inicio de las mismas.

En las pruebas funcionales se especifican todos los escenarios posibles. Por cada escenario se validan las respuestas que da el sistema a cada una de las actividades que puede realizar el usuario en el mismo, además del resultado obtenido. Las variables pueden tomar distintos valores, válida (V), para cuando la entrada de datos es correcta y en caso contrario inválida (I) y no aplicada (NA), para cuando no es necesaria la validación de la variable en ese escenario.

4.3.1 Pruebas funcionales realizadas

Con el objetivo de verificar que el sistema cumpla con los requisitos especificados por el cliente, se realizaron casos de prueba de caja negra. En la Tabla 4.9 y Tabla 4.10 se muestra un caso de prueba diseñado para el caso de uso “Administrar servicio”, donde se realiza la descripción de las variables del mismo y los resultados de las diferentes iteraciones realizadas. El resto de los casos de pruebas se encuentran disponibles en el Anexo 5

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre del servicio.	Campo de texto	No	Debe introducir un texto.
[2]	Precio	Campo de texto	No	Debe introducir un texto.
[3]	Pago	Campo seleccionable	No	Debe seleccionar un valor

[4]	Descripción	Campo de texto	No	Debe introducir un texto.
[5]	Tipo.	Campo seleccionable	No	Debe seleccionar un valor
[6]	Estado.	Campo seleccionable	No	Debe seleccionar un valor
[7]	Proveedor	Campo seleccionable	No	Debe seleccionar un valor

Tabla 4.9: Descripción de las variables del caso de uso "Administrar servicio" (Fuente: elaboración propia)

Flujos	Var 1. Nombre	Var 2.Descripción	Var 3. Precio	Var 4. Pago	Var 5. Tipo	Var 6. Estado	Var 7. Proveedor	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear servicio.	V	V	V	V	V	V	V	*Crea un nuevo servicio.*	Satisfactoria	
	I	V	V	V	V	V	V	*Muestra el mensaje: "Introduzca el nombre del servicio".*	Satisfactoria	
	V	I	V	V	V	V	V	*Crea un nuevo servicio.*	Satisfactoria	
	V	V	I	V	V	V	V	*Muestra el mensaje "Introduzca un precio".*	Satisfactoria	
Modificar servicio	*V*	*V*	*V*	*V*	*V*	*V*	*V*	*Modifica el servicio.*	Satisfactoria	
	I	V	V	V	V	V	V	*Muestra el mensaje: "Introduzca el nombre del servicio".*	Satisfactoria	
	V	I	V	V	V	V	V	*Modifica el servicio.*	Satisfactoria	
	V	V	I	V	V	V	V	*Muestra el mensaje: "Introduzca un precio".*	Satisfactoria	

Flujos	Var 1. Nombre	Var 2.Descripción	Var 3. Precio	Var 4. Pago	Var 5. Tipo	Var 6. Estado	Var 7. Proveedor	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Habilitar servicio	V	NA	NA	NA	NA	NA	NA	*Habilita el servicio*	Satisfactoria	
	I	NA	NA	NA	NA	NA	NA	*Muestra el mensaje: "Se debe seleccionar al menos un elemento."*	Satisfactoria	
Deshabilitar servicio.	V	NA	NA	NA	NA	NA	NA	*Deshabilita el servicio*	Satisfactoria	
	I	NA	NA	NA	NA	NA	NA	*Muestra el mensaje: "Se debe seleccionar al menos un elemento."*	Satisfactoria	
Eliminar servicio.	V	NA	NA	NA	NA	NA	NA	*Elimina un servicio*	Satisfactoria	
	I	NA	NA	NA	NA	NA	NA	*Muestra el mensaje: "Se debe seleccionar al menos un elemento."*	Satisfactoria	

Tabla 4.10: Iteraciones de las variables del caso de uso "Administrar servicio" (Fuente: elaboración propia)

4.7 Validación de las variables

La aplicación de las pruebas al *software* se dividió en tres iteraciones. La primeras dos iteraciones fueron realizadas por el grupo de calidad del CISED, en la cual se aplicaron pruebas funcionales sobre las interfaces de los casos de uso que forman parte de los módulos de administración, clientes y reportes. En la primera iteración se encontraron cuatro casos de uso con dificultades y en la segunda iteración tres. La tercera iteración la realizó el centro Calisoft, donde se encontró un caso de uso con dificultad y dos no conformidades, emitiendo el acta de liberación del proyecto que se encuentra en el Anexo

6. Actualmente el sistema no posee ninguna dificultad ya que se le dio respuesta a las dificultades detectadas en la iteración tres. En la Figura 4.21 se muestra el comportamiento del desarrollo de los casos de uso en las tres iteraciones realizadas.

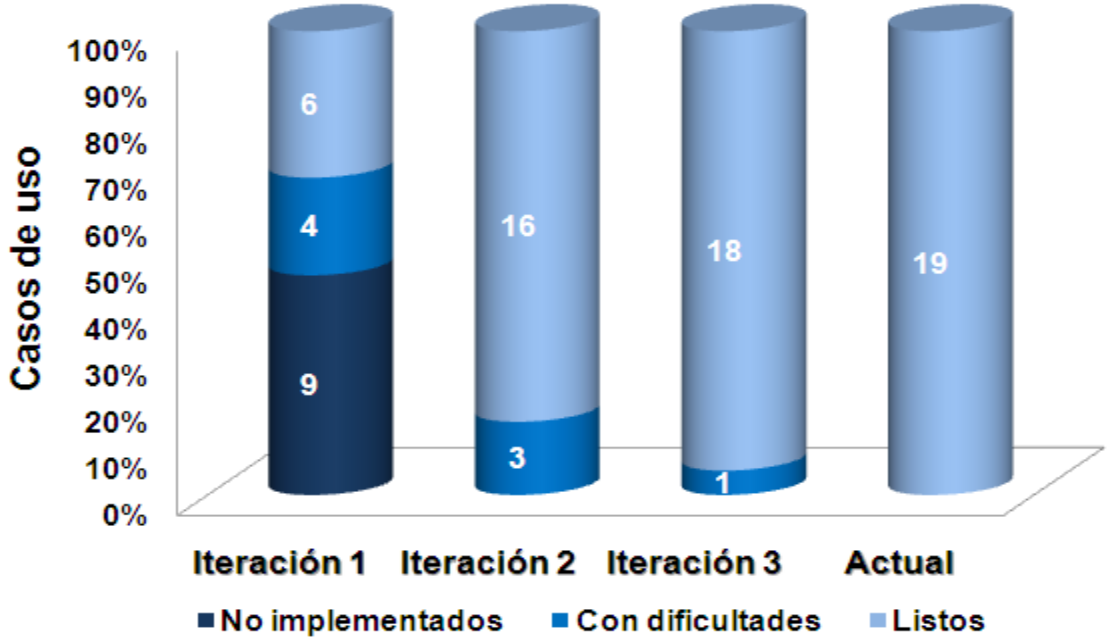


Figura 4.21: Comportamiento del desarrollo de los casos de uso (Fuente: elaboración propia)

Después de realizadas todas las pruebas al sistema se puede indicar que la gestión de los procesos de reportes, la gestión de los procesos de administración y la comunicación de los clientes con el proveedor de servicio SAIME, se clasifican de bien, pues se le dieron respuesta a todas las no conformidades encontradas durante las iteraciones de pruebas, tuvo una complejidad media y se realizó un correcto control de todas las actividades, aplicando pruebas reiteradamente y comprobando que se realizaban todas las validaciones en el sistema, prestando seguridad al mismo. Se puede concluir que la creación del *software* es factible, a pesar de que el esfuerzo fue alto y su tiempo de desarrollo fue de 7 meses.

4.8 Conclusiones

La realización de las pruebas a lo largo del ciclo de vida del desarrollo del *software* permitió identificar los errores cometidos y realizar una retroalimentación de los mismos, corrigiéndolos y evitando posibles fallos. La aplicación de pruebas funcionales en los módulos administración, clientes y reportes fue de gran importancia, pues se realizó una interacción directa con la interfaz, probando el correcto funcionamiento de los casos de

uso y demostrando que los módulos cumplen con todos los requisitos especificados por el cliente, presentando la calidad requerida para su puesta en práctica.

Conclusiones generales

Con la realización de un análisis profundo del proceso de recaudación de pagos de SAIME, se logró una mejor comprensión del funcionamiento de los procesos de administración y reportes, así como de la interacción con los clientes. Esto permitió mejorar considerablemente el flujo de actividades de los mismos.

Al realizar un estudio de sistemas que utilizan entornos web para la gestión de información y creación de reportes, surgieron varias soluciones. No se pusieron en práctica ninguno de los sistemas que arrojó la investigación, algunos pertenecen a entidades propietarias necesitando de una inversión inicial para poder utilizarlos. También la versión gratuita de WebSuitePro que existe está bastante limitada y SiGestCTC no puede realizar las actividades específicas de los módulos.

La propuesta de utilizar el *framework* Symfony, PostgreSQL, Doctrine y NetBeans como herramientas para la creación de la solución, agilizó el proceso de desarrollo. Éstas simplificaron la creación de los módulos y el acceso a los datos, permitiendo al mismo tiempo la obtención de una aplicación robusta.

Después de aplicar la metodología de desarrollo RUP y el lenguaje de modelado UML para realizar el modelado del sistema, se obtuvo una descripción detallada de los procesos, esto permitió tener una mejor comprensión de los requisitos especificados por el cliente, así como la interpretación correcta de las necesidades del sistema. Se logró un mejor entendimiento de las funcionalidades que se utilizaron e implementaron en los módulos.

La automatización de los procesos que se modelaron en los primeros flujos de RUP, es la solución a algunos problemas presentes en las actividades de recaudación de pagos del SAIME, ya que brinda la forma de saber las actividades realizadas en el día y facilita la administración de los datos de la empresa. La implementación del módulo cliente puede mejorar el acceso a los servicios que ofrece SAIME, permitiendo una comunicación activa entre el usuario y la información, eliminando las barreras geográficas.

La aplicación de pruebas funcionales en los módulos y la validación de las variables, demostró el cumplimiento de los requisitos especificados por el cliente, la obtención de la calidad requerida para su puesta en práctica y la factibilidad de su creación.

Recomendaciones

Luego de finalizado el presente trabajo, aunque se han cumplido los objetivos trazados inicialmente, se debe tener en cuenta, que los módulos administración, clientes y reportes desarrollados, pueden ser perfeccionados y actualizados en un futuro para adecuarse a los cambios que ocurran en SAIME, empresa para la cual se ha creado.

Para lograr una mejora en el proceso de recaudación de pagos de SAIME, se recomienda:

- Utilizar una pasarela de pagos para que los clientes puedan realizar sus pagos de forma rápida y segura sin tener que ir al banco.
- Agregar nuevas funcionalidades a la aplicación teniendo en cuenta las opiniones de los clientes.

Referencias bibliográficas

1. SOMMERVILLE, I. Ingeniería del software. 2005. p.
2. SAFFADY, W. Informática documental para bibliotecas., 1986. p.
3. POTENCIER, F. La guía definitiva de Symfony. p.
4. LETELIER, P. L., PENADÉ; CARMEN, MARÍA; CANÓS, JOSÉ. Metodologías Agiles en el Desarrollo de Software., p.
5. LEDESMA, J. D. F. Sistemas organizacionales. Teoría y práctica. p.
6. LARMAN, C. UML y Patrones, introducción al análisis y diseño orientado a objeto., p.
7. GRANOLLERS, T. L., JESÚS. Diseño de sistemas interactivos centrados en el usuario., 2005. p.
8. FALGUERAS, B. C. Ingeniería del software. 2003. p.
9. DEITEL, H. M. D., PAUL J. Cómo programar en C++. 2003. p.
10. DEITE, H. D., PAUL J. Cómo programar en Java. 2004. p.
11. DEBRAUWER, L. V. D. H., FIEN. UML 2: iniciación, ejemplos y ejercicios corregidos. 2005. p.
12. COBO, Á. G., PATRICIA. 2005. PHP y MySQL. Tecnología para el desarrollo de aplicaciones web. p.
13. CABALLERO, J. M. Sistemas Operativos en Entornos monousuario y multiusuario., 2003. p.
14. BROCHARD, J. Internet information services 6., 2006. p.
15. BELL, D. Java para estudiantes. 2003. p.
16. ALONSO AMO, F. M. N., LOÏC. Introducción a la ingeniería del software. 2005. p.
17. http://www.opengestion.com/. Octubre 2010]. Disponible en: http://www.opengestion.com/
18. Libro Jobeet. Noviembre 2010]. Disponible en: http://www.librosweb.es/jobeet/
19. SiGestCTC. Enero 2011]. Disponible en: http://www.ilustrados.com/publicaciones/EkppppukAFiOMMmFyM.php

Bibliografía

1. ALONSO AMO, F. M. N., LOÏC. Introducción a la ingeniería del software. 2005. p.
2. BELL, D. Java para estudiantes. 2003. p.
3. BROCHARD, J. Internet information services 6., 2006. p.
4. CABALLERO, J. M. Sistemas Operativos en Entornos monousuario y multiusuario., 2003. p.
5. COBO, Á. G., PATRICIA. 2005. PHP y MySQL. Tecnología para el desarrollo de aplicaciones web. p.
6. DEBRAUWER, L. V. D. H., FIEN. UML 2: iniciación, ejemplos y ejercicios corregidos. 2005. p.
7. DEITE, H. D., PAUL J. Cómo programar en Java. 2004. p.
8. DEITEL, H. M. D., PAUL J. Cómo programar en C++. 2003. p.
9. Doctrine. Noviembre del 2010]. Disponible en: http://www.doctrine-project.org/
10. EBAY. Paypal. Disponible en: http://www.paypal.es/
11. FALGUERAS, B. C. Ingeniería del software. 2003. p.
12. FRANCOIS ZANINOTTO, F. P. La guía definitiva de Symfony, 2007.
13. GRANOLLERS, T. L., JESÚS. Diseño de sistemas interactivos centrados en el usuario., 2005. p.
14. http://www.opengestion.com/. Octubre 2010]. Disponible en: http://www.opengestion.com/
15. LANCKER, L. V. XHTML 1 y CSS 1 y 2.1: los nuevos estándares de la web 2.0. 2007. p.
16. LARMAN, C. UML y Patrones, introducción al análisis y diseño orientado a objeto., p.
17. LEDESMA, J. D. F. Sistemas organizacionales. Teoría y práctica. p.
18. LETELIER, P. L., PENADÉ; CARMEN, MARÍA; CANÓS, JOSÉ. Metodologías Agiles en el Desarrollo de Software., p.
19. Libro Jobeet. Noviembre 2010]. Disponible en: http://www.librosweb.es/jobeet/
20. MORA, S. L. Programación de aplicaciones web: historia, principios básicos y clientes web. 2002. p.
21. PHP. Febrero del 2011]. Disponible en: http://www.dirphp.com/
22. POTENCIER, F. La guía definitiva de Symfony. p.

GLOSARIO DE TÉRMINOS

23. PRO, W. S. Web Suite Pro, Noviembre 2010]. Disponible en: https://websuitepro.com/
24. S.L, O. G. Gestión empresarial online, 2011. [Disponible en: http://www.mygestion.com/
25. SAFFADY, W. Informática documental para bibliotecas., 1986. p.
26. SAREN. SAREN 2010. [Disponible en: http://www.saren.gob.ve
27. SiGestCTC. Enero 2011]. Disponible en: http://www.ilustrados.com/publicaciones/EkppppukAFiOMMmFyM.php
28. SOMMERVILLE, I. Ingeniería del software. 2005. p.
29. Symfony. Diciembre 2010]. Disponible en: http://www.symfony.es/
30. TCPDF. Marzo 2011]. Disponible en: http://www.tcpdf.org/
31. Web Estilo. Marzo 2011]. Disponible en: http://www.webestilo.com/

Glosario de términos

AJAX: acrónimo de *Asynchronous JavaScript And* XML (JavaScript Asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones hacen posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Apache: servidor web HTTP, de código abierto y multiplataforma.

Aplicación de escritorio: es la aplicación creada para ejecutarse en un ordenador de escritorio sobre un sistema operativo de interfaz visual.

Aplicación web: aplicaciones que los usuarios pueden utilizar accediendo a un servidor a través de Internet o de una intranet mediante un navegador.

Aplicación: programa informático que proporciona servicios de alto nivel al usuario, generalmente utilizando otros programas más básicos que se sitúan por debajo.

CSS: las hojas de estilo en cascada (en inglés: *Cascading Style Sheets*) es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

DOM: Modelo de Objetos del Documento o Modelo en Objetos para la representación de Documentos, es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

Framework: estructura conceptual y tecnológica de soporte definida, normalmente, con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte a programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

HTML: acrónimo inglés de *Hypertext Markup Language (*lenguaje de formato de documentos de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

GLOSARIO DE TÉRMINOS

HTTP (Protocolo de Transferencia de Hipertexto): protocolo usado en las transacciones de la web. Define la sintaxis y la semántica que utilizan los elementos *software* de la arquitectura web (clientes, servidores) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IDE: acrónimo inglés de *Integrated Development Environment* (Entorno de Desarrollo Integrado). Es un programa informático compuesto por un conjunto de herramientas de programación.

jQuery: es una biblioteca o *framework* de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

Multiplataforma: es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de *software*, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en los sistemas operativos Windows, en GNU/GNU/Linux y en MacOS.

Orientado a objetos: hace referencia a la Programación Orientada a Objetos, un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento.

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

Plugins: es un pequeño programa de computadora que extiende las capacidades de otro programa de mayor tamaño, adicionándole nuevas funcionalidades.

PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD *(Berkeley Software Distribution)*, la cual no tiene muchas restricciones.

RAM: denominada así por el acrónimo en inglés del nombre *Random Access Memory*, es un tipo de memora de estado sólido de acceso aleatorio, es decir que nos permite leer y escribir información en ella sin necesidad de un orden estricto.

SFTP: es un protocolo para intercambiar ficheros entre dos máquinas.

TCP/IP: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), se utiliza a nivel mundial para conectarse a Internet y a los servidores web.

URL: significa *Uniform Resource Locator*, en español, localizador uniforme de recurso y se refiere a la dirección única que identifica a una página web en Internet.

XHTML: acrónimo inglés de *Extensible Hyper Text Markup Language* (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

XML: es un lenguaje de marcado extensible (en inglés: *eXtensible Markup Language*) que usa etiquetas, desarrollado por el W3C. Permite definir la gramática de lenguajes específicos.

Anexo 1: Requisitos funcionales

RF - 1. Administrar servicios.

RF - 1.1. Crear nuevo servicio

- Nombre del servicio
- Tipo de pago
- Cantidad de unidades del tipo de pago
- Descripción del servicio
- Tipo de servicio
- Estado
- Proveedor
- Guardar los datos del servicio

RF - 1.2. Modificar servicios.

RF - 1.2.1. Listar los servicios.

RF - 1.2.2. Filtrar los servicios por:

- Nombre
- Estado
- Proveedor
- Creado

RF - 1.2.3. Seleccionar el servicio que desea modificar.

RF - 1.2.4. Mostrar los campos del servicio para ser modificados.

RF - 1.2.4.1. Notificar a los clientes afectados por tener el servicio solicitado en caso de que ocurran cambios en el precio de dicho servicio.

RF - 1.2.4.2. Renombrar el servicio en todas las solicitudes activas que contengan dicho servicio, en caso de que el cambio sea para el nombre.

RF - 1.2.5. Validar los cambios efectuados.

RF - 1.2.6. Guardar los nuevos datos del servicio.

RF - 1.3. Habilitar servicios.

RF - 1.3.1. Listar los servicios.

RF - 1.3.1.1. Filtrar los servicios por:

- Todos los servicios
- Servicios habilitados

- Servicios deshabilitados

RF - 1.3.2. Marcar opción de habilitación.

RF - 1.3.3. Guardar preferencias.

RF - 1.4. Deshabilitar servicios.

RF - 1.4.1. Listar los servicios.

RF - 1.4.1.1. Filtrar los servicios por:

- Todos los servicios
- Servicios habilitados
- Servicios deshabilitados

RF - 1.4.2. Marcar opción de deshabilitación.

RF - 1.4.3. Guardar preferencias.

RF - 1.5. Eliminar servicios existentes.

RF - 1.5.1. Listar los servicios existentes.

RF - 1.5.1.1. Filtrar los servicios por:

- Nombre
- Estado
- Proveedor
- Creado

RF - 1.5.2. Seleccionar varios servicios.

RF - 1.5.3. Eliminar servicios seleccionados.

