Universidad de las Ciencias Informáticas
Facultad 1

Sistema de Integración Bancaria, desarrollo de los módulos Autenticación, Solicitudes y Pagos

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

Autor
Isidro Martínez Suárez

Tutores
Ing. Johan Rodríguez Hernández
Ing. Noichel Juan Hernández

Julio de 2011
La Habana, Cuba

Declaración de autoría

Declaro que soy el único autor del trabajo titulado “Sistema de Integración Bancaria, desarrollo de los módulos Autenticación, Solicitudes y Pagos”, y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente declaración a los ___ días del mes de _____ del año 2011.

_____

Isidro Martínez Suárez

Autor

_____

Ing. Noichel Juan Hernández

Tutor

_____

Ing. Johan Rodríguez Hernández

Tutor

Agradecimientos

A la Revolución cubana y su líder indiscutible Fidel Castro Ruz, por permitirnos ser parte de este gran colectivo.

A la Universidad de las Ciencias Informáticas…
A mis padres, por tanta dedicación y sacrificio…
A mi Hermano, por creer siempre en mí…
A mi novia, por apoyarme en todo momento, más malos que buenos…
A Nilda y Angelito por darme siempre su apoyo incondicional…
A Yoandra, El Kerendon y El Yoan, por ser más que amigos…
A mi tutor, profesor y amigo Noichel, si no fuera por ti, esto no fuera posible…
A todos los que influyeron en mi formación personal y profesional…
Gracias a TODOS…

Dedicatoria

A mis padres queridos que me han enseñado siempre el camino correcto a seguir para poder ser una mejor persona cada día. Este triunfo también es de ustedes.

A mi abuela Aleida mi segunda mamá ya te voy a poder comprar tu vestidito rojo.

A mi Abuela Mimi lejos pero siempre presente.

A mi Abuelo Papi que aunque la vida no te haya dejado ver este sueño hacerse realidad, se que desde algún lugar estas disfrutando como tú sabes hacerlo y cuidándome como siempre lo hiciste desde los primeros días de mi vida.

A mi novia que siempre estuvo conmigo dándome esa fuerza cuando casi no quedaba ninguna, eres especial, este sueño hecho realidad también es tuyo.

A mi hermano el DJ Young del reggaetón ahora te toca a ti.

A mis suegros Nilda y Angelito mejores no podían ser.

A Maiken y familia lejos pero siempre presentes, sin tu ayuda incondicional muchas cosas faltaran en este momento.

A todos mis amigos y compañeros de la uci y de la calle que siempre han estado en todo momento.

A Onnier, Antonio P., Antonio T., El Guille, Reinier, Raidel, El Leonis, Uffo, El fino y Angelito el piquete de siempre…

Resumen

En aras de llevar a cabo el acuerdo de colaboración entre Cuba y la República Bolivariana de Venezuela firmado en el marco de la ALBA[1]surge el proyecto Sistema Integración Bancaria, el cual cuenta con tres subsistemas (Solicitudes y Pagos, Comunicación Bancaria y Actualización de pagos) para poder realizar el proceso de recaudación de pagos del SAIME con una mejor calidad.

En el presente trabajo se realiza la implementación de los módulos “Autenticación, Solicitudes y Pagos” del subsistema de Solicitudes y Pagos, los cuales se encargan de mantener la seguridad del subsistema, darle la posibilidad al usuario de gestionar sus solicitudes de servicio (creándolas, modificándolas o eliminándolas) y registrar los pagos de las solicitudes de servicios realizadas.

Para todo esto se realizó el estudio de sistemas similares, se estudió la metodología y las tecnologías de libre de costo. Definiendo por el proyecto utilizar como metodología de desarrollo RUP, *Visual Paradigm* como herramienta Case para el modelado, *Symfony* como *framework* de desarrollo, como sistema gestor de base datos PostgreSQL y PHP como lenguaje de programación.

Después de realizar todo este estudio se elaboró una propuesta de solución de acuerdo con la situación problémica detectada. Realizando el análisis y diseño para llegar a cabo una mejor implementación donde se generaron artefactos como: el diagrama de despliegue y el diagrama de componente y finalmente la validación de la solución, mediante las pruebas de conciliación de caja negra.