RF - 1.5.3.1. Verificar que los servicios seleccionados no estén solicitados en ninguna solicitud activa.

RF - 2. Administrar tipos de pago.

RF - 2.1. Crear nuevo tipo de pago.

- Nombre
- Unidad de medida
- Costo de la unidad

RF - 2.1.1. Guardar el tipo de pago creado.

RF - 2.2. Modificar tipo de pago.

RF - 2.2.1. Listar los tipos de pago existentes.

RF - 2.2.2. Seleccionar un tipo de pago.

RF - 2.2.3. Guardar los datos del tipo de pago modificados.

RF - 2.2.3.1. Validar el formato de los datos entrados.

RF - 2.2.4. Verificar que el tipo de pago esté siendo utilizado en solicitudes activas.

RF - 2.2.5. Modificar los tipos de pago seleccionados.

RF - 2.2.6. Notificar a los clientes cuya solicitud aumentó de precio.

RF - 2.3. Eliminar tipo de pago.

RF - 2.3.1. Listar los tipos de pago existentes.

RF - 2.3.2. Seleccionar varios tipos de pago.

RF - 2.3.3. Verificar que el tipo de pago esté siendo utilizado en solicitudes activas.

RF - 2.3.3.1. Eliminar el tipo de pago teniendo en cuenta:

- Mantener el monto total de la solicitud activa que lo utilizaba.
- Mantener las unidades de medida que describen los servicios de la solicitud.

RF - 2.3.4. Eliminar los tipos de pago seleccionados.

RF - 3. Administrar clientes.

RF - 3.1. Registrar nuevo cliente.

RF - 3.1.1. Validar el formato de los datos entrados.

RF - 3.1.2. Almacenar los datos de cliente nuevo:

- Nombre de usuario
- Contraseña
- Repetir Contraseña
- Nombre y apellidos
- Cédula
- RIF
- SMS
- Correo electrónico
- Teléfono
- Fax
- Captcha (Prueba de Turing pública y automática para diferenciar máquinas y humanos)

RF - 3.1.3. Capturar contraseñas entradas.

RF - 3.1.3.1. Validar fortaleza de la contraseña.

RF - 3.1.3.2. Verificar que sean iguales las contraseñas.

RF - 3.1.4. Guardar los datos entrados.

RF - 3.1.5. Enviar correo electrónico de verificación.

RF - 3.1.5.1. Generar URL única para el usuario.

RF - 3.1.6. Dar de alta al cliente cuando entre por su URL única.

RF - 3.2. Modificar clientes registrados.

RF - 3.2.1. Validar el formato de los datos entrados por el usuario.

RF - 3.2.2. Guardar los datos.

RF - 3.3 Recuperar contraseña olvidada.

RF – 3.3.1. Mostrar recuperar contraseña.

RF – 3.3.2. Solicitar el correo electrónico del cliente.

RF - 3.3.3. Mostrar Captcha.

RF – 3.3.3.1. Generar nuevo Captcha.

RF - 3.3.4. Entrar Captcha.

RF – 3.3.4.1. Validar Captcha.

RF – 3.3.5. Enviar un correo electrónico con URL única de recuperación.

RF – 3.3.5.1. Generar URL única para recuperación de contraseña.

RF – 3.3.6. Capturar las contraseñas entradas.

RF – 3.3.7. Validar las contraseñas.

RF – 3.3.7.1. Verificar que sean iguales.

RF – 3.3.7.2. Verificar fortaleza.

RF – 3.3.8. Guardar nueva contraseña.

RF – 3.3.9. Enviar correo electrónico de contraseña cambiada.

RF – 3.4 Deshabilitar cliente.

RF - 3.4.1 Listar los clientes registrados en el sistema.

RF - 3.4.1.1 Filtrar el listado de los clientes por:

- Cédula.

RF – 3.4.2 Seleccionar un cliente.

RF – 3.4.3 Alertar de la acción de deshabilitación.

RF – 3.4.3.1 Notificar al cliente.

RF – 3.4.3.2 Registrar un nuevo aviso.

RF – 3.4.3.3 Seleccionar vía del aviso (correo electrónico, SMS).

RF – 3.4.3.4. Registrar motivo de deshabilitación.

RF – 3.4.3.5 Enviar aviso al cliente.

RF – 3.4.4 Deshabilitar cliente.

RF – 3.5. Eliminar clientes.

RF – 3.5.1. Listar los clientes registrados en el sistema.

RF – 3.5.1.1. Filtrar el listado de los clientes por:

- Cédula

RF – 3.5.2. Seleccionar un cliente.

RF – 3.5.3. Verificar que el cliente a eliminar no tenga servicios pagados y no liquidados.

RF – 3.5.4. Eliminar el cliente seleccionado.

RF – 3.5.4.1. Alertar de la acción.

RF – 3.6 Habilitar cliente.

RF - 3.6.1 Listar los clientes registrados en el sistema.

RF - 3.6.1.1Filtrar el listado de los clientes por:

- Cédula.

RF – 3.6.2 Seleccionar un cliente.

RF – 3.6.3 Habilitar cliente.

RF - 4. Administrar bancos recaudadores.

RF - 4.1. Crear un nuevo banco recaudador.

RF - 4.1.1. Almacenar los datos del nuevo banco recaudador:

- Código del banco asignado por el SIB
- Nombre del banco
- RIF
- Dirección postal de la casa matriz del banco
- Nombre y apellido del contacto
- Cédula de identidad del contacto
- Correo electrónico
- Número de teléfono
- Fax
- Si presta servicio de punto de venta

RF - 4.1.1.1. Validar el formato de los datos entrados por el usuario.

RF - 4.1.1.2. Guardar los datos.

RF - 4.2. Modificar un banco recaudador.

RF - 4.2.1. Mostrar una lista con los bancos recaudadores.

RF - 4.2.2. Seleccionar un banco recaudador.

RF - 4.2.3. Validar el formato de los datos entrados por el usuario.

RF - 4.2.4. Guardar los datos.

RF - 4.3. Eliminar banco recaudador.

RF - 4.3.1. Mostrar una lista con los bancos recaudadores.

RF - 4.3.2. Seleccionar un banco recaudador.

RF - 4.3.3. Eliminar el banco recaudador seleccionado.

RF - 4.3.3.1. Verificar que el banco recaudador no tiene verificaciones de pago pendiente.

RF - 4.3.3.2. Mostrar alerta de eliminación.

RF - 4.3.3.3. Marcar como eliminado.

RF - 5. Administrar cuentas recaudadoras en bancos.

RF - 5.1. Crear nueva cuenta recaudadora.

RF - 5.1.1. Almacenar los datos de una nueva cuenta recaudadora:

- Código
- Banco
- Proveedor
- Estado (corriente o ahorro)
- Monto

RF - 5.1.2. Validar los datos entrados.

RF - 5.1.3. Guardar los datos.

RF - 5.2. Modificar una cuenta recaudadora existente.

RF - 5.2.1. Mostrar las cuentas recaudadoras agrupadas en los bancos.

RF - 5.2.2. Seleccionar una cuenta recaudadora para modificarla.

RF - 5.2.3. Validar los datos modificados.

RF - 5.2.4. Guardar los cambios efectuados.

RF - 5.3. Eliminar cuenta recaudadora.

RF - 5.3.1. Mostrar una lista con las cuentas recaudadoras por bancos.

RF - 5.3.2. Seleccionar una cuenta recaudadora.

RF - 5.3.3. Eliminar la cuenta recaudadora seleccionada.

RF - 5.3.3.1. Verificar que la cuenta no tiene depósitos de servicios sin liquidar en el sistema.

RF - 5.3.3.2. Mostrar alerta de eliminación de cuenta.

RF - 5.3.3.3. Marcar como eliminada la cuenta.

RF - 6. Administrar proveedor.

RF - 2.1. Crear nuevo proveedor.

- Nombre
- Descripción

RF - 2.1.1. Guardar el proveedor creado.

RF - 2.2. Modificar proveedor.

RF - 2.2.1. Listar los proveedores existentes.

RF - 2.2.2. Seleccionar un proveedor.

RF - 2.2.3. Guardar los datos del proveedor modificados.

RF - 2.2.3.1. Validar el formato de los datos entrados.

RF - 2.2.5. Modificar los tipos de pago seleccionados.

RF - 2.3. Eliminar proveedor.

RF - 2.3.1. Listar los proveedores existentes.

RF - 2.3.2. Seleccionar varios proveedores.

RF - 2.3.4. Eliminar los tipos de pago seleccionados.

RF - 7. Gestionar mensajes de correo electrónico.

RF - 7.1. Mostrar las categorías básicas de mensajes vía correo electrónico:

- Cambio de contraseña
- Nuevo usuario
- Mensaje a banco para mecanismo de pago en línea
- Error de envío de avisos automáticos

RF - 7.1.1. Crear categorías adicionales.

RF - 7.1.1.1. Guardar datos entrados.

RF - 7.1.2. Nombre de la categoría.

RF - 7.1.3. Descripción de la categoría.

RF - 7.1.4. Eliminar categorías adicionales.

RF - 7.1.4.1. Listar categorías adicionales.

RF - 7.1.4.2. Seleccionar varias categorías.

RF - 7.1.4.3. Eliminar categorías.

RF - 7.1.4.4. Verificar que no haya mensajes creados para esas categorías.

RF - 7.2. Seleccionar la categoría.

RF - 7.3. Guardar los datos del mensaje.

- Categoría
- Asunto
- Cuerpo del mensaje

RF - 7.4. Eliminar mensajes existentes.

RF - 7.4.1. Mostrar los mensajes creados.

RF - 7.4.1.1. Filtrar por categorías.

RF - 7.4.2. Seleccionar los mensajes a eliminar.

RF - 7.4.3. Eliminar mensajes.

RF - 7.4.3.1. Alertar de la necesidad de que las categorías básicas tengan al menos un mensaje.

RF - 7.5. Modificar mensajes.

RF - 7.5.1. Mostrar los mensajes creados.

RF - 7.5.1.1. Filtrar por categorías.

RF - 7.5.2. Seleccionar un mensaje a modificar.

RF - 7.5.3. Guardar datos entrados.

RF - 7.5.3.1. Alertar de la necesidad de que las categorías básicas tengan al menos un mensaje.

RF - 8. Generar reporte diario de servicios vendidos por su estado.

RF – 8.1 Listar estados de servicios existentes

RF – 8.2 Seleccionar criterio de reporte por estado de servicios.

RF – 8.3 Mostrar una tabla con los servicios vendidos de ese estado.

RF – 8.4. Imprimir reporte.

RF - 8.4.1. Generar pdf con reporte.

RF - 9. Generar reporte de pagos diarios por servicios y solicitudes liquidadas.

RF - 9.1. Generar el reporte en un pdf.

RF - 9.3.1. Mostrar una tabla con la cantidad recaudada (servicios liquidados y solicitudes liquidadas).

RF - 9.3.2. Mostrar el total recaudado.

RF - 10. Generar reporte de pagos diarios por tipo de servicio.

RF – 10.1 Listar tipos de servicios existentes

RF – 10.2 Seleccionar criterio de reporte por tipo de servicios.

RF – 10.3 Mostrar una tabla con los servicios vendidos de ese tipo, su costo y su estado.

RF – 10.4. Imprimir reporte.

RF - 10.4.1. Generar pdf con reporte.

RF - 11. Gestionar preferencias de los clientes.

RF - 11.1. Mostrar un listado de las notificaciones que permite el sistema:

- Notificaciones de liquidación de un servicio pagado por él.
- Notificaciones de expiración de solicitud
- Notificaciones de deshabilitación de usuario

RF - 11.2. Seleccionar las notificaciones que desea recibir.

RF - 11.3. Seleccionar los medios por los que recibir notificaciones.

RF - 11.3.1. Listar los medios de recepción de notificaciones:

- Correo electrónico
- SMS

RF - 11.3.2. Seleccionar los medios deseados.

RF - 11.4. Guardar preferencias de notificaciones del sistema.

RF - 12. Gestionar avisos del proveedor al cliente.

RF - 12.1. Crear lista de distribución.

RF - 12.1.1. Definir nombre a lista de distribución.

RF - 12.1.2. Listar los clientes registrados para el proveedor.

RF - 12.1.3. Seleccionar clientes que se desea añadir a la lista creada.

RF - 12.1.3.1. Seleccionar todos los clientes.

RF - 12.1.3.2. Invertir selección actual.

RF - 12.1.4. Adicionar los clientes seleccionados a la lista.

RF - 12.1.5. Guardar la lista de distribución.

RF - 12.2. Enviar aviso promocional o correo electrónico.

- Fecha
- Destinatarios
- Fecha de envío

RF - 12.3. Enviar manualmente aviso promocional o correo electrónico.

RF - 12.3.1. Listar las posibilidades de destinatario.

RF - 12.3.1.1. Listar los grupos creados en orden alfabético.

RF - 12.3.1.2. Listar los usuarios del cliente en orden alfabético.

RF - 12.3.1.3. Alternar entre lista de cliente y lista de grupos y viceversa.

RF - 12.3.2. Seleccionar destinatario.

RF - 12.3.2.1. Adicionar grupo seleccionado.

RF - 12.3.2.2. Adicionar usuario seleccionado.

RF - 12.3.2.3. Permitir la selección de grupos o clientes específicos para un mismo envío.

RF - 12.3.3. Listar los correos electrónicos y avisos promocionales creados.

RF - 12.3.4. Seleccionar el aviso promocional o correo electrónico que se desea enviar.

RF - 12.3.5. Enviar aviso promocional o correo electrónico.

Anexo 2: Diagramas de clases del análisis

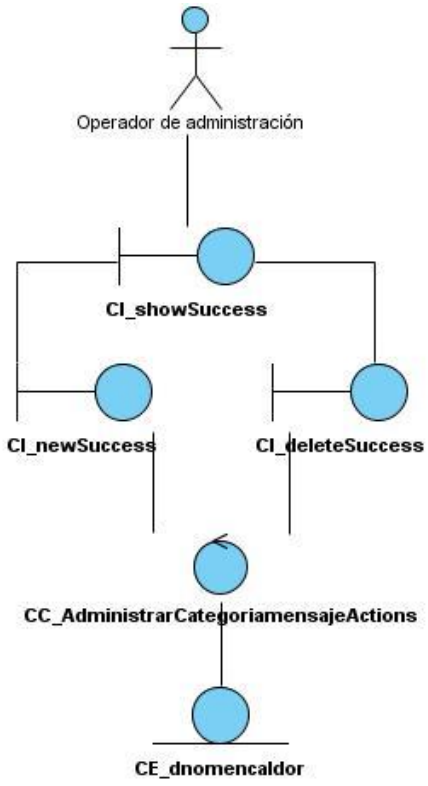


Figura 22: Diagrama de clases del análisis "Administrar categoría" (Fuente: elaboración propia)

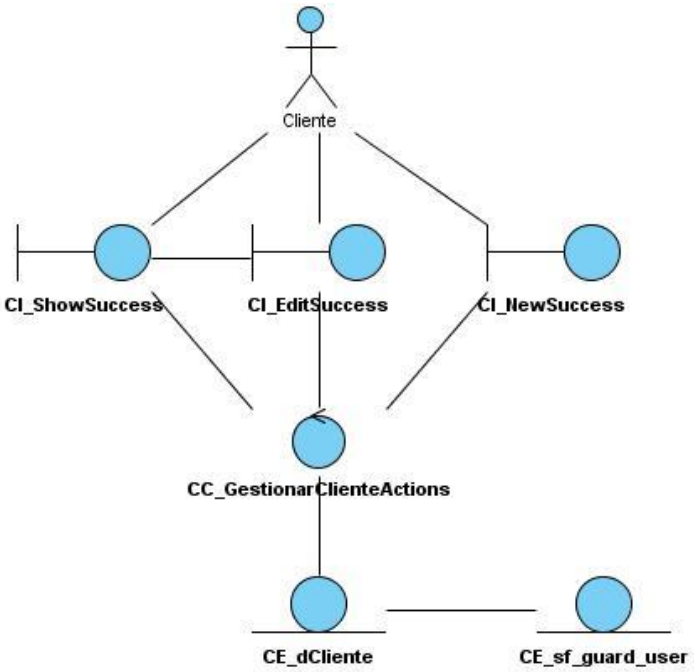


Figura 23: Diagrama de clases del análisis "Administrar cliente" (Fuente: elaboración propia)

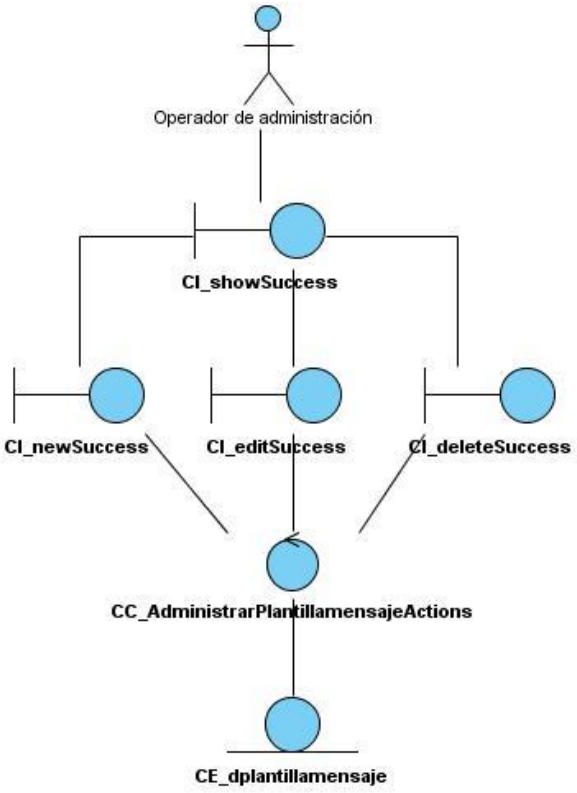


Figura 24: Diagrama de clases del análisis "Administrar mensajes de correo" (Fuente: elaboración propia)

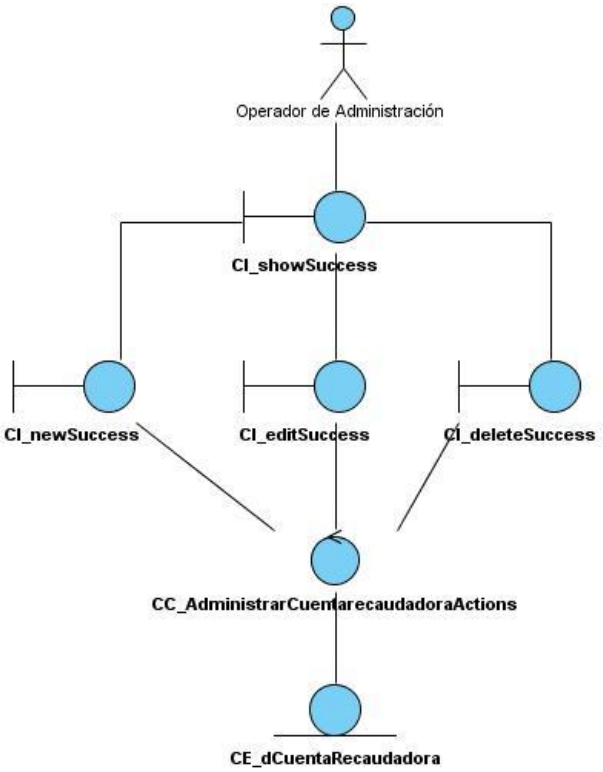


Figura 25: Diagrama de clases del análisis "Administrar cuentas recaudadoras" (Fuente: elaboración propia)

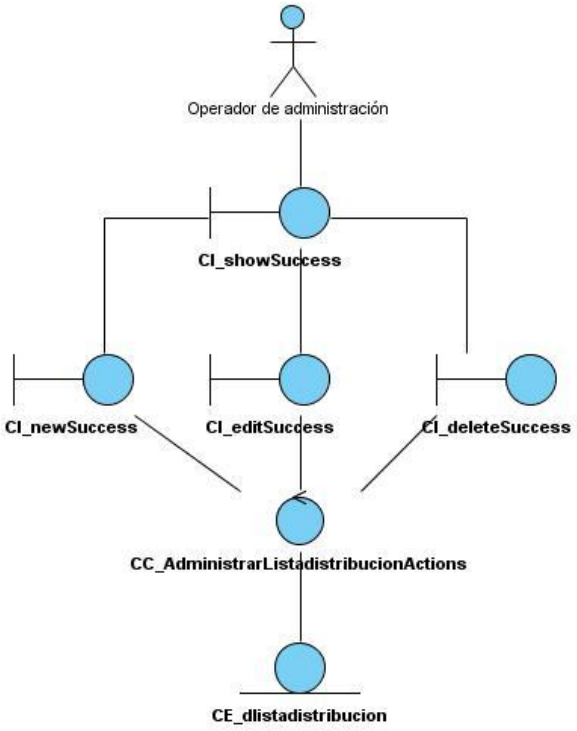


Figura 26: Diagrama de clases del análisis "Administrar lista de distribución" (Fuente: elaboración propia)

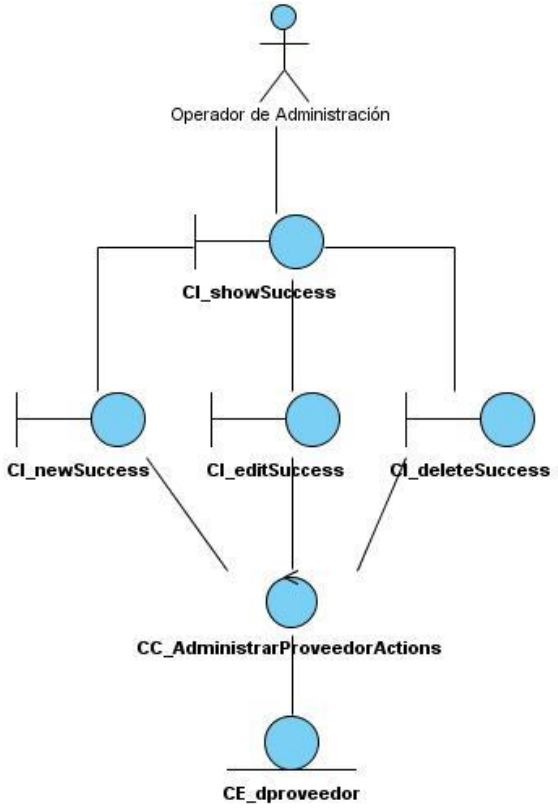


Figura 27: Diagrama de clases del análisis "Administrar proveedor" (Fuente: elaboración propia)

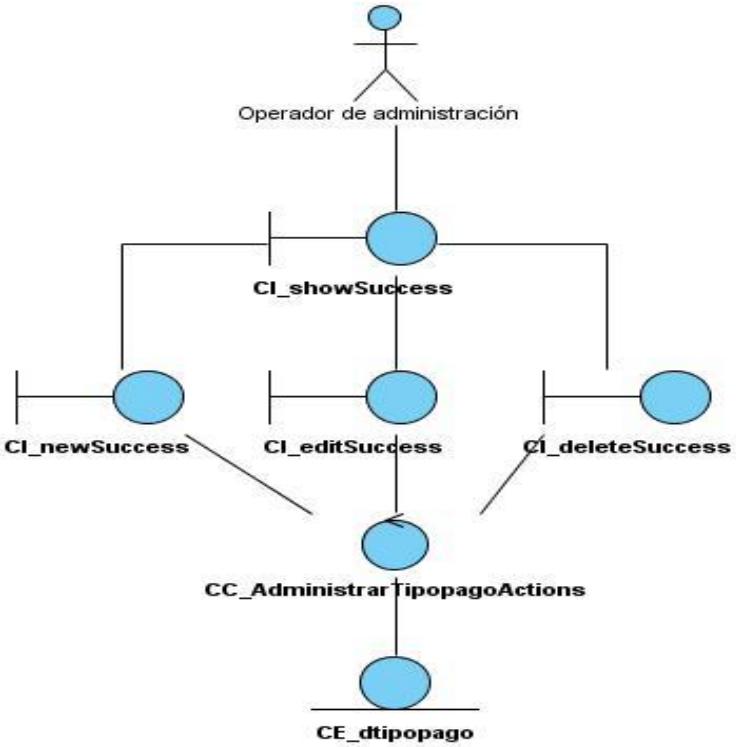


Figura 28: Diagrama de clases del análisis "Administrar tipo de pago" (Fuente: elaboración propia)

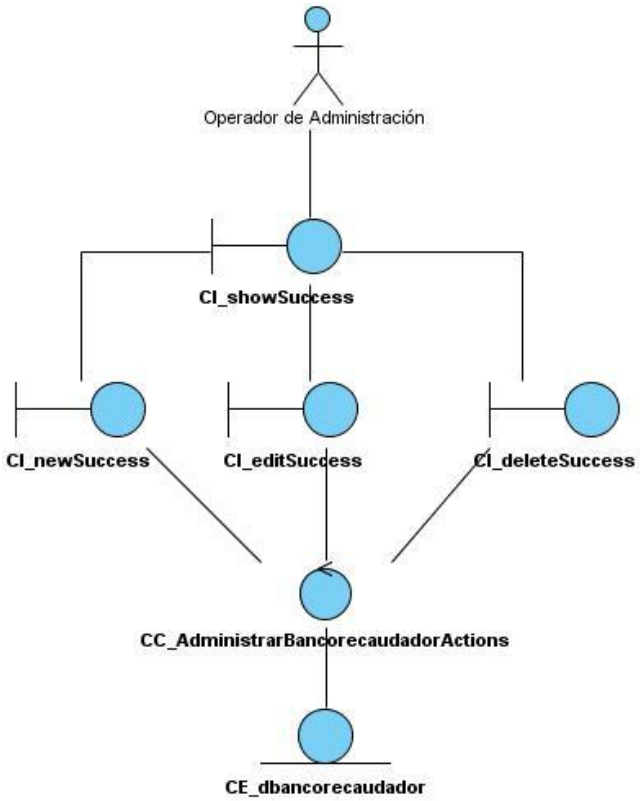


Figura 29: Diagrama de clases del análisis "Administrar banco recaudador" (Fuente: elaboración propia)

Figura 30: Diagrama de clases del análisis "Enviar aviso promocional manual o mensaje de correo" (Fuente: elaboración propia)

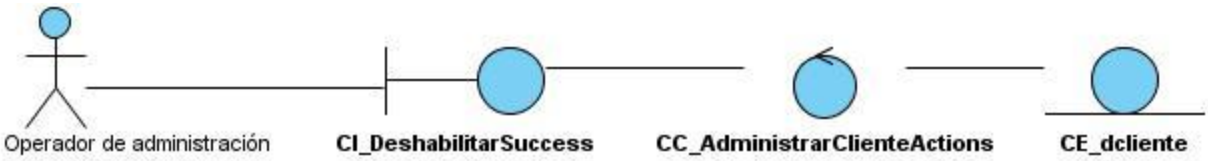


Figura 31: Diagrama de clases del análisis "Deshabilitar cliente" (Fuente: elaboración propia)

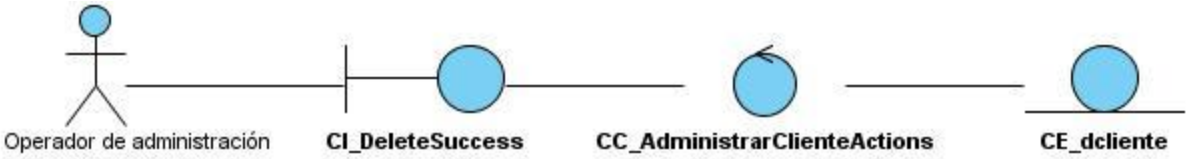


Figura 32: Diagrama de clases del análisis "Eliminar cliente" (Fuente: elaboración propia)

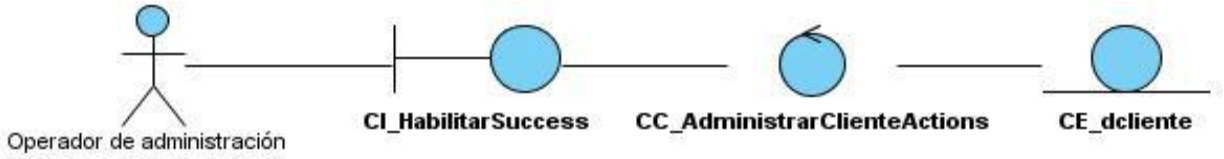


Figura 33: Diagrama de clases del análisis "Habilitar cliente" (Fuente: elaboración propia)

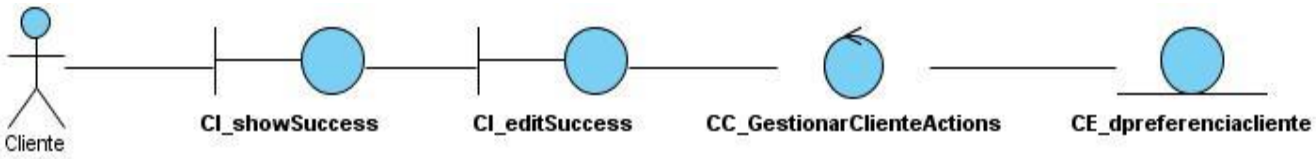


Figura 34: Diagrama de clases del análisis "Modificar preferencias de los clientes" (Fuente: elaboración propia)

Figura 35: Diagrama de clases del análisis "Recuperar contraseña" (Fuente: elaboración propia)

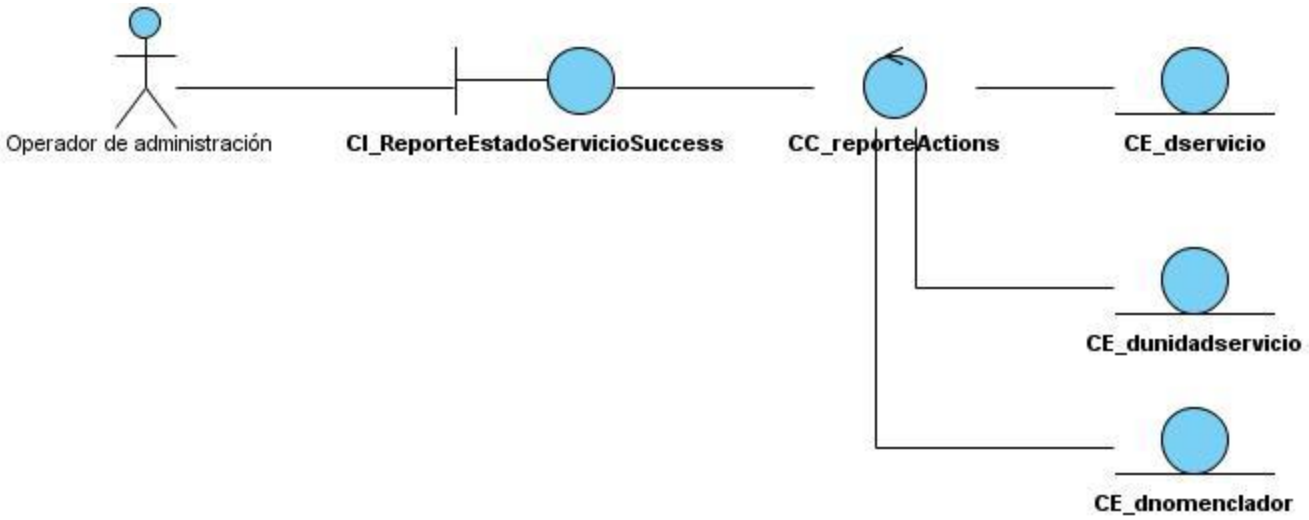


Figura 36: Diagrama de clases del análisis "Generar reporte de servicios por estado" (Fuente: elaboración propia)

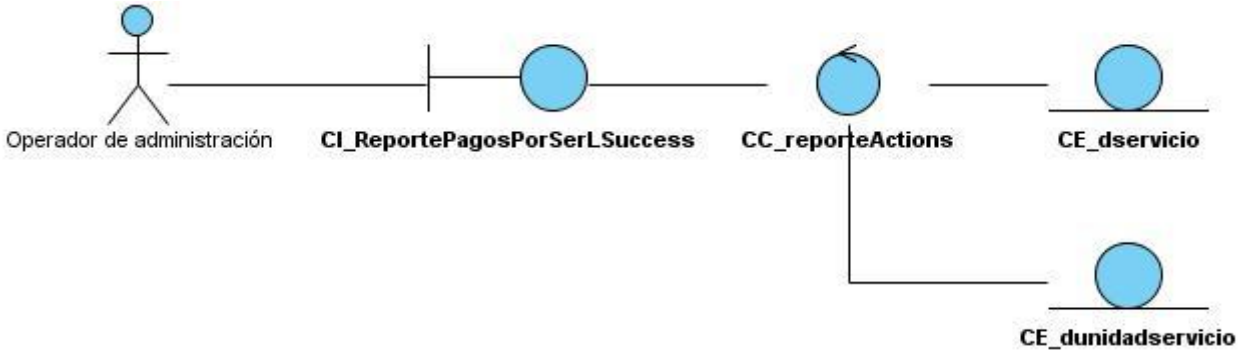


Figura 37: Diagrama de clases del análisis "Generar reporte de pagos diarios por servicios liquidados" (Fuente: elaboración propia)

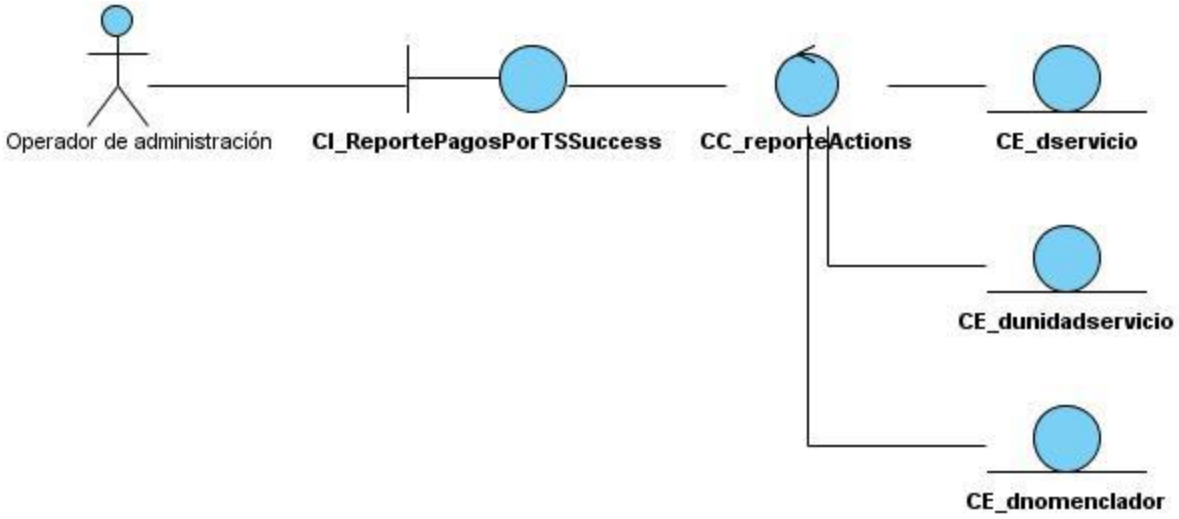


Figura 38: Diagrama de clases del análisis "Generar reporte de pagos de servicios por tipo" (Fuente: elaboración propia)

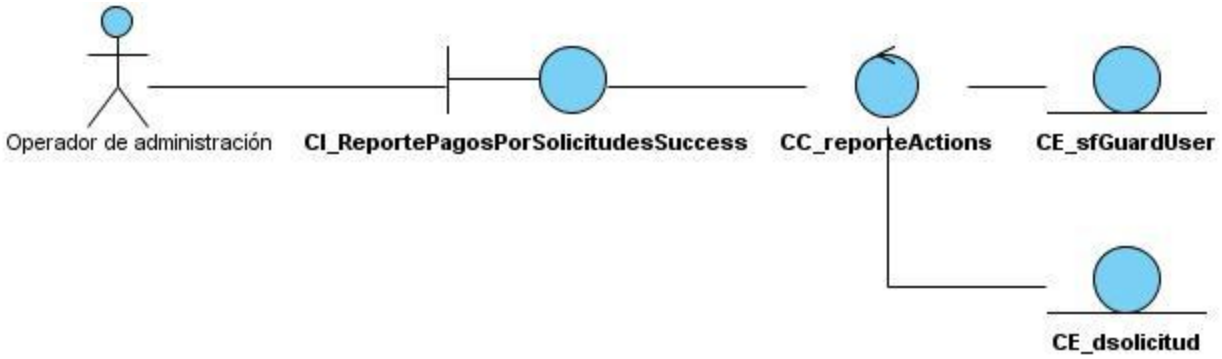


Figura 39: Diagrama de clases del análisis "Generar reporte de pagos diarios por solicitudes liquidadas" (Fuente: elaboración propia)

Anexo 3: Diagramas de clases del diseño

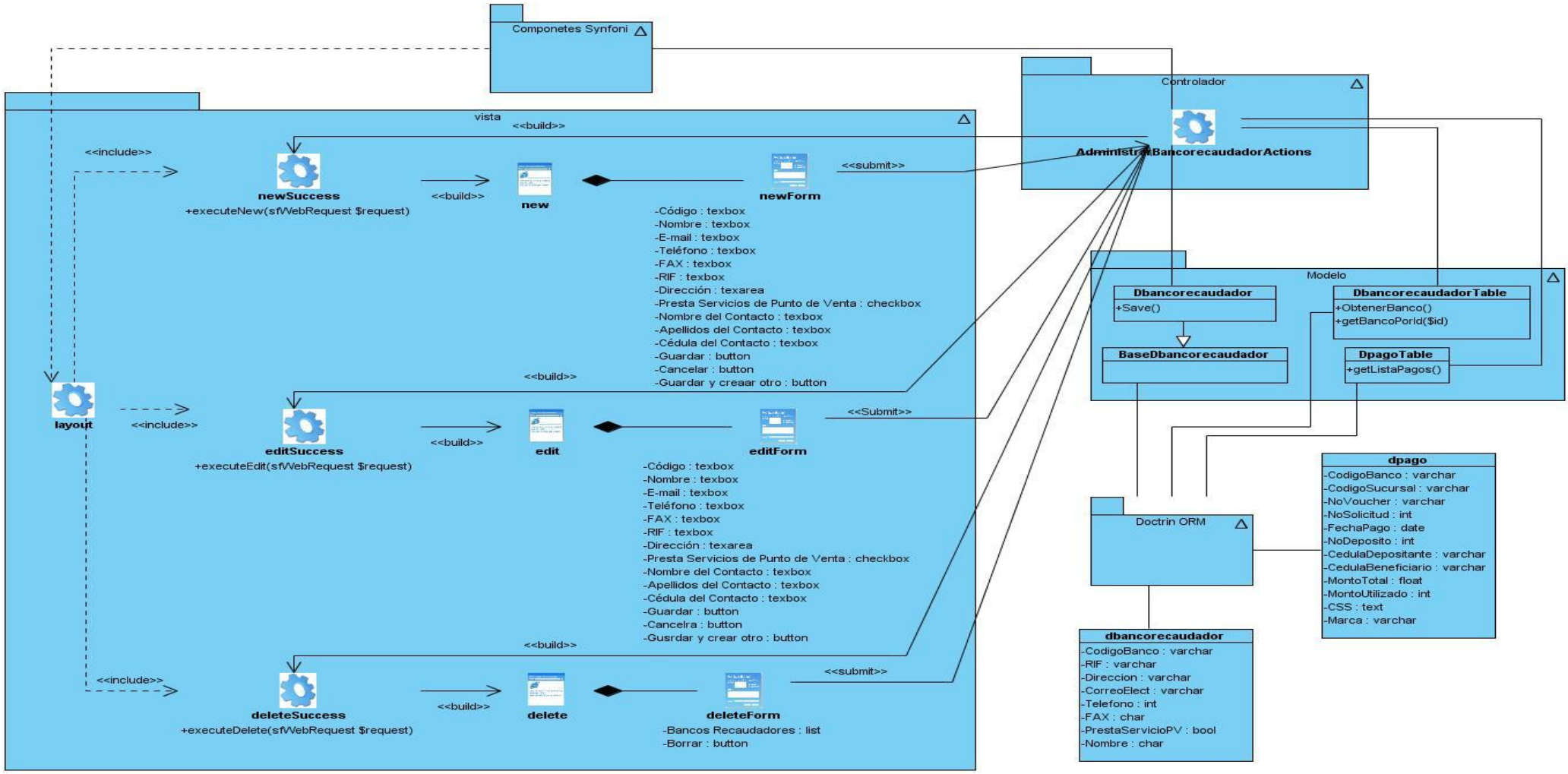


Figura 40: Diagrama de clases del diseño "Administrar banco recaudador" (Fuente: elaboración propia)