[1] Alternativa Bolivariana para las Américas

Índice de contenidos

Introducción

El vertiginoso desarrollo de la informática y las comunicaciones en las últimas cuatro décadas, ha propiciado que la industria mundial del software sea increíblemente competitiva. Hoy, un producto no es bueno solo porque solucione un problema, sino porque es único, necesario y se distingue por su calidad.

En el marco de la Alternativa Bolivariana para las Américas (ALBA), han sido firmados entre Cuba y la República Bolivariana de Venezuela una serie de acuerdos para la realización de múltiples proyectos con el objetivo de afianzar el desarrollo social conjunto y de las diferentes esferas productivas de ambos países. En este sentido, la mayoría de las entidades y organizaciones gubernamentales de ambas partes, se han beneficiado producto de estos proyectos.

El Servicio Administrativo de Identificación, Migración y Extranjería (SAIME), es una institución del gobierno venezolano que se encuentra a cargo de los servicios del registro civil. Dicha institución se dedica oficialmente a la emisión de pasaportes, tarjetas de identidad y otros documentos jurídicos de los ciudadanos venezolanos, así como extranjeros residentes en el país, y ha sido pionero en revolucionar la imagen corporativa gracias, en gran parte, a los sistemas informáticos desarrollados.

Con el paso del tiempo los procesos cambian y surgen nuevas demandas de los sistemas informáticos, con el objetivo de hacer los procesos más eficientes y seguros. Una de las necesidades surgidas a raíz d estos cambios, es la recaudación de pagos de SAIME en los bancos de la República Bolivariana de Venezuela, debido a que muchos de los procesos que se llevan a cabo, se realizan de forma manual y esto resulta poco fiable. Esto provoca a su vez, una serie de inconvenientes tanto para los clientes como para los trabajadores del SAIME, debido a la cantidad de procesos a realizar, la cantidad de datos involucrados, así como la sensibilidad de la información manejada en estos procesos.

Por lo que la dirección de gestión administrativa del SAIME, requiere de un sistema que garantice la fiabilidad y seguridad de la recaudación de pagos efectuados por los clientes en las sucursales de sus bancos recaudadores ya que:

- No hay una manera efectiva de comprobar la legitimidad del voucher presentado por el cliente, provocando inseguridad en el proceso de conciliación de pago.

- No hay una manera efectiva de comprobar que el pago realizado por el cliente y el costo de los servicios solicitados es el mismo.

Por las insuficiencias planteadas se formula el siguiente **problema científico de la investigación:** ¿Cómo garantizar la autenticación, las solicitudes y los pagos en el subsistema de gestión "Solicitudes y pagos"?; estableciendo como **objeto de estudio** los procesos de autenticación, solicitudes y pagos mediante medios electrónicos.

Para dar solución a la situación problémica descrita anteriormente, se plantea como **objetivo de la investigación:** Implementar los módulos Autenticación, Solicitudes y Pagos como parte del Sistema de Integración Bancaria (SIB).

El **campo de acción** queda enmarcado en los procesos de solicitud de servicios y pagos, en el Servicio Administrativo de Identificación, Migración y Extranjería (SAIME).

Para dar cumplimiento al objetivo de la investigación se derivan los **objetivos específicos:**

- Analizar y diseñarlos módulos de autenticación, solicitudes y pagos del subsistema "Solicitudes y pagos" perteneciente al SIB.
- Implementar los módulos de autenticación, solicitudes y pagos del subsistema "Solicitudes y pagos" perteneciente al SIB.
- Realizar las pruebas de liberación requeridas de los módulos de autenticación solicitudes y pagos, del subsistema "Solicitudes y pagos" perteneciente al SIB.

Conociendo lo anteriormente reflejado se plantea la siguiente **hipótesis:** Si se implementan los módulos de Autenticación, Solicitudes y Pagos del subsistema "Solicitudes y pagos", se logrará mejorar el proceso de notificación de pago bancario entre los bancos recaudadores y clientes venezolanos con el SAIME.

Variables:

- Independiente: Implementación de los módulos Autenticación, Solicitudes y Pagos.
- Dependiente: Proceso de notificación de pagos bancarios.

Tabla 1: Operacionalización de las variables (Elaboración propia).