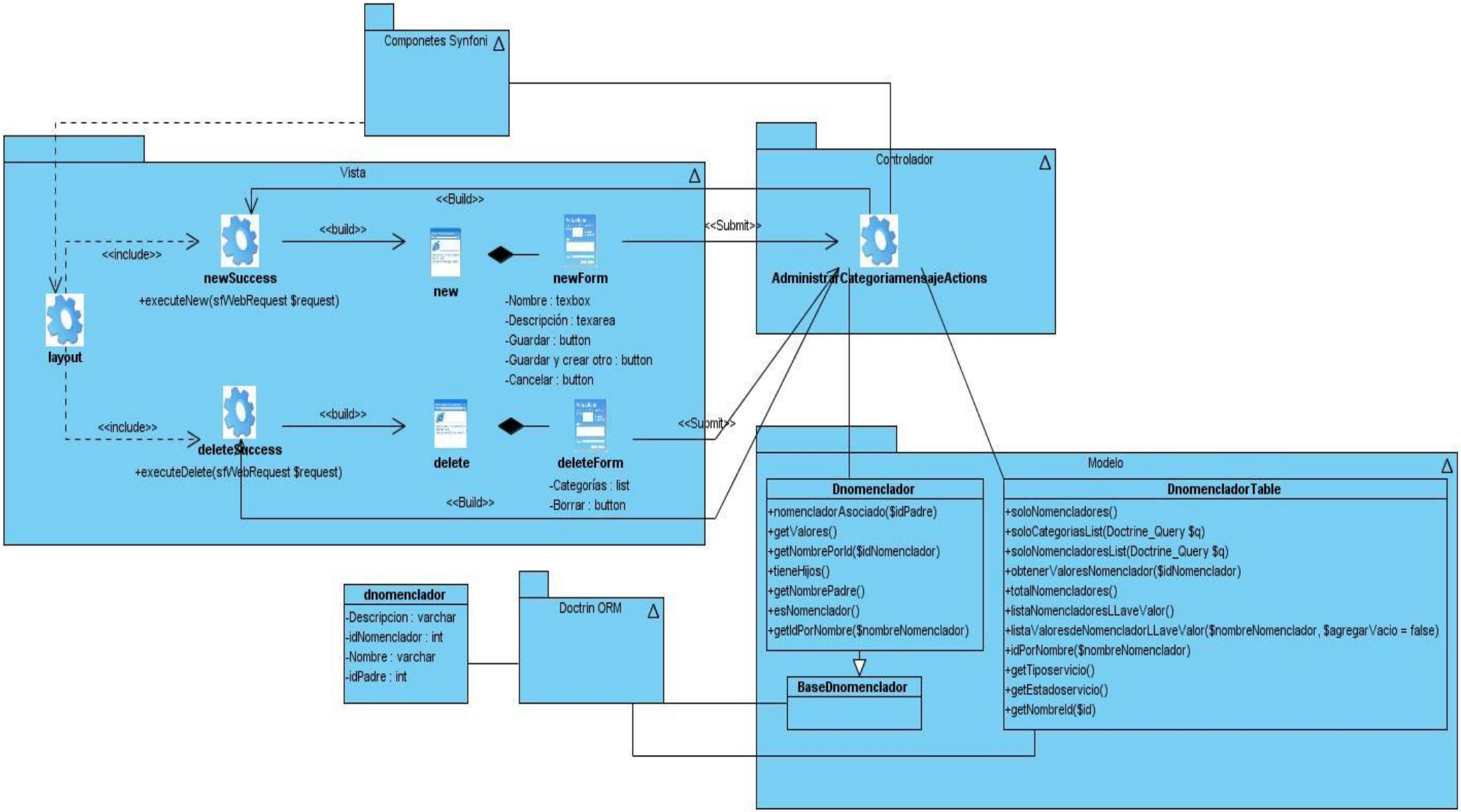


Figura 41: Diagrama de clases del diseño "Administrar categoría" (Fuente: elaboración propia)

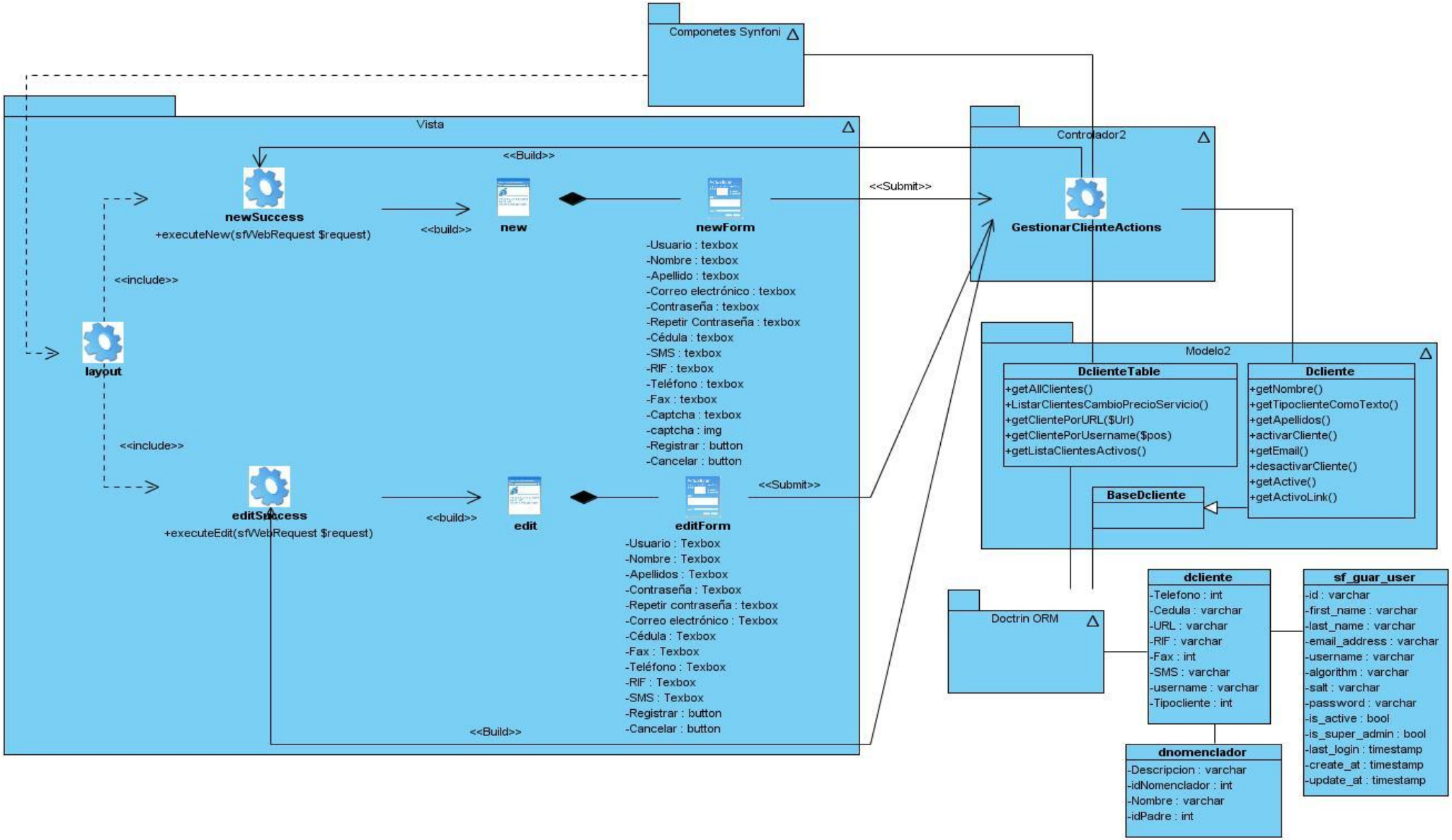


Figura 42: Diagrama de clases del diseño "Administrar cliente" (Fuente: elaboración propia)

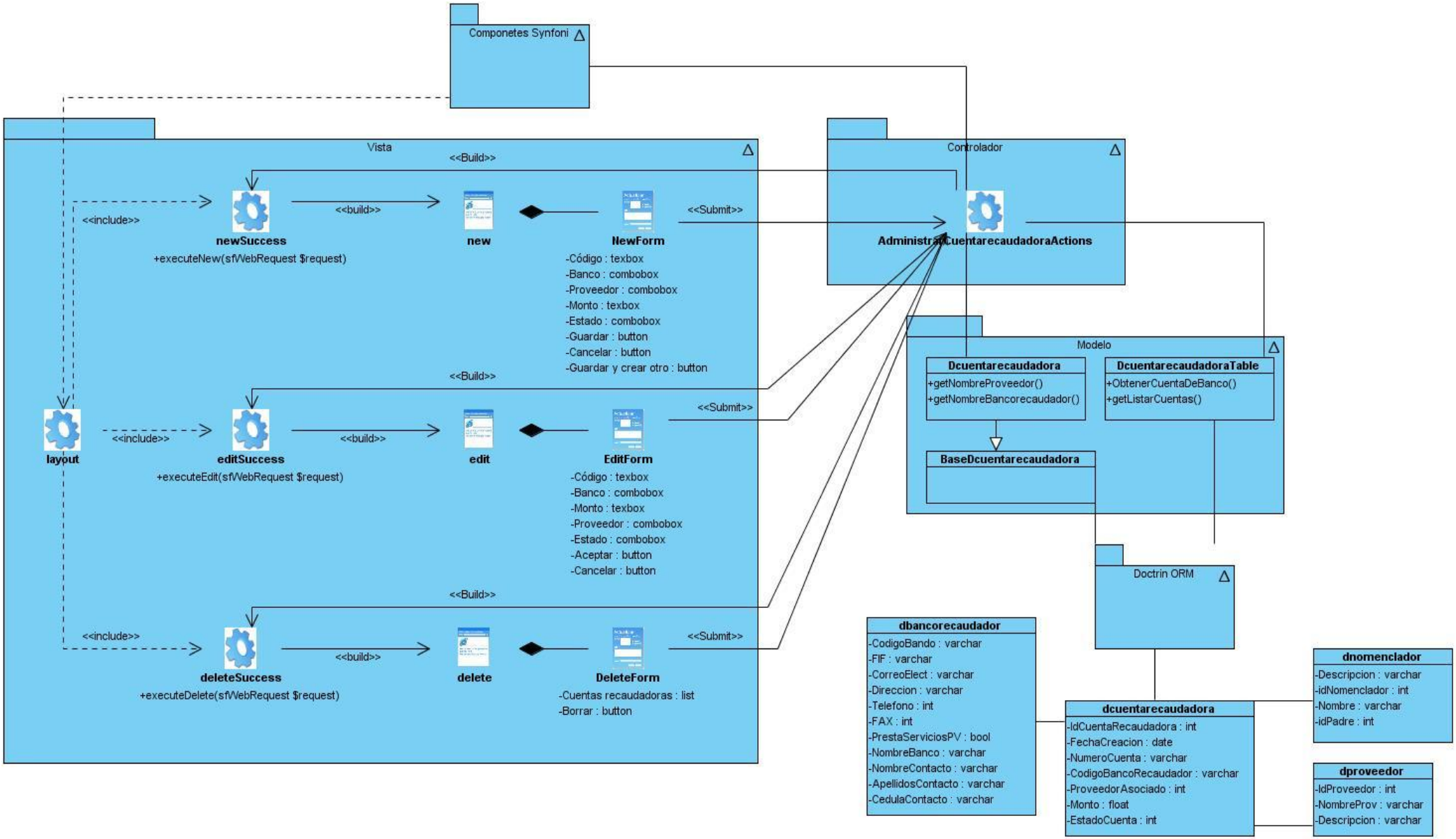


Figura 43: Diagrama de clases del diseño "Administrar cuantas recaudadoras" (Fuente: elaboración propia)

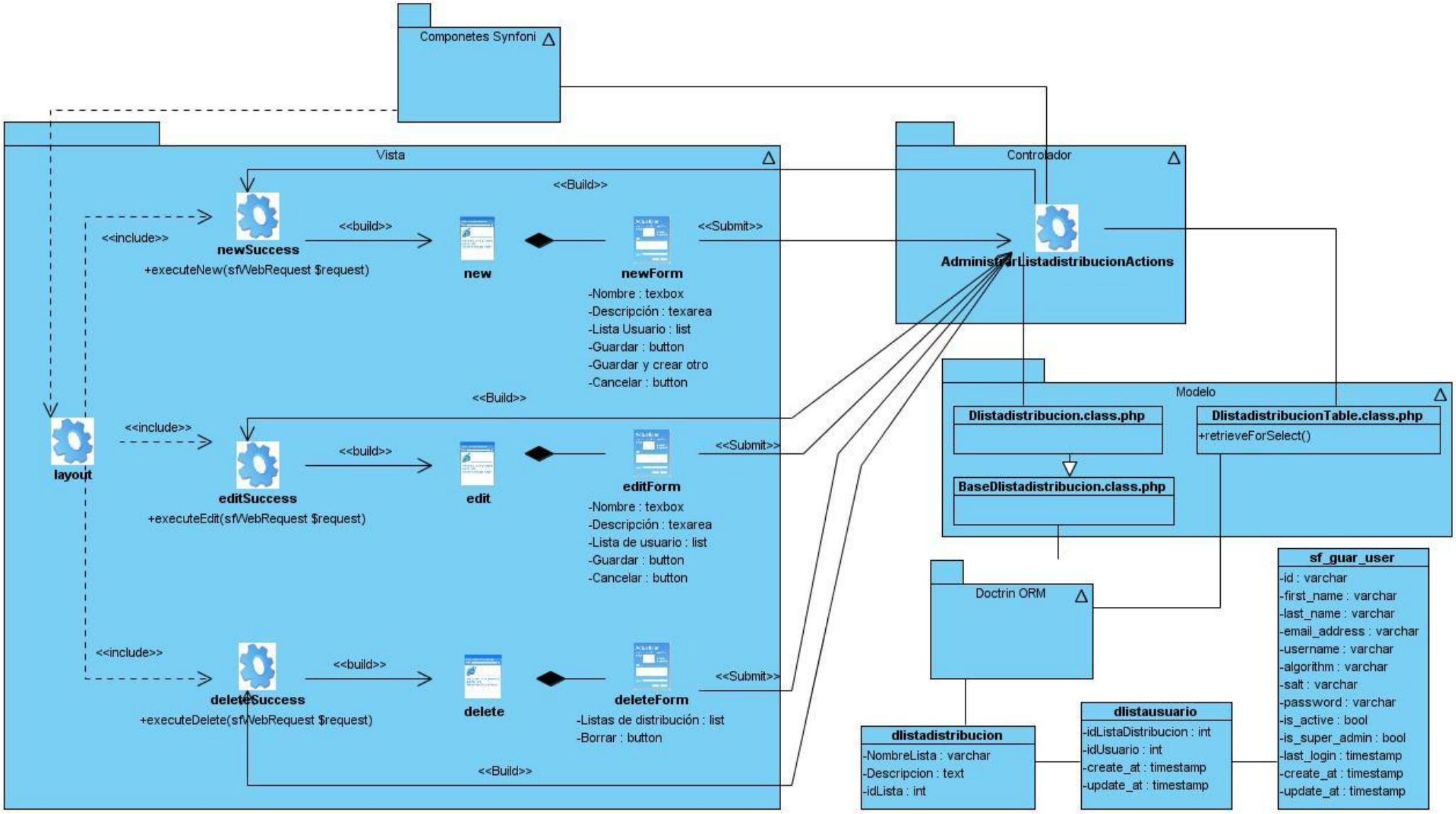


Figura 44: Diagrama de clases del diseño "Administrar lista de distribución" (Fuente: elaboración propia)

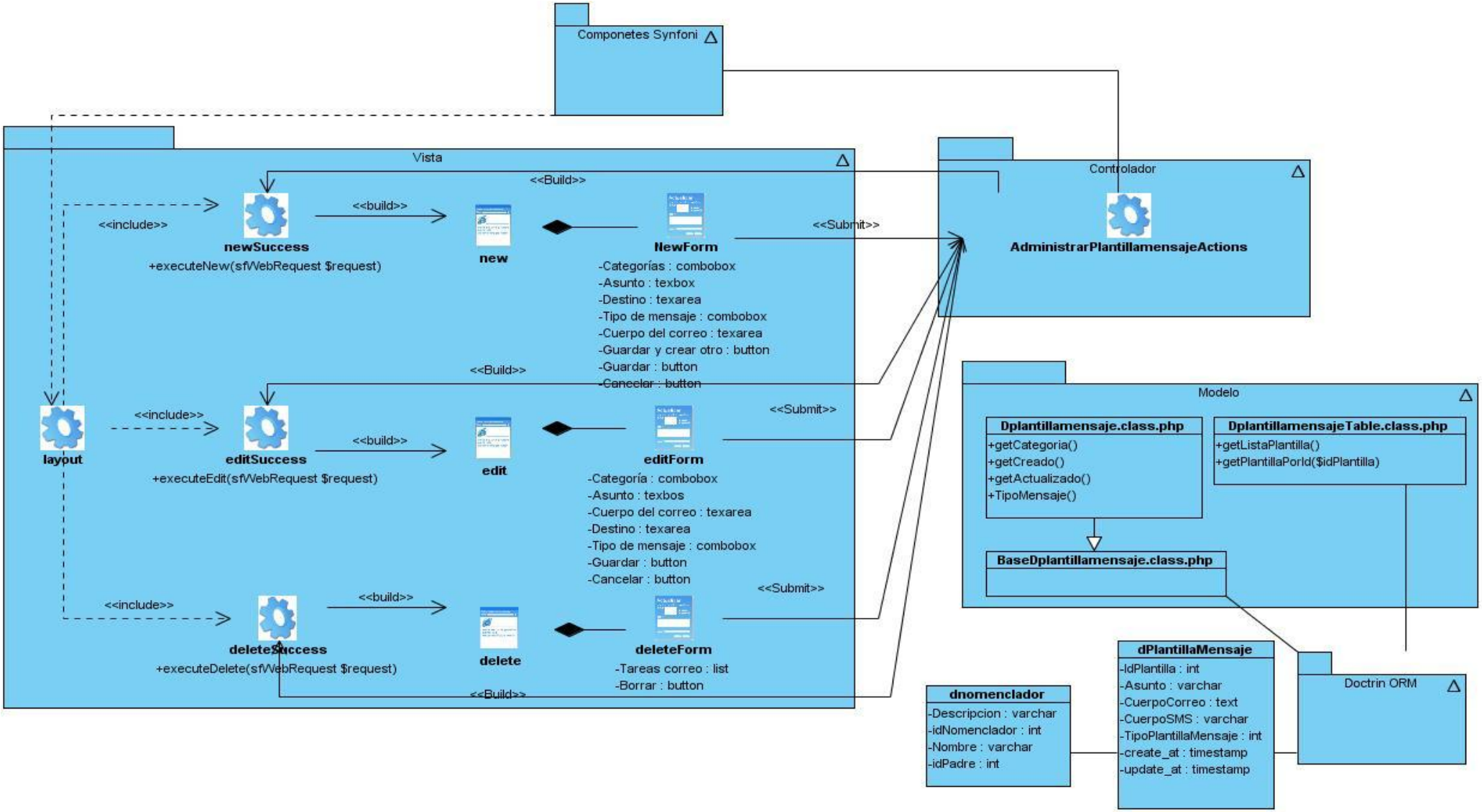


Figura 45: Diagrama de clases del diseño "Administrar mensaje de correo" (Fuente: elaboración propia)

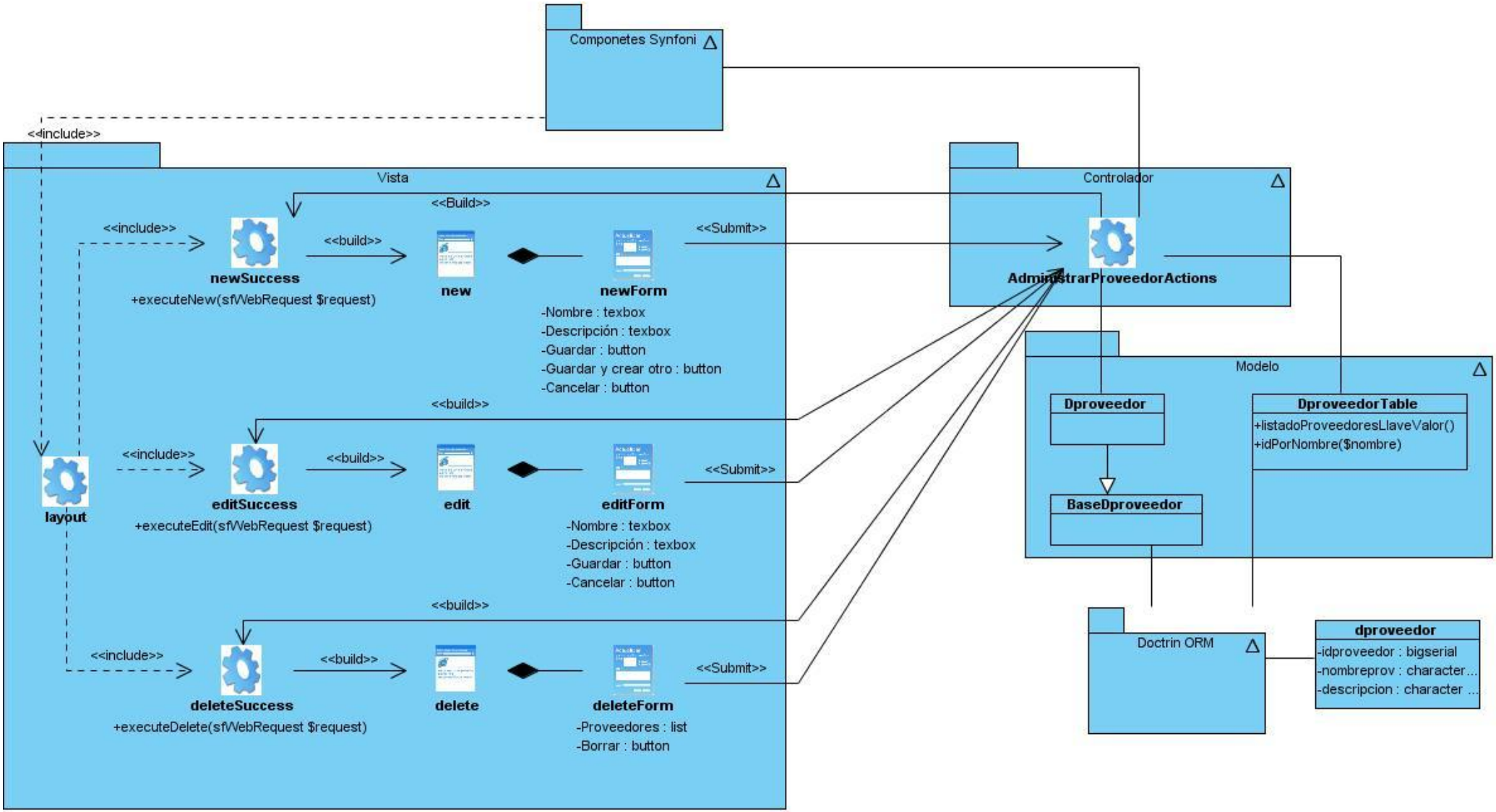


Figura 46: Diagrama de clases del diseño "Administrar proveedor" (Fuente: elaboración propia)

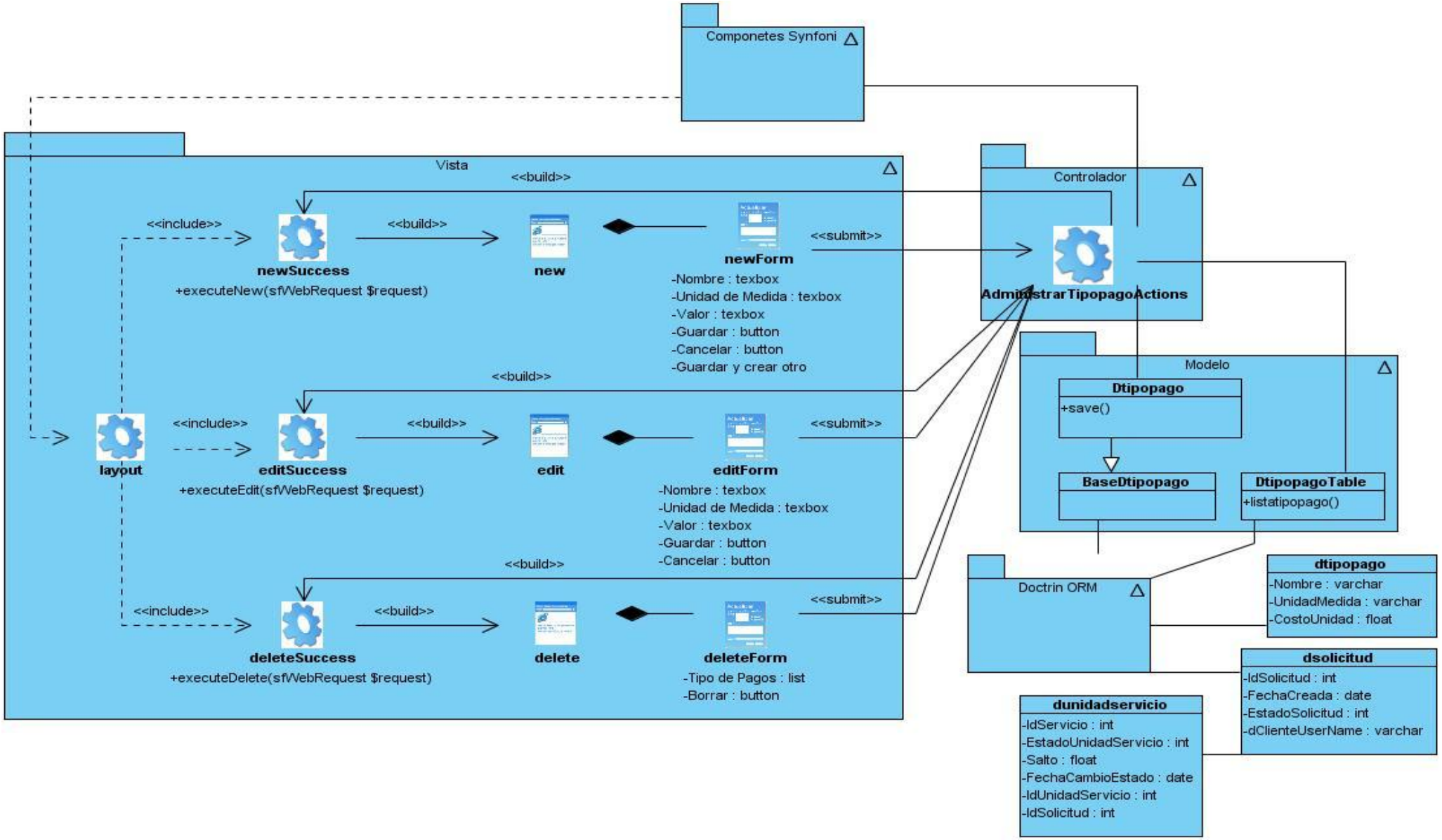


Figura 47: Diagrama de clases del diseño "Administrar tipos de pago" (Fuente: elaboración propia)

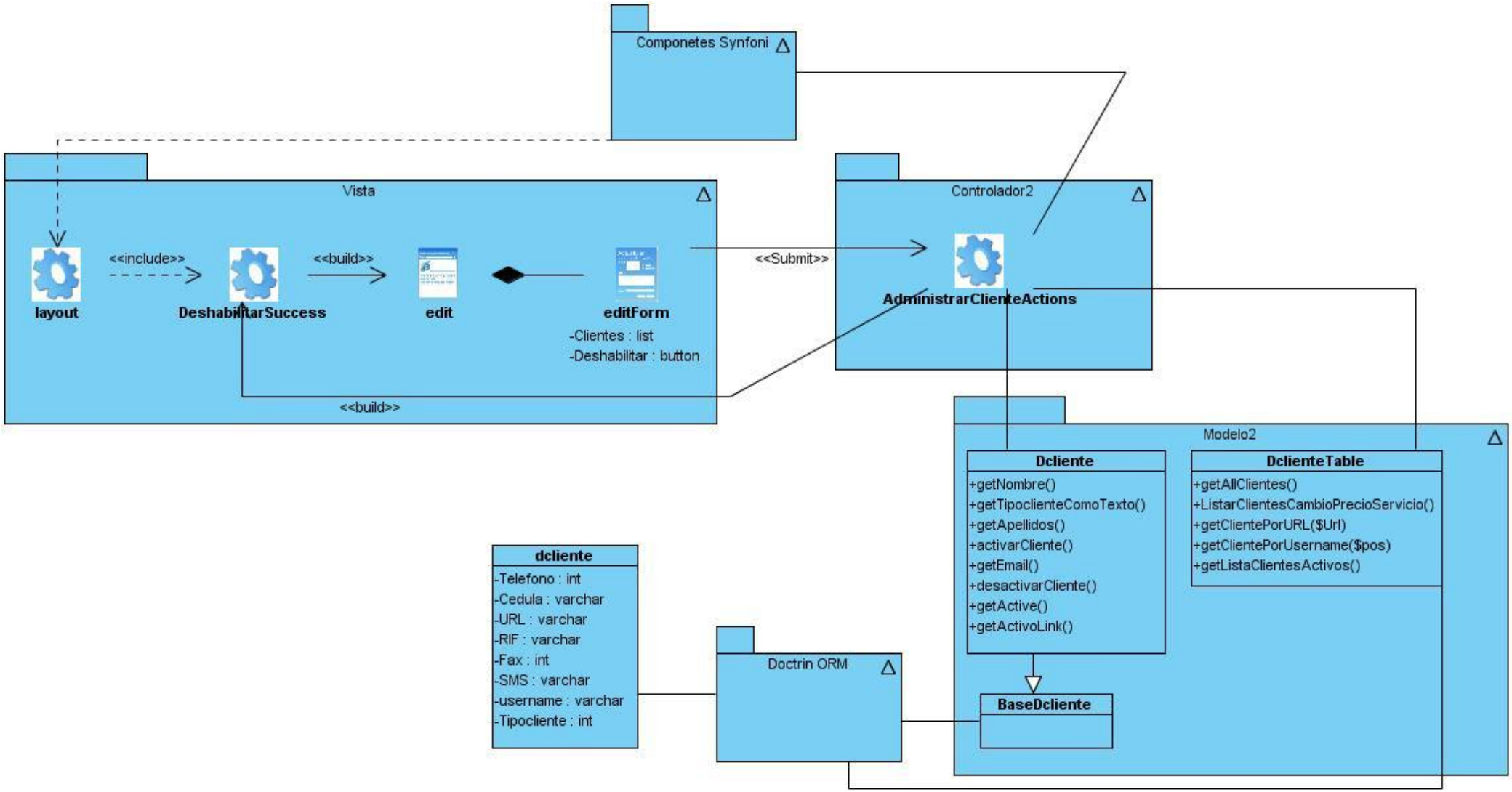


Figura 48: Diagrama de clases del diseño "Deshabilitar cliente" (Fuente: elaboración propia)

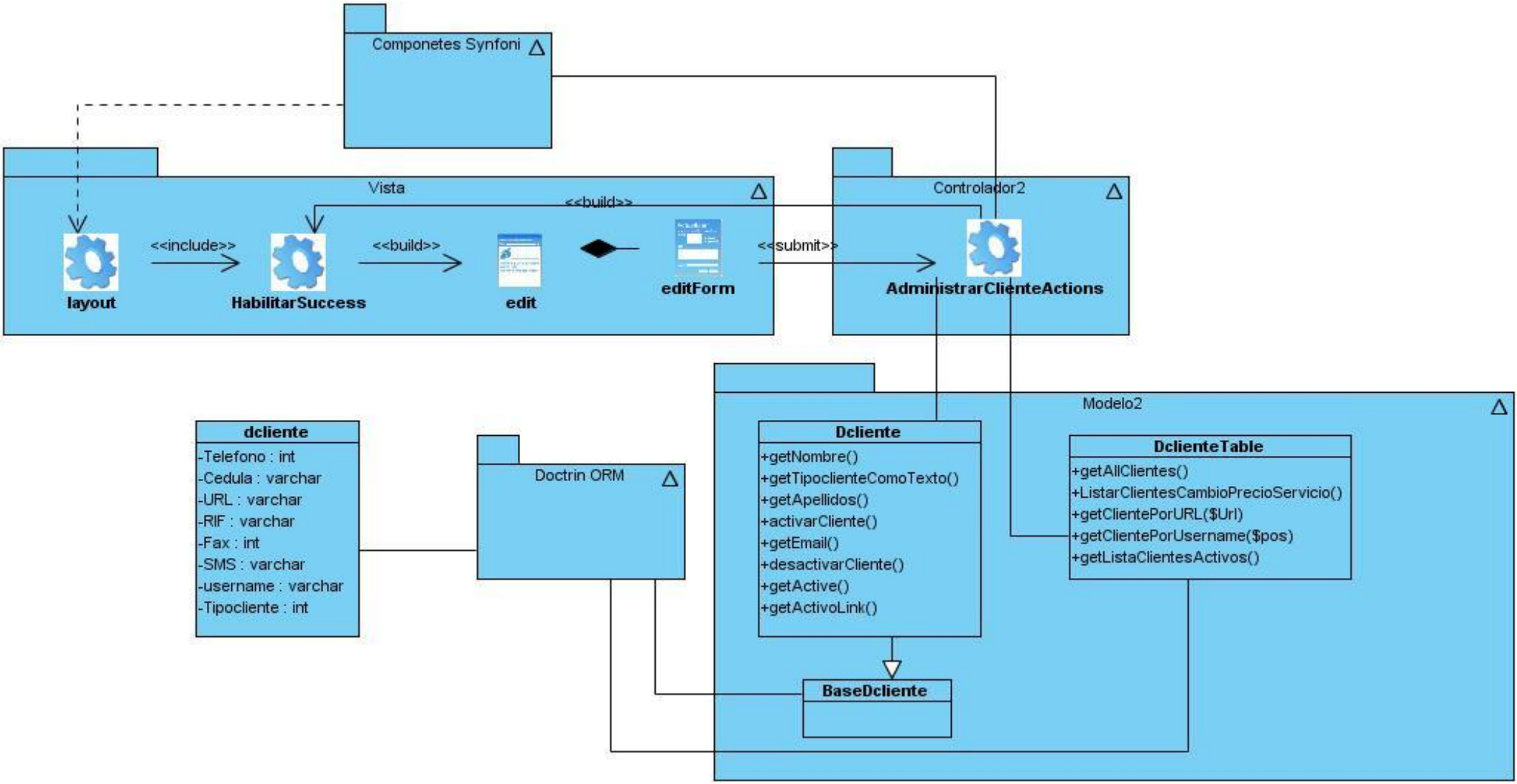


Figura 49: Diagrama de clases del diseño "Habilitar cliente" (Fuente: elaboración propia)

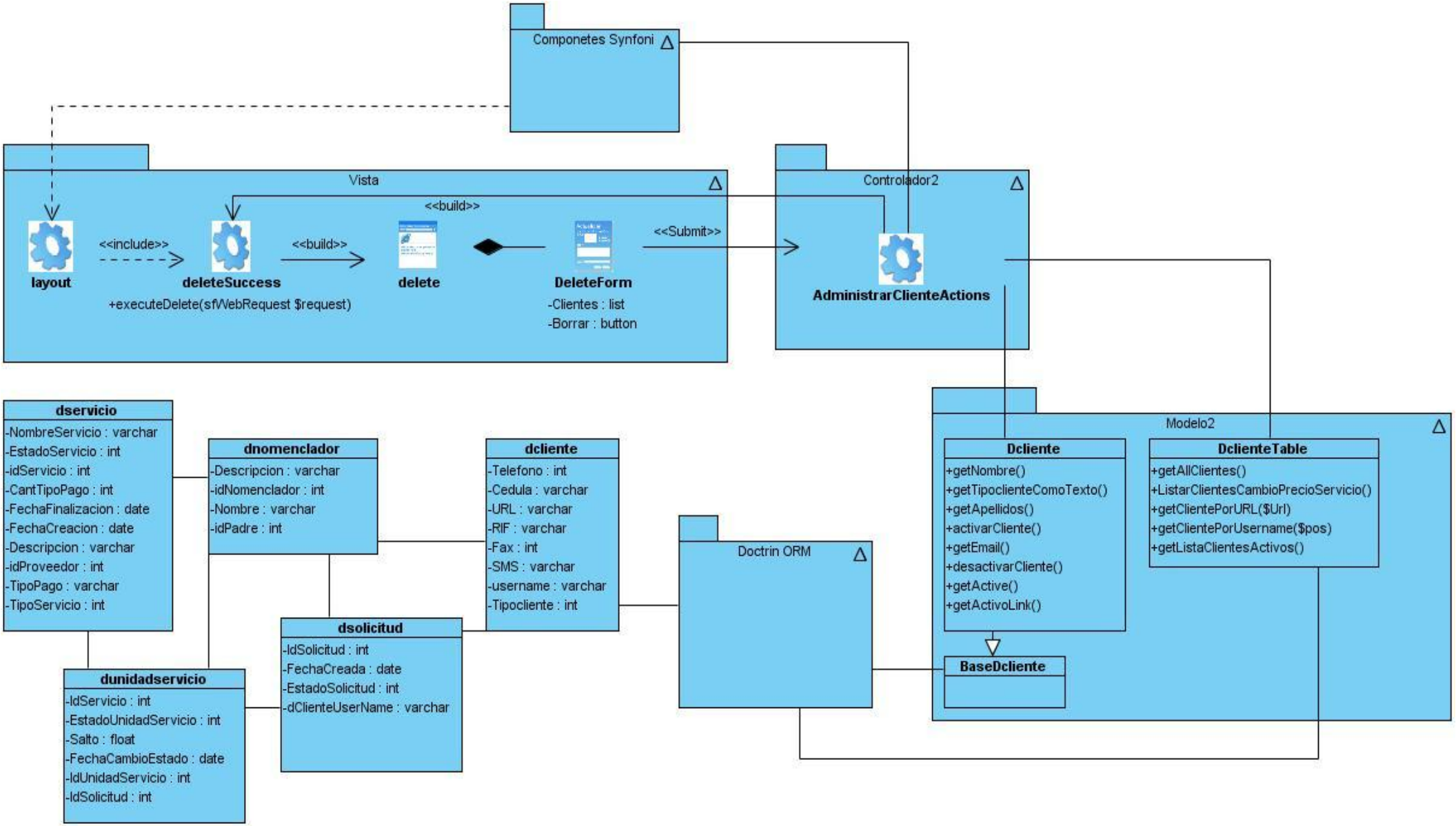


Figura 50: Diagrama de clases del diseño "Eliminar cliente" (Fuente: elaboración propia)

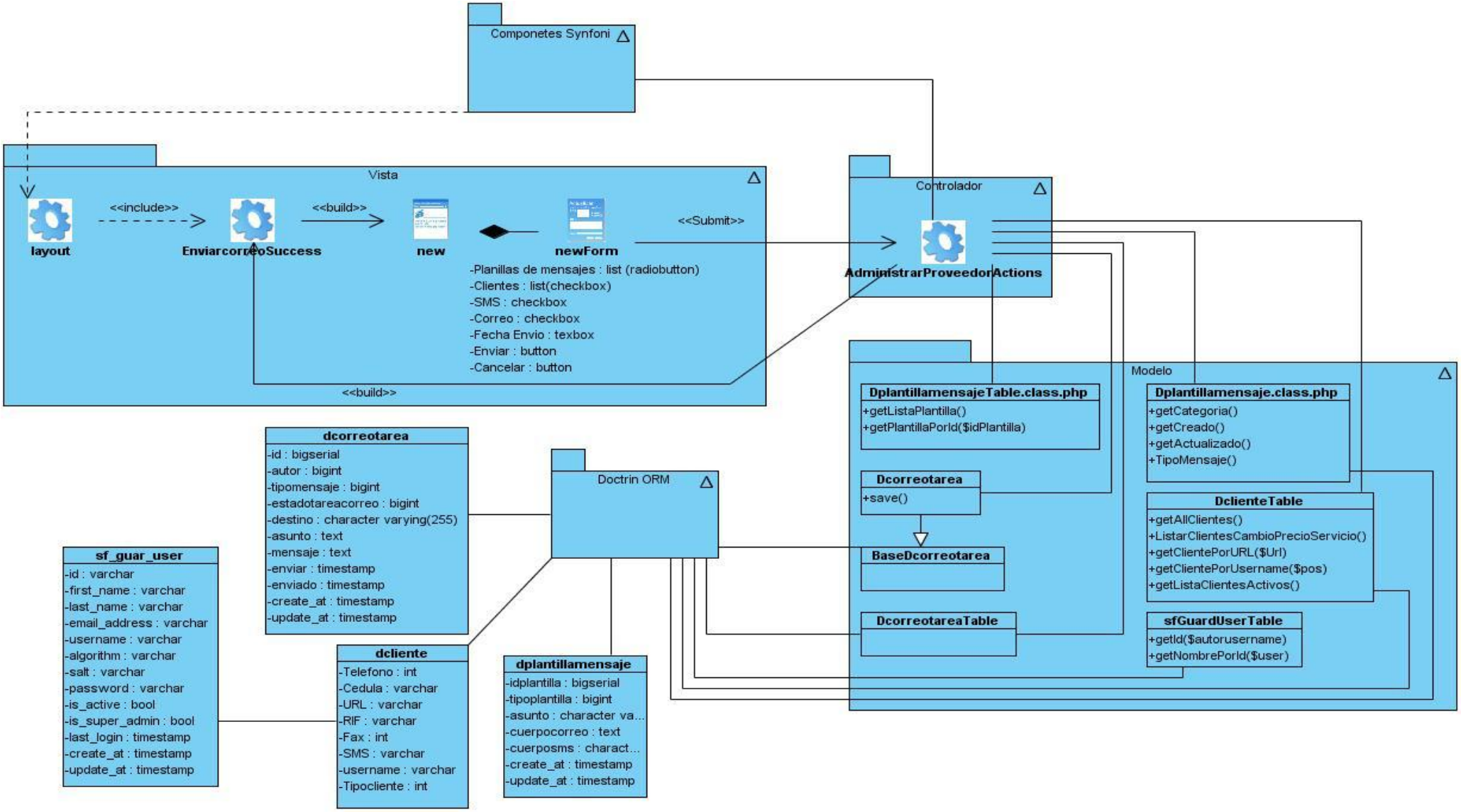


Figura 51: Diagrama de clases del diseño "Enviar aviso promocional manual o mensaje de correo" (Fuente: elaboración propia)