Variable	Dimensión	Indicadores
Implementación de los módulos Autenticación, Solicitudes y Pagos	Chequeo de datos	Cantidad de no conformidades: • De 11 a 20 - Mal • De 6 a 10 - Regular • De 0 a 5 - Bien
Proceso de notificación de pagos bancarios	Chequeo de datos	Cantidad de no conformidades: • De 11 a 20 - Mal • De 6 a 10 - Regular • De 0 a 5 - Bien

Para darle cumplimiento a los objetivos trazados se decide desarrollar las siguientes **tareas de la investigación científica:**

- Elaboración del diseño teórico de la investigación.
- Realización del estudio de antecedentes y estado del arte de la investigación.
- Identificación de los requisitos funcionales y no funcionales.
- Análisis y diseño de los módulos autenticación, solicitudes y pagos.
- Implementación de los casos de usos.
- Diseño y realización de pruebas de calidad al sistema.

Se utilizaron como **métodos de investigación científica:**

Métodos teóricos:

- Analítico - Sintético: se consultaron y estudiaron varias bibliografías para definir una solución informática. Realizando un resumen de los aspectos más importantes.
- Modelación: se realizó un modelo para definir cómo quedarán establecidos los procesos que se automatizarán.

Métodos empíricos:

- Observación: se realizó un estudio del proceso de recaudación de pagos de SAIME.
- Estudio documental: Se empleó en la consulta de documentos, entendiéndose este término, en sentido amplio, como todo material de índole permanente, es decir, al que se puede acudir como

fuente o referencia en cualquier momento o lugar, sin que se altere su naturaleza o sentido, para que aporte información o rinda cuentas de una realidad o acontecimiento a nuestro trabajo.

- Medición: se realizó la medición de las variables registrando los resultados obtenidos.

Resultados esperados

Con la elaboración de este trabajo de diploma se esperan los siguientes resultados:

- Sistema informático que permitirá mejorar el sistema de notificaciones de pagos del SAIME con los clientes y bancos recaudadores.
- Se contará una forma efectiva de comprobar la legitimidad de voucher presentado por el cliente brindando seguridad a la hora de conciliar los pagos.

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 los sistemas similares y tecnologías que existen en la actualidad. Se define la metodología, herramientas y tecnologías a utilizar para desarrollar la solución.

Capítulo 2 Propuesta de solución: se expone el proceso a informatizar brindando posteriormente la propuesta de solución, se especifican los requisitos del sistema, identificando los requerimientos funcionales y no funcionales y se identifican los casos de uso del sistema. Se presentan las fases de Análisis y Diseño definidas por la metodología establecida para dar solución al problema científico.

Capítulo 3 Implementación: se define el diagrama de despliegue y se describe cómo quedará el sistema una vez que haya sido desplegado. Se realizan además los diagramas de componentes, mostrando la forma en la que se desarrolló la aplicación.

Capítulo 4 Prueba: se realizan las pruebas funcionales del sistema para comprobar la eficiencia de las clases y operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se abordan los principales conceptos que son utilizados en el transcurso de la investigación. Se realiza un análisis sobre la actualidad del problema y de las soluciones existentes del entorno del tema a investigar; así como un estudio que incluye varias de las tecnologías actuales para el desarrollo de software, las principales herramientas y las metodologías que serán empleadas en la implementación del sistema que se pretende desarrollar.

1.2 Conceptos asociados al dominio del problema

Son aquellos conceptos asociados a la fundamentación teórica, donde se realizó un estudio sobre las formas de autenticación existentes en la actualidad, los sistemas de pagos electrónicos y las solicitudes de servicio, así como los sistemas existentes que utilizan estas operaciones para su beneficio.

1.2.1 ¿Qué es una autenticación?

La autenticación es uno de los requerimientos primordiales de los sistemas informáticos que desempeñan tareas importantes, debido que son los mecanismos de seguridad adecuados a la información que se intenta proteger. Este conjunto de mecanismos ha de incluir al menos, un sistema que permita identificar a las entidades, las cuales no son más que los elementos activos del sistema, generalmente usuarios, que intentan acceder al sistema mediante procesos tan simples como una contraseña o tan complejos como un dispositivo analizador de patrones retínales. (1)

Actualmente, entre las principales formas que se utilizan para identificar a una persona se encuentran, el aspecto físico o la manera de hablar, las cuales son operaciones demasiado complejas para una computadora, debido a que el objetivo de los sistemas de identificación de usuarios[2] no consisten en identificar a una persona, sino verificar que la misma, es quien dice ser realmente. Los métodos de autenticación se suelen dividir en tres grandes categorías, las cuales están en función de la forma en que se utilizan para la verificación de la identidad:

[2]Es un usuario el individuo que utiliza o trabaja con algún objeto o dispositivo o que usa algún servicio en particular.