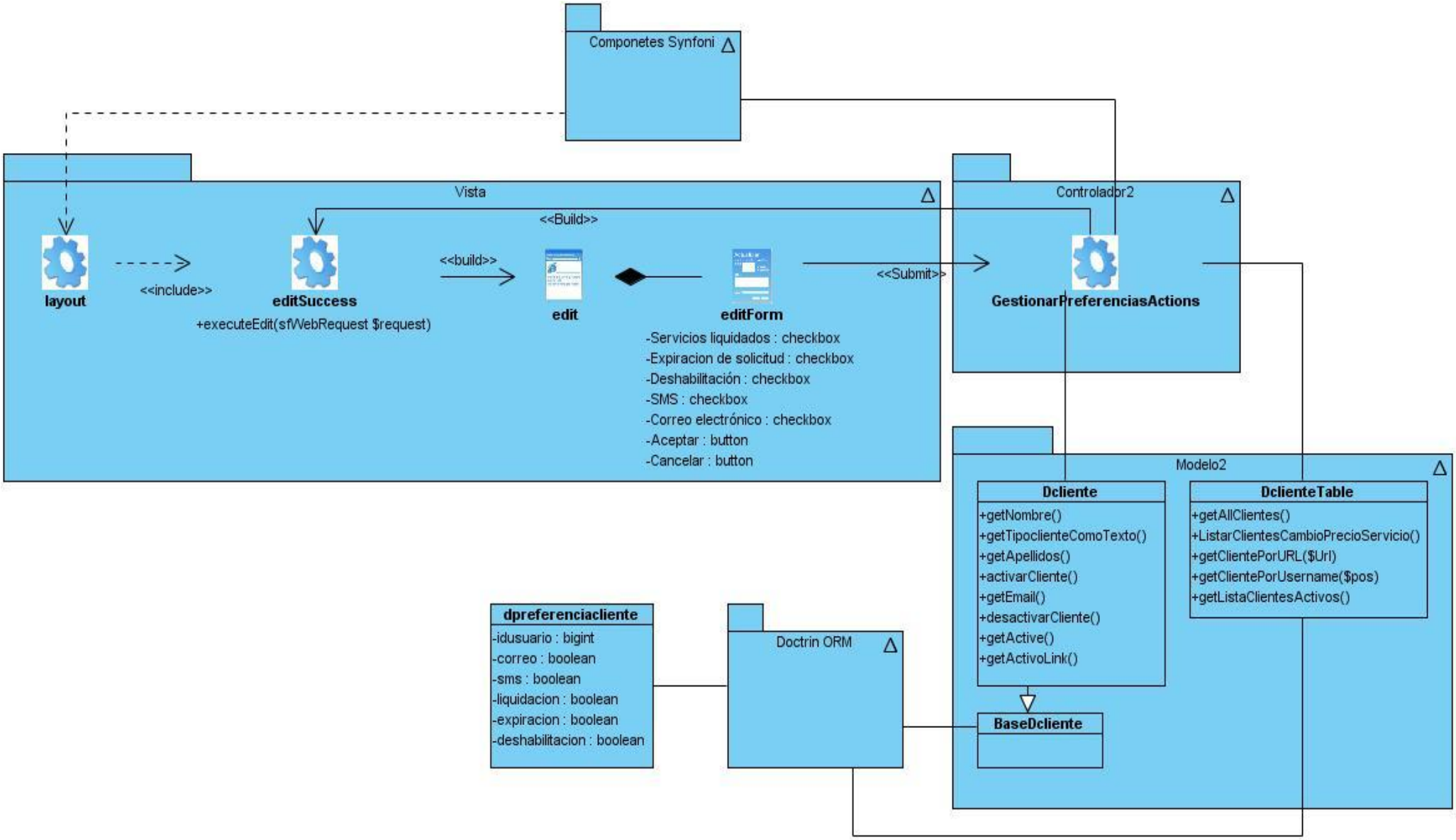


Figura 52: Diagrama de clases del diseño "Modificar preferencia de los clientes" (Fuente: elaboración propia)

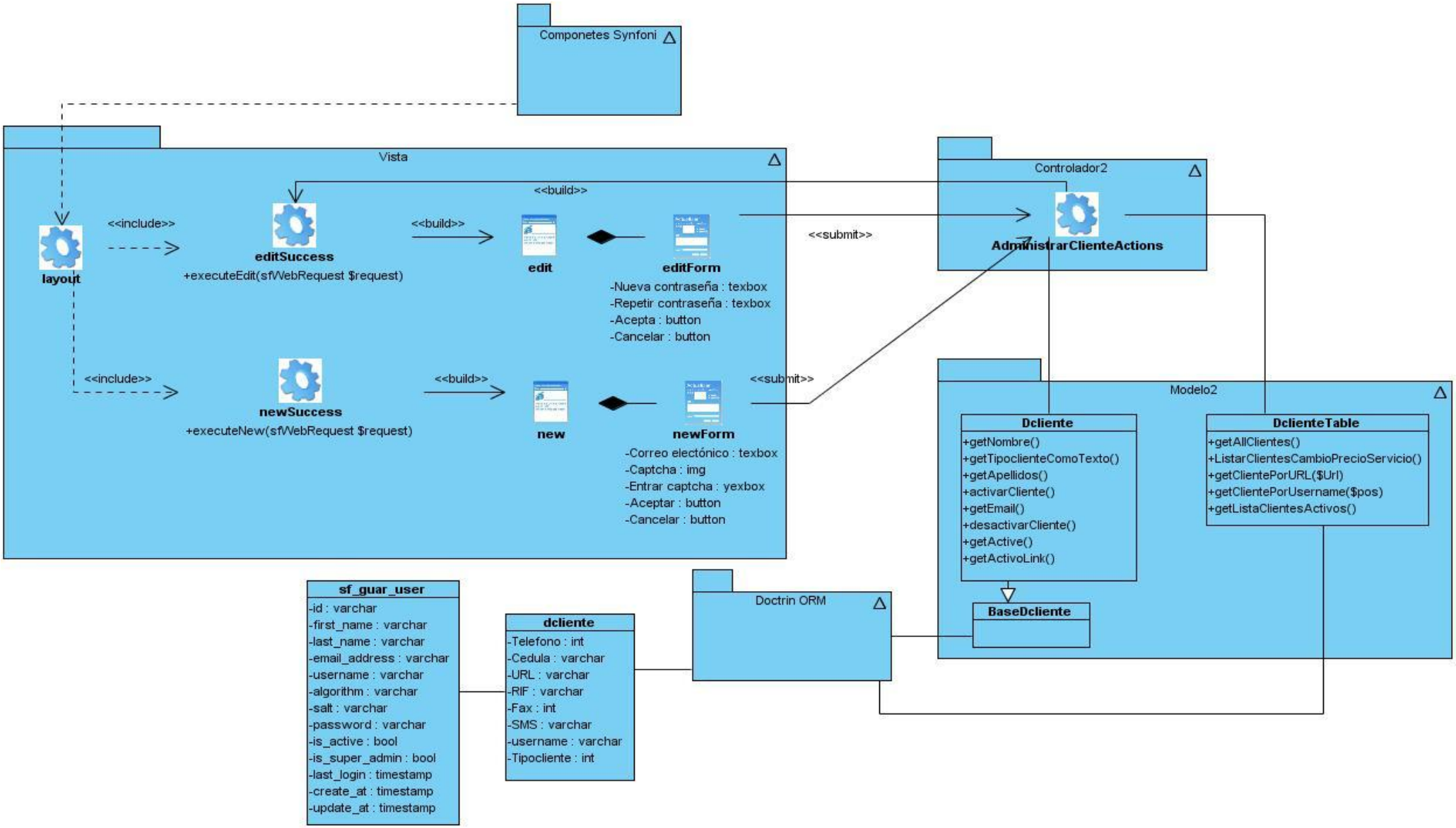


Figura 53: Diagrama de clases del diseño "Recuperar contraseña" (Fuente: elaboración propia)

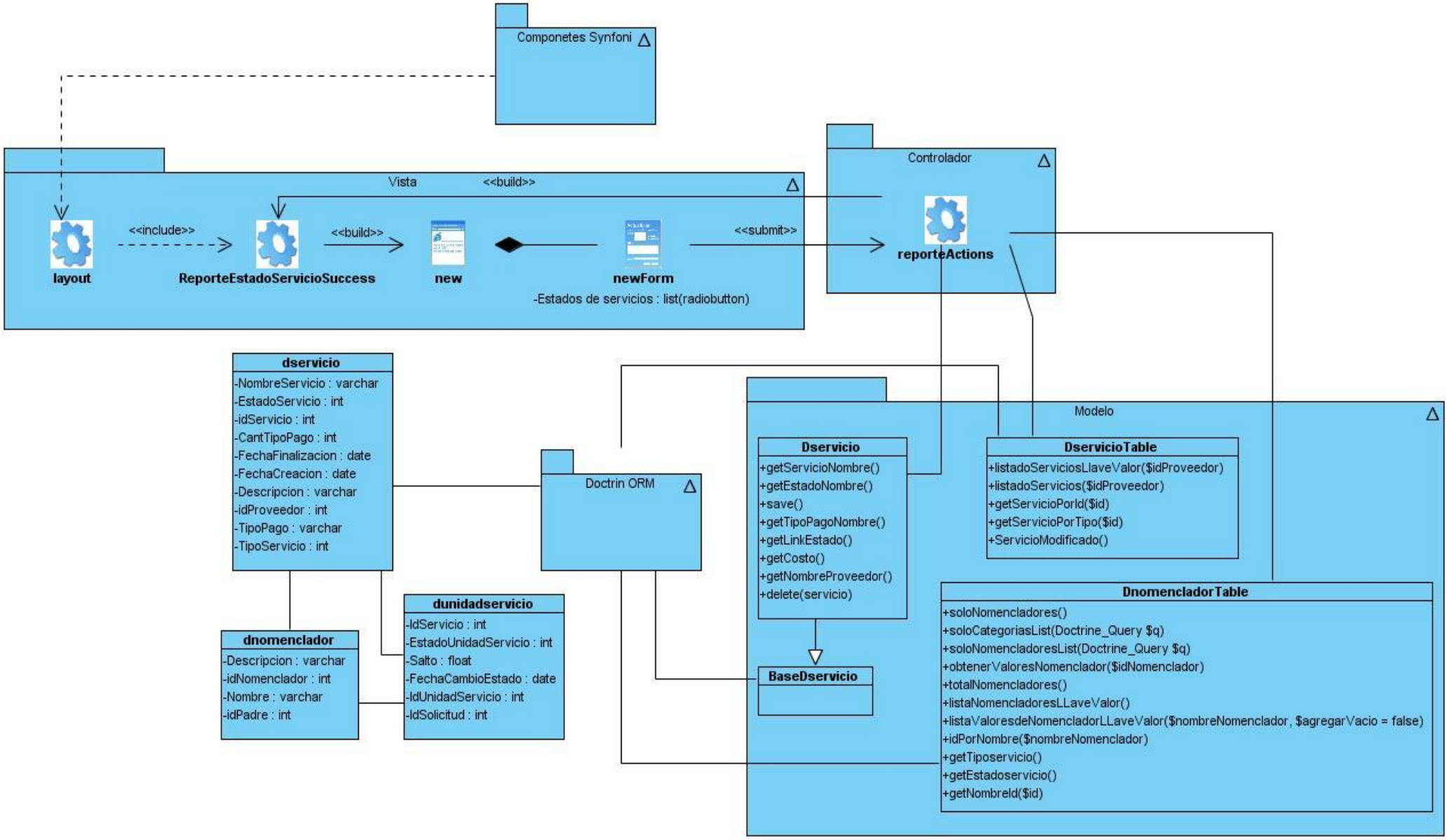


Figura 54: Diagrama de clases del diseño "Generar reporte de servicios por estado" (Fuente: elaboración propia)

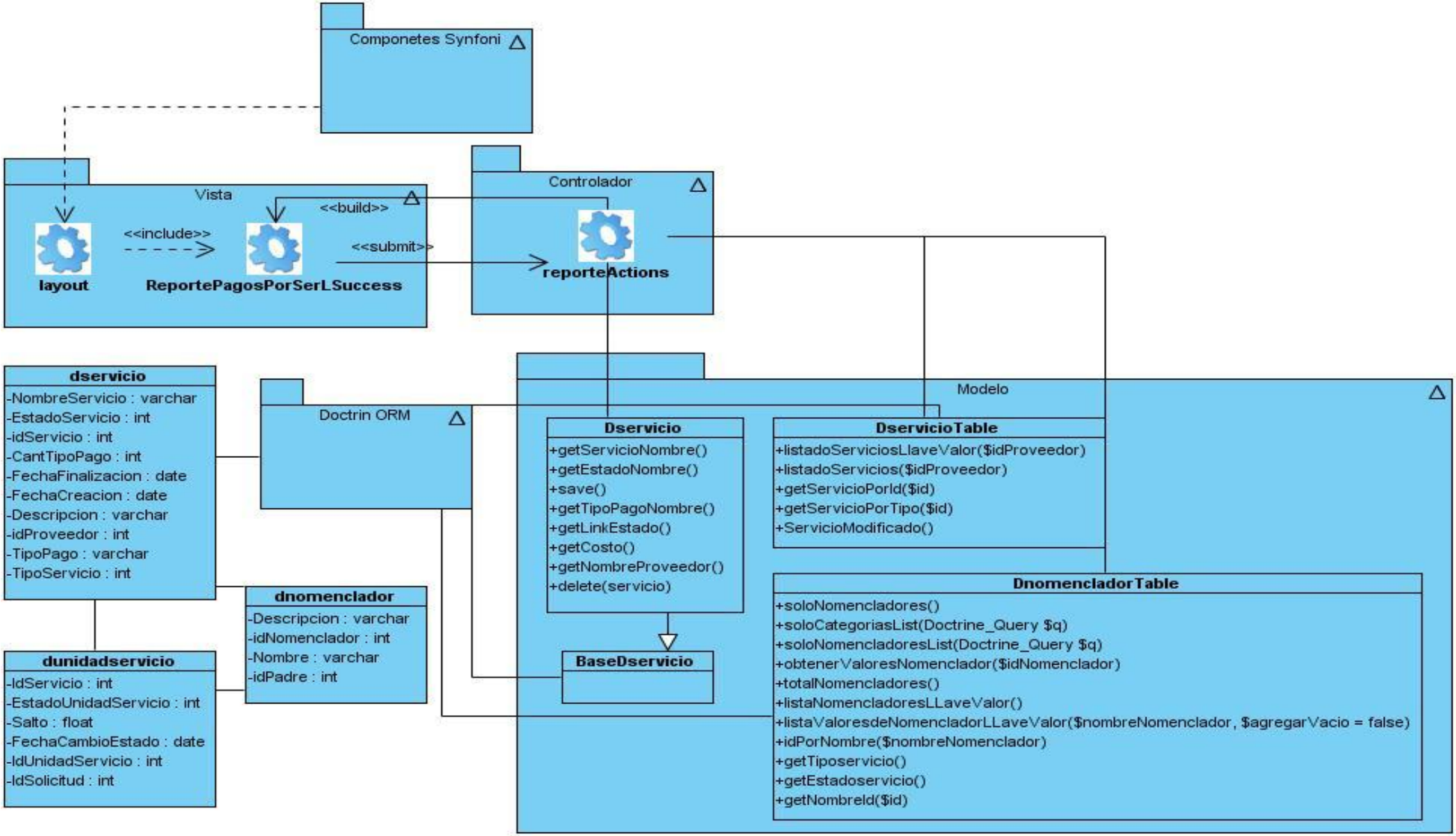


Figura 55: Diagrama de clases del diseño "Generar reporte de pagos diarios por servicios liquidados" (Fuente: elaboración propia)

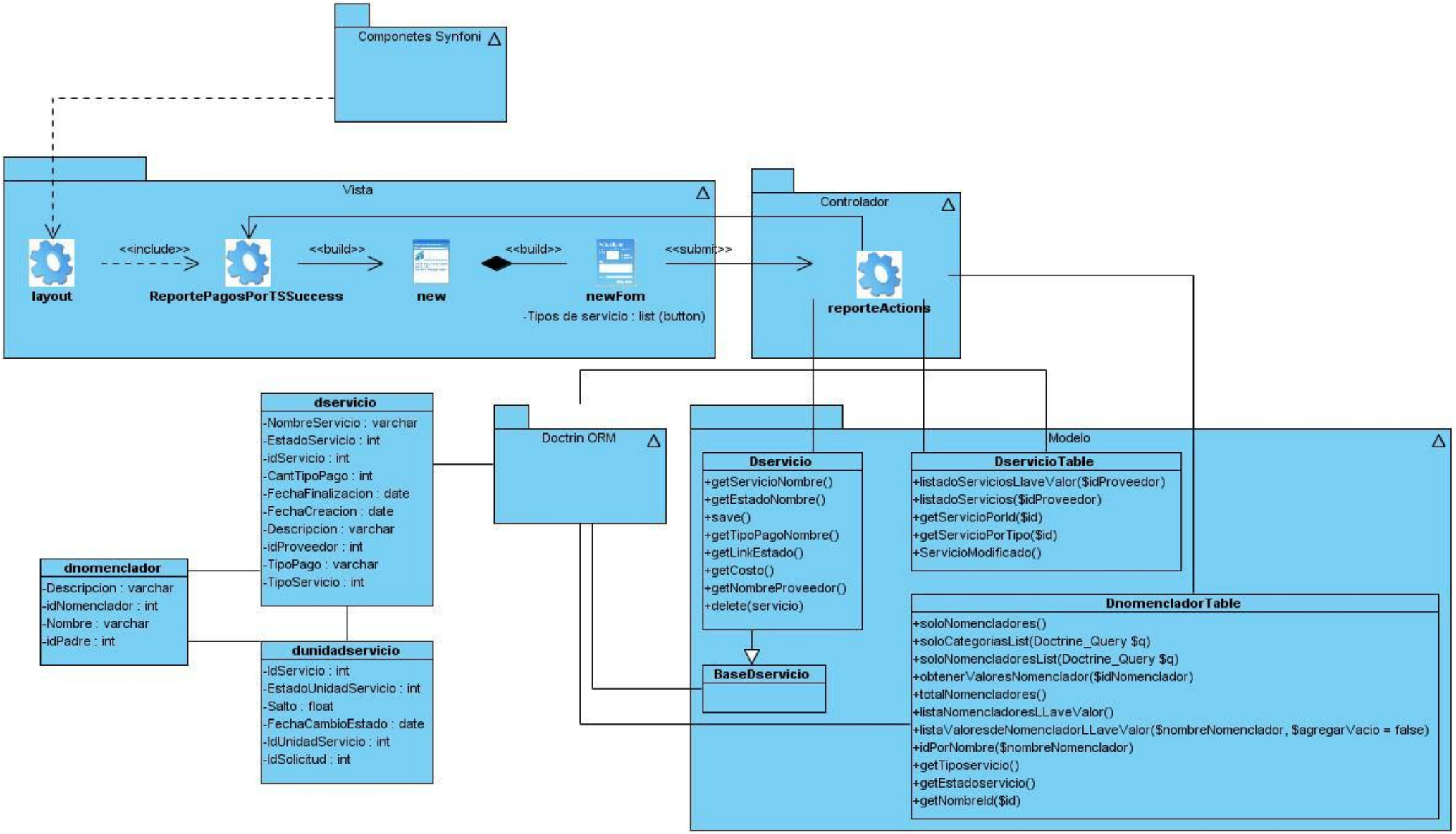


Figura 56: Diagrama de clases del diseño "Generar reporte de servicios por tipo" (Fuente: elaboración propia)

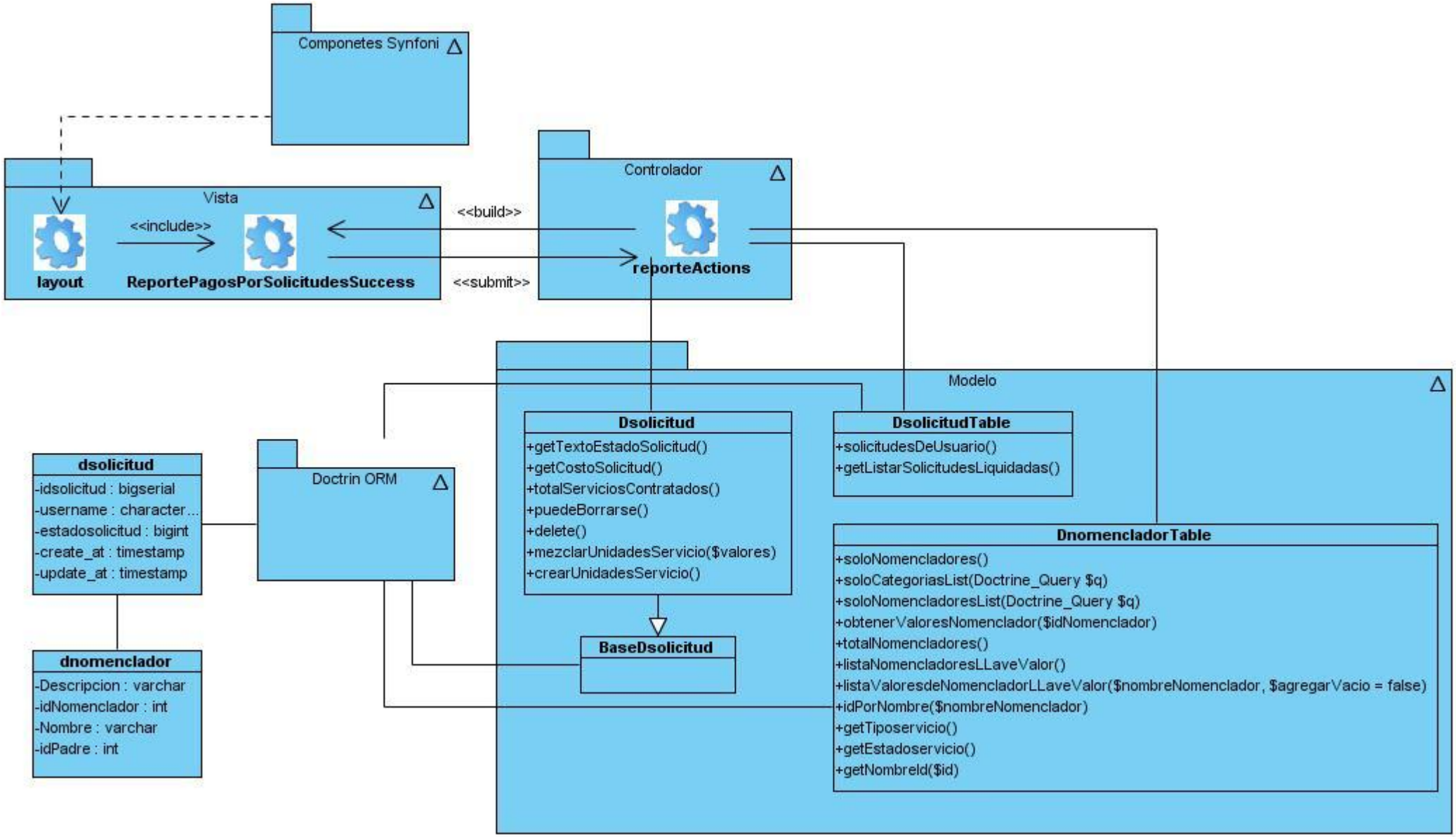


Figura 57: Diagrama de clases del diseño "Generar reporte de pagos diarios por solicitudes liquidadas" (Fuente: elaboración propia)

Anexo 4: Diagramas de componentes

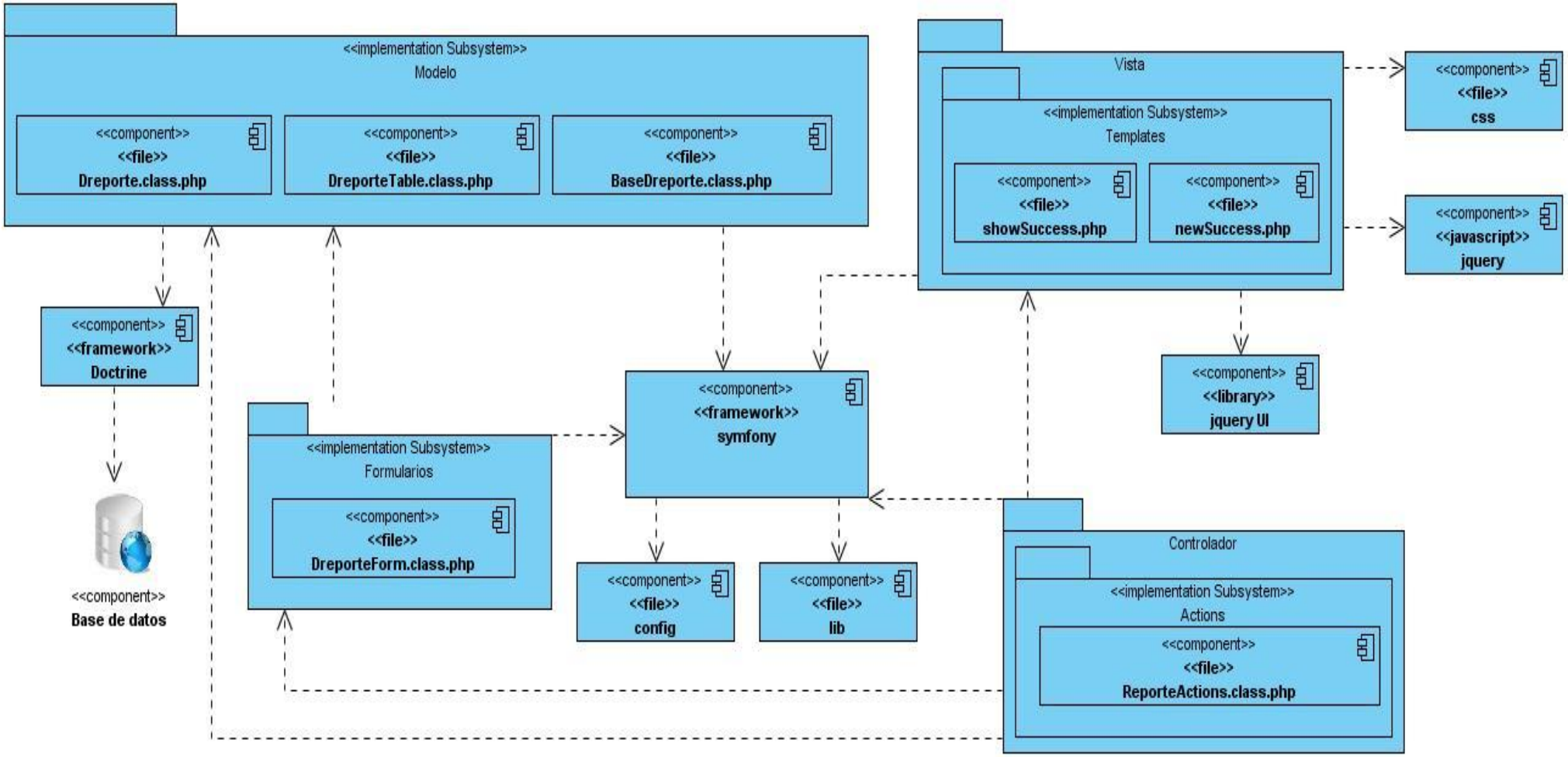


Figura 58: Diagramas de componentes Módulo reportes (Fuente: elaboración propia)

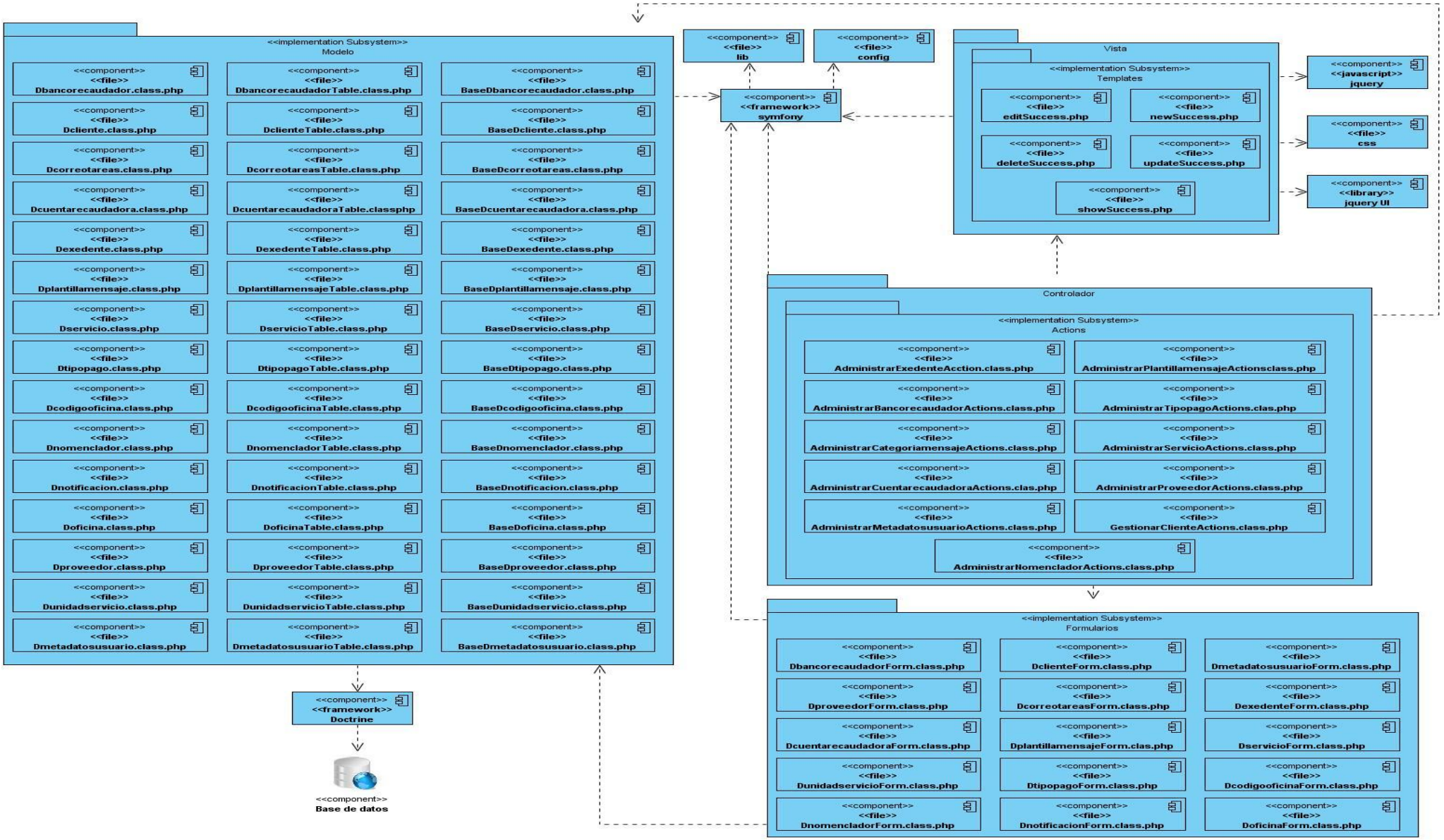


Figura 59: Diagramas de componentes Módulo administración (Fuente: elaboración propia)

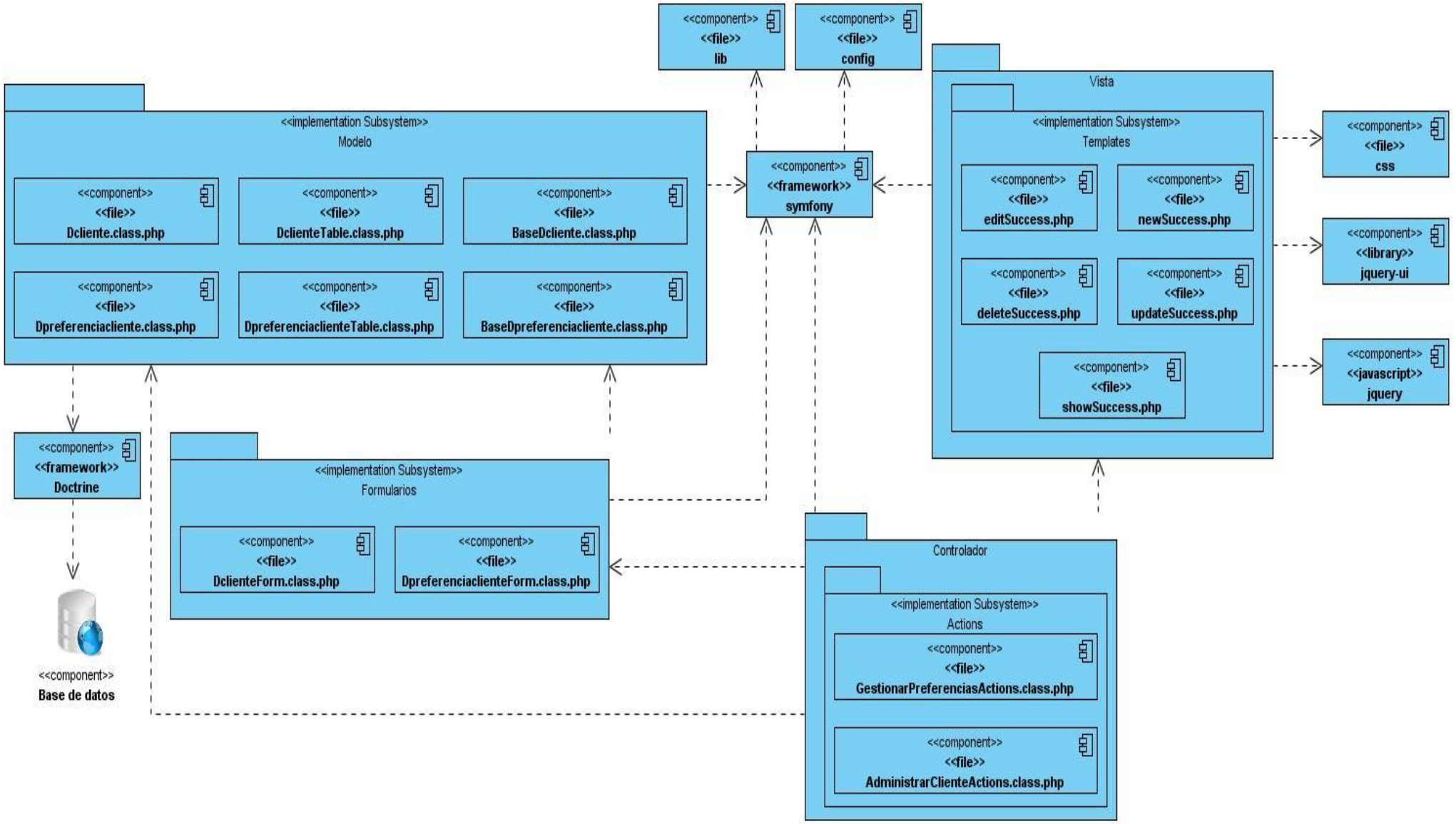


Figura 60: Diagrama de componentes del Módulo cliente (Fuente: elaboración propia)

Anexo 5: Casos de prueba de caja negra

Administrar bancos recaudadores

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Código	Campo de texto	No	Debe introducir un texto.
[2]	Nombre del banco	Campo de texto	No	Debe introducir un texto.
[3]	E-mail	Campo de texto	No	Debe introducir un texto.
[4]	Teléfono	Campo de texto	No	Debe introducir un texto.
[5]	FAX	Campo de texto	No	Debe introducir un texto.
[6]	RIF	Campo de texto	No	Debe introducir un texto.
[7]	Dirección	Campo seleccionable	No	Debe seleccionar un valor

[8]	Presta servicio de punto de venta	Campo seleccionable	No	Debe seleccionar un valor
[9]	Nombre del contacto	Campo de texto	No	Debe introducir un texto.
[10]	Apellido del contacto	Campo de texto	No	Debe introducir un texto.
[11]	Cédula del contacto	Campo de texto	No	Debe introducir un texto.

Tabla 11: Descripción de las variables del caso de uso "Administrar banco recaudador". (Fuente: elaboración propia)

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear banco recaudador.	V	V	V	V	V	V	V	V	V	V	V	Crea un nuevo banco recaudador.	Satisfactoria	
	I	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Introduzca el código del banco" o "El código del banco tiene que contener 4 dígitos."	Satisfactoria	
	V	V	I	V	V	V	V	V	V	V	V	Muestra el mensaje "Incorrecto".	Satisfactoria	
	V	V	V	I	V	V	V	V	V	V	V	Muestra el mensaje" El número telefónico no es válido."	Satisfactoria	

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	I	V	V	V	V	V	V	Muestra el mensaje" El número del fax no es válido".	Satisfactoria	
	V	V	V	V	V	V	V	V	I	V	V	Muestra el mensaje "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres" o "Introduzca el nombre del contacto."	Satisfactoria	

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	V	V	V	V	I	V	Muestra el mensaje "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres" o "Introduzca el apellido del contacto."	Satisfactoria	
	V	V	V	V	V	V	V	V	V	V	I	Muestra el mensaje "El número de cédula no es válido" o "Introduzca la cédula del contacto."	Satisfactoria	
Modificar banco recauda	V	V	V	V	V	V	V	V	V	V	V	Modifica un banco recaudador.	Satisfactoria	

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
dor	I	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Introduzca el código del banco" o "El código del banco tiene que contener 4	Satisfactoria	
	V	V	I	V	V	V	V	V	V	V	V	Muestra el mensaje "Incorrecto".	Satisfactoria	
	V	V	V	I	V	V	V	V	V	V	V	Muestra el mensaje" El número telefónico no es válido."	Satisfactoria	
	V	V	V	V	I	V	V	V	V	V	V	Muestra el mensaje" El número del fax no es válido".	Satisfactoria	

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	V	V	V	I	V	V	Muestra el mensaje “Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres” o “Introduzca el nombre del contacto.”	Satisfactoria	
	V	V	V	V	V	V	V	V	V	I	V	Muestra el mensaje “Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres” o “Introduzca el apellido del contacto.”	Satisfactoria	

Flujos	Var 1. Código	Var 2. Nombre del banco	Var 3 .E-mail	Var 4. Teléfono	Var 5. FAX	Var 6. RIF	Var 7. Dirección	Var 8. Presta servicio de punto de venta	Var 9. Nombre del contacto	Var 10. Apellidos del contacto	Var 11. Cédula del contacto	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	V	V	V	V	V	I	Muestra el mensaje “El número de cédula no es válido” o “Introduzca la cédula del contacto.”	Satisfactoria	
Eliminar banco recaudador.	V	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Elimina un banco recaudador.	Satisfactoria	
	I	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Muestra el mensaje: “Se debe seleccionar al menos un elemento.”	Satisfactoria	

Tabla 12: Iteraciones de las variables del caso de uso "Administrar banco recaudador". (Fuente: elaboración propia)

Administrar categoría

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Categoría	Campo de texto	No	Debe introducir un texto.
[2]	Descripción	Campo de texto	No	Debe introducir un texto.

Tabla 13: Descripción de las variables del caso de uso "Administrar categoría". (Fuente: elaboración propia)

Flujos	Var 1. Categoría	Var 2. Descripción	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear categoría.	V	V	Crea una nueva categoría.	Satisfactoria	
	I	V	Muestra el mensaje “Introduzca el nombre” o “Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres.”	Satisfactoria	
Eliminar categoría.	V	NA	Elimina una categoría.	Satisfactoria	

Flujos	Var 1. Categoría	Var 2. Descripción	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	I	NA	Muestra el mensaje: “Se debe seleccionar al menos un elemento.”	Satisfactoria	

Tabla 14: Iteraciones de las variables del caso de uso "Administrar categoría". (Fuente: elaboración propia)

Administrar clientes

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Usuario	Campo de texto	No	Debe introducir un texto.
[2]	Nombre	Campo de texto	No	Debe introducir un texto.
[3]	Apellidos	Campo de texto	No	Debe introducir un texto.
[4]	Correo electrónico	Campo de texto	No	Debe introducir un texto.
[5]	Contraseña	Campo de texto	No	Debe introducir un texto.
[6]	Repetir Contraseña	Campo de texto	No	Debe introducir un texto.

[7]	Cédula	Campo seleccionable	No	Debe seleccionar un valor
[8]	SMS	Campo seleccionable	No	Debe seleccionar un valor
[9]	RIF	Campo de texto	No	Debe introducir un texto.
[10]	Teléfono	Campo de texto	No	Debe introducir un texto.
[11]	FAX	Campo de texto	No	Debe introducir un texto.
[12]	Captcha	Campo de texto	No	Debe introducir un texto.

Tabla 15: Descripción de las variables del caso de uso "Administrar cliente". (Fuente: elaboración propia)

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear cliente.	V	V	V	V	V	V	V	V	V	V	V	V	Crea un nuevo cliente.	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	I	V	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar el usuario" o "Valor no válido, solo puede contener números, letras, "_" y de 3 a 100 caracteres."	Satisfactoria	
	V	I	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar un nombre" o "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres."	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	I	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su apellido" o "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres."	Satisfactoria	
	V	V	V	I	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar una dirección de correo" o "La dirección de correo no es válida."	Satisfactoria	
	V	V	V	V	I	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su contraseña" o "La contraseña es muy pequeña" o "Las dos contraseñas deben ser iguales."	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	I	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su contraseña" o "La contraseña es muy pequeña" o "Las dos contraseñas deben ser iguales."	Satisfactoria	
	V	V	V	V	V	V	I	V	V	V	V	V	Muestra el mensaje" Introduzca la cédula del contacto" o "El número de cédula no es válido"	Satisfactoria	
	V	V	V	V	V	V	V	I	V	V	V	V	Muestra el mensaje" El número del SMS no es válido".	Satisfactoria	
	V	V	V	V	V	V	V	V	I	V	V	V	Muestra el mensaje "La cadena es demasiado corta."	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	V	V	V	V	I	V	V	Muestra el mensaje “El número telefónico no es válido.”	Satisfactoria	
	V	V	V	V	V	V	V	V	V	V	I	V	Muestra el mensaje “El número del fax no es válido.”	Satisfactoria	
	V	V	V	V	V	V	V	V	V	V	V	I	Muestra el mensaje “Campo obligatorio” o “Incorrecto”	Satisfactoria	
Modificar cliente.	V	V	V	V	V	V	V	V	V	V	V	V	Modifica un cliente.	Satisfactoria	
	I	V	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje” Debe entrar el usuario” o “Valor no válido, solo puede contener números, letras, "_" y de 3 a 100 caracteres.”	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	I	V	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar un nombre" o "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres."	Satisfactoria	
	V	V	I	V	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su apellido" o "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres."	Satisfactoria	
	V	V	V	I	V	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar una dirección de correo" o "La dirección de correo no es válida."	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	I	V	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su contraseña" o "La contraseña es muy pequeña" o "Las dos contraseñas deben ser iguales."	Satisfactoria	
	V	V	V	V	V	I	V	V	V	V	V	V	Muestra el mensaje" Debe entrar su contraseña" o "La contraseña es muy pequeña" o "Las dos contraseñas deben ser iguales."	Satisfactoria	
	V	V	V	V	V	V	I	V	V	V	V	V	Muestra el mensaje" Introduzca la cédula del contacto" o "El número de cédula no es válido"	Satisfactoria	

Flujos	Var 1. Usuario	Var 2. Nombre	Var 3 .Apellidos	Var 4. Correo electrónico	Var 5. Contraseña	Var 6. Repetir contraseña	Var 7. Cédula	Var 8. SMS	Var 9. RIF	Var 10. Teléfono	Var 11. FAX	Var 12. Captcha	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	V	V	V	V	I	V	V	V	V	Muestra el mensaje" El número del SMS no es válido".	Satisfactoria	
	V	V	V	V	V	V	V	V	I	V	V	V	Muestra el mensaje "La cadena es demasiado corta."	Satisfactoria	
	V	V	V	V	V	V	V	V	V	I	V	V	Muestra el mensaje "El número telefónico no es válido."	Satisfactoria	
	V	V	V	V	V	V	V	V	V	V	I	V	Muestra el mensaje "El número del fax no es válido."	Satisfactoria	
	V	V	V	V	V	V	V	V	V	V	V	I	Muestra el mensaje "Campo obligatorio" o "Incorrecto"	Satisfactoria	

Tabla 16: Iteraciones de las variables del caso de uso "Administrar cliente". (Fuente: elaboración propia)

Administrar cuantas recaudadoras

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Código	Campo de texto	No	Debe introducir un texto.
[2]	Banco	Campo seleccionable	No	Debe seleccionar un valor
[3]	Proveedor	Campo seleccionable	No	Debe seleccionar un valor
[4]	Monto	Campo de texto	No	Debe introducir un texto.
[5]	Estado	Campo seleccionable	No	Debe seleccionar un valor

Tabla 17: Descripción de las variables del caso de uso "Administrar cuenta recaudadora". (Fuente: elaboración propia)

Flujos	Var 1. Código	Var 2. Banco	Var 3 .Proveedor	Var 4. Monto	Var 5. Estado	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear cuenta recaudadora.	V	V	V	V	V	Crea una nueva cuenta recaudadora	Satisfactoria	

Flujos	Var 1. Código	Var 2. Banco	Var 3 .Proveedor	Var 4. Monto	Var 5. Estado	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	I	V	V	V	V	Muestra el mensaje" Introduzca el número de la cuenta" o "Formato de cuenta incorrecto, ej:0007-0141-64-0060183212"	Satisfactoria	
	V	V	V	I	V	Muestra el mensaje" Campo obligatorio" o "Incorrecto"	Satisfactoria	
Modificar cuenta recaudadora.	V	V	V	V	V	Modifica una cuenta recaudadora.	Satisfactoria	
	I	V	V	V	V	Muestra el mensaje" Introduzca el número de la cuenta" o "Formato de cuenta incorrecto, ej:0007-0141-64-0060183212"	Satisfactoria	
	V	V	V	I	V	Muestra el mensaje" Campo obligatorio" o "Incorrecto"	Satisfactoria	

Flujos	Var 1. Código	Var 2. Banco	Var 3 .Proveedor	Var 4. Monto	Var 5. Estado	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Eliminar banco recaudador.	V	NA	NA	NA	NA	Elimina una cuenta recaudadora.	Satisfactoria	
	I	NA	NA	NA	NA	Muestra el mensaje: "Se debe seleccionar al menos un elemento."	Satisfactoria	

Tabla 18: Iteraciones de las variables del caso de uso "Administrar cuenta recaudadora". (Fuente: elaboración propia)

Administrar listas de distribución

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo de texto	No	Debe introducir un texto.
[2]	Descripción	Campo de texto	No	Debe introducir un texto.
[3]	Lista de Usuarios	Campo seleccionable	No	Debe seleccionar un valor

Tabla 19: Descripción de las variables del caso de uso "Administrar lista de distribución". (Fuente: elaboración propia)

Flujos	Var 1. Nombre	Var 2. Descripción	Var 3 Lista de Usuarios	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear lista de distribución.	V	V	V	Crea una nueva lista de distribución.	Satisfactoria	
	I	V	V	Muestra el mensaje “Introduzca el nombre de la lista” o “Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres.”	Satisfactoria	
	V	V	I	Muestra el mensaje “La lista debe contener al	Satisfactoria	
Modificar lista de distribución.	V	V	V	Modifica una lista de distribución	Satisfactoria	

Flujos	Var 1. Nombre	Var 2. Descripción	Var 3 Lista de Usuarios	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	I	V	V	Muestra el mensaje "Introduzca el nombre de la lista" o "Valor no válido, solo puede contener números, letras y de 3 a 150 caracteres."	Satisfactoria	
	V	V	I	Muestra el mensaje "La lista debe contener al menos un usuario."	Satisfactoria	
Eliminar lista de distribución.	V	NA	NA	Elimina una lista de distribución.	Satisfactoria	
	I	NA	NA	Muestra el mensaje: "Se debe seleccionar al menos un elemento."	Satisfactoria	

Tabla 20: Iteraciones de las variables del caso de uso "Administrar lista de distribución". (Fuente: elaboración propia)

Administrar mensajes de correo

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Asunto	Campo de texto	No	Debe introducir un texto.
[2]	Correo electrónico	Campo de texto	No	Debe introducir un texto.
[3]	SMS	Campo de texto	No	Debe introducir un texto.
[4]	Categoría	Campo de texto	No	Debe introducir un texto.

Tabla 21: Descripción de las variables del caso de uso "Administrar mensaje de correo". (Fuente: elaboración propia)

Flujos	Var 1. Asunto	Var 2. Correo	Var 3 .SMS	Var 4. Categoría	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear mensajes de correo electrónico.	V	V	V	V	Crea un nuevo mensaje de correo.	Satisfactoria	
	I	V	V	V	Muestra el mensaje “Escribir el asunto”.	Satisfactoria	

Flujos	Var 1. Asunto	Var 2. Correo	Var 3 .SMS	Var 4. Categoría	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	I	V	V	Muestra el mensaje "El cuerpo del mensaje no puede estar vacío."	Satisfactoria	
	V	V	I	V	Muestra el mensaje "El cuerpo del SMS no puede estar vacío".	Satisfactoria	
Modificar mensajes de correo electrónico.	V	V	V	V	Modifica un mensaje de correo.	Satisfactoria	
	I	V	V	V	Muestra el mensaje "Escribir el asunto".	Satisfactoria	
	V	I	V	V	Muestra el mensaje "El cuerpo del mensaje no puede estar vacío."	Satisfactoria	
	V	V	I	V	Muestra el mensaje "El cuerpo del SMS no puede estar vacío".	Satisfactoria	
Eliminar mensajes de correo	V	NA	NA	NA	Elimina un mensaje de correo.	Satisfactoria	

Flujos	Var 1. Asunto	Var 2. Correo	Var 3 .SMS	Var 4. Categoría	Respuesta del Sistema	Resultado de la Prueba	Observaciones
electrónico.	I	NA	NA	NA	Muestra el mensaje: "Se debe seleccionar al menos un elemento."	Satisfactoria	

Tabla 22: Iteraciones de las variables del caso de uso "Administrar mensajes de correo". (Fuente: elaboración propia)

Administrar proveedor

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo de texto	No	Debe introducir un texto.
[2]	Descripción	Campo de texto	No	Debe introducir un texto.

Tabla 23: Iteraciones de las variables del caso de uso "Administrar proveedor". (Fuente: elaboración propia)

Flujos	Var 1. Nombre	Var 2. Descripción	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear proveedor.	V	V	Crea un nuevo proveedor.	Satisfactoria	
	I	V	Muestra el mensaje “Introduzca el nombre”.	Satisfactoria	
Modificar proveedor	V	V	Modifica proveedor.	Satisfactoria	
	I	V	Muestra el mensaje “Introduzca el nombre”.	Satisfactoria	
Eliminar proveedor	V	NA	Elimina un mensaje de correo.	Satisfactoria	
	I	NA	Muestra el mensaje: “Se debe seleccionar al menos un elemento.”	Satisfactoria	

Tabla 24: Iteraciones de las variables del caso de uso "Administrar proveedor". (Fuente: elaboración propia)

Administrar tipos de pago

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo de texto	No	Debe introducir un texto.
[2]	Unidad de medida	Campo de texto	No	Debe introducir un texto.
[3]	Valor	Campo de texto	No	Debe introducir un texto.

Tabla 25: Descripción de las variables del caso de uso "Administrar tipos de pago". (Fuente: elaboración propia)

Flujos	Var 2. Nombre	Var 3 .Unidad de medida	Var 4. Valor	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Crear tipo de pago	V	V	V	Crea un nuevo tipo de pago.	Satisfactoria	
	I	V	V	Muestra el mensaje" Debe entrar un Nombre".	Satisfactoria	
	V	I	V	Muestra el mensaje" Introduzca la unidad de medida".	Satisfactoria	
	V	V	I	Muestra el mensaje "Campo obligatorio" o "Incorrecto".	Satisfactoria	

Flujos	Var 2. Nombre	Var 3 .Unidad de medida	Var 4. Valor	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Modificar tipo de pago	V	V	V	Crea un nuevo tipo de pago.	Satisfactoria	
	I	V	V	Muestra el mensaje" Debe entrar un Nombre".	Satisfactoria	
	V	I	V	Muestra el mensaje" Introduzca la unidad de medida".	Satisfactoria	
	V	V	I	Muestra el mensaje "Campo obligatorio" o "Incorrecto".	Satisfactoria	
Eliminar tipo de pago	V	NA	NA	Elimina un banco recaudador.	Satisfactoria	
	I	NA	NA	Muestra el mensaje: "Se debe seleccionar al menos un elemento."	Satisfactoria	

Tabla 26: Iteraciones de las variables del caso de uso "Administrar tipo de pago". (Fuente: elaboración propia)

Deshabilitar cliente

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo seleccionable	No	Debe seleccionar un valor

Tabla 27: Descripción de las variables del caso de uso "Deshabilitar cliente". (Fuente: elaboración propia)

Flujos	Var 1. Nombre	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Deshabilitar cliente.	V	Deshabilita el usuario.	Satisfactoria	
	I	Muestra el mensaje "Se debe seleccionar al menos un elemento"	Satisfactoria	

Tabla 28: Iteraciones de las variables del caso de uso "Deshabilitar cliente". (Fuente: elaboración propia)

Eliminar cliente

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo seleccionable	No	Debe seleccionar un valor

Tabla 29: Descripción de las variables del caso de uso "Eliminar cliente". (Elaboración propia)

Flujos	Var 1. Nombre	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Habilitar cliente.	V	Borrar el usuario.	Satisfactoria	
	I	Muestra el mensaje "Se debe seleccionar al menos un elemento" o "El cliente no se puede eliminar porque tiene solicitudes pendientes."	Satisfactoria	

Tabla 30: Iteraciones de las variables del caso de uso "Eliminar cliente". (Fuente: elaboración propia)

Habilitar cliente

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo seleccionable	No	Debe seleccionar un valor

Tabla 31: Descripción de las variables del caso de uso "Habilitar cliente". (Fuente: elaboración propia)

Flujos	Var 1. Nombre	Respuesta del Sistema	Resultado de la Prueba	Observaciones

Flujos	Var 1. Nombre	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Habilitar cliente.	V	Habilita el usuario.	Satisfactoria	
	I	Muestra el mensaje “Se debe seleccionar al menos un elemento”	Satisfactoria	

Tabla 32: Iteraciones de las variables del caso de uso "Habilitar cliente". (Fuente: elaboración propia)

Recuperar contraseña

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Correo electrónico	Campo de texto	No	Debe introducir un texto.
[2]	Captcha	Campo de texto	No	Debe introducir un texto.
[3]	Contraseña	Campo de texto	No	Debe introducir un texto.
[4]	Repetir contraseña	Campo de texto	No	Debe introducir un texto.

Tabla 33: Descripción de las variables del caso de uso "Recuperar contraseña". (Fuente: elaboración propia)

Flujos	Var 1. Correo electrónico	Var 2. Captcha	Var 3. Contraseña	Var 4. Repetir contraseña	Respuesta del Sistema	Resultado de la Prueba	Observaciones
Recuperar contraseña.	V	V	V	V	Recupera la contraseña.	Satisfactoria	
	I	V	V	V	Muestra el mensaje “Debe entrar una dirección de correo” o “La dirección de correo no es válida”	Satisfactoria	
	V	I	V	V	Muestra el mensaje “Incorrecto”	Satisfactoria	
	V	V	I	V	Muestra el mensaje” Debe entrar su contraseña” o “La contraseña es muy pequeña” o “Las dos contraseñas deben ser iguales.”	Satisfactoria	

Flujos	Var 1. Correo electrónico	Var 2. Captcha	Var 3. Contraseña	Var 4. Repetir contraseña	Respuesta del Sistema	Resultado de la Prueba	Observaciones
	V	V	V	I	Muestra el mensaje" Debe entrar su contraseña" o "La contraseña es muy pequeña" o "Las dos contraseñas deben ser iguales."	Satisfactoria	

Tabla 34: Iteraciones de las variables del caso de uso "Recuperar contraseña". (Fuente: elaboración propia)

Anexo 6: Acta de liberación de productos software

Acta de Liberación de Productos Software

Fecha de liberación: 1ro de abril de 2011.

Emitida a favor de: Integración con Bancos.

1. Datos del producto

Artefactos	Versión	Estado final	Cantidad de iteraciones	Tipos de pruebas realizadas
Módulo Integración Bancos. Aplicación Web.	V 1.0	2 NC	3	Pruebas Funcionales
Módulo Integración Bancos. Gestión de Comunicación Bancaria.	V 1.0	0 NC	3	Pruebas Funcionales
Módulo Integración Bancos. Servicio de Actualización de Pagos.	V 1.0	0 NC	3	Pruebas Funcionales

Anexo:

Se recomienda se tenga en cuenta para próximas etapas de prueba lo siguiente:

1. La versión 1.0 del SIB se desarrolló a partir de la versión 1.3 de los Requisitos Funcionales. A medida que se iba ampliando el negocio, optimizándose los procesos y discutiendo el documento fue preciso tomar decisiones que tenían de cierta forma impacto en los RF. A continuación se exponen los elementos fundamentales que influyeron en la situación antes descrita:

- Se capturaron en condiciones adversas: (poca atención de la contrapartida, 1 solo especialista, la contrapartida desinteresada etc.).
- Cambios reiterados del cliente.
- No hubo proceso de refinamiento de requerimientos.
- Contradicciones detectadas por los analistas del equipo de desarrollo.
- Es necesaria una segunda iteración de los requerimientos para cumplir con las necesidades de SAIME.

- La comunicación entre las Aplicaciones de Escritorio y la Aplicación web no se establece hasta que se integren con los Bancos Recaudadores, por lo que no se pudo hacer una prueba de esta integración.

NC Pendientes	Respuesta del Equipo de Desarrollo
Aplicación web: Cuando la aplicación se carga en el navegador web Chromium, aparece un error de interfaz con algunas imágenes corridas. Esto sucede con los botones Guardar y Guardar y crear otro. No se garantiza la compatibilidad con otros navegadores.	Esta anomalía es producto de la implementación o no por parte del navegador de algunos elementos de los usados en los CSS, por lo que se debe especificar en el documento a los clientes el uso de navegadores que respeten los estándares, ejemplo: Firefox.
Administrar Servicio: Cuando dos usuarios concurrentes están realizando una acción sobre un mismo objeto y este se elimina por uno de ellos y posteriormente sin actualizar el listado el otro usuario lo elimina, el error que se muestra al último usuario que intenta eliminarlo no indica que el mismo ya no existe en el sistema.	El error mostrado es el error estándar de las aplicaciones para cuando una petición al sitio de una URL no existe, que es lo que sucede en este caso al usuario no refrescar la página. El proceso de validación de este tipo de errores forma parte de una iteración posterior del desarrollo de la aplicación, por lo que en un primer momento se pone la validación básica, y la validación más profunda y específica se deja para futuras iteraciones del producto.

_____ _____

Alionuska Velázquez Cintra
Responsable Calisoft

Johann Rodríguez Hernández
Responsable Proyecto

Tayché Capote García

Firmado digitalmente por Tayché Capote García
Nombre de reconocimiento (DN): 0.9.2342.19200300.100.1.1=tcapote, cn=Tayché Capote García, serialNumber=14320, givenName=Tayché, sn=Capote García, c=CU
Fecha: 2011.04.05 10:13:54 -04'00'