- Algo que el usuario sabe o conoce y el resto de las personas no Ej. Un *password*[3].
- Algo que el usuario posee o lleva consigo Ej. Una tarjeta de identidad.
- Una característica física del usuario o un acato involuntario del mismo, esta última categoría se conoce con el nombre de autenticación biométrica Ej. La huella dactilar que no es más que una característica física del usuario y la firma que podría considerarse como un acto involuntario debido a que para rubricar la firma no se piensa en el diseño de cada trazo, se realiza de forma involuntaria Esta última categoría se conoce con el nombre de autenticación biométrica.

1.2.2 Sistemas de autenticación

Un sistema de autenticación se basa en combinar diferentes métodos de autenticación, como es el caso de las tarjetas de crédito, donde se combina el método uno con el dos (algo que el usuario lleva consigo, que no es más que la tarjeta y algo que el usuario sabe o conoce que no es más que el ping de la tarjeta).

También debe poseer determinadas características para ser fiable, soportando con éxito cualquier tipo de ataque provocado por cualquier agente externo. Ser económicamente factible para la organización que necesite de su seguridad e incluir dentro de estas características, una no técnica sino humana siendo quizás la más importante: un sistema de autenticación ha de ser aceptable para los usuarios, que serán al final quienes lo utilicen.

Ejemplo de esto sería imaginariamente un potencial sistema de identificación para acceder a los recursos de la Universidad, el cual consistiera en un dispositivo que fuera capaz de realizar un análisis de sangre a un usuario y así comprobar que es quien dice ser; seguramente sería barato y altamente fiable, pero nadie aceptaría dar un poco de sangre cada vez que desee consultar el correo.

1.2.3 Sistemas basados en algo conocido: contraseñas

El modelo de autenticación más básico consiste en decidir si un usuario es quien dice ser, simplemente basándonos en una prueba de conocimiento que sólo ese usuario puede superar. Ya desde Alí Babá y su "*Ábrete, Sésamo"* hasta los más modernos sistemas Unix, esa prueba de conocimiento no es más que una contraseña que en principio es secreta. Evidentemente, esta aproximación es la más vulnerable a todo tipo de ataques, pero también la más barata, por lo que se convierte en la técnica más utilizada en entornos que no precisan de una alta seguridad. También se utiliza como complemento a otros

[3] Es una serie secreta de caracteres que permite a un usuario tener acceso a un archivo, a un ordenador, o a un programa.

mecanismos de autenticación, por ejemplo en el caso del Número de Identificación Personal (PIN) a la hora de utilizar cajeros automáticos. (2)

En todos los esquemas de autenticación basados en contraseñas se cumple el mismo protocolo: las entidades que participan en la autenticación acuerdan una clave, la cual han de mantener en secreto si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante otra, se limita a mostrarle su conocimiento de esa clave común y si ésta es correcta, se otorga el acceso a un recurso. Lo habitual es que existan unos roles preestablecidos, con una entidad activa que desea autenticarse y otra pasiva que admite o rechaza a la anterior. (2)

Este esquema es muy frágil, pues basta con que una de las partes no mantenga la contraseña en secreto para que toda la seguridad del modelo se pierda. Por ejemplo, si el usuario de una máquina comparte su clave con un tercero, o si ese tercero consigue leerla y rompe su cifrado, automáticamente esa persona puede autenticarse ante el sistema con éxito con la identidad de un usuario que no le corresponde. (2)

1.2.4 Sistemas basados en algo poseído: tarjetas inteligentes

Una tarjeta inteligente (o *smartcard*) es un dispositivo de seguridad del tamaño de una tarjeta de crédito, resistente a la adulteración, que ofrece funciones para un almacenamiento seguro de información. En la práctica, las tarjetas inteligentes poseen un chip empotrado en la propia tarjeta que puede implementar un sistema de ficheros cifrado y funciones criptográficas, y puede además detectar activamente intentos no válidos de acceso a la información almacenada. Este chip inteligente es el que las diferencia de las simples tarjetas de crédito, que solamente incorporan una banda magnética donde va almacenada cierta información del propietario de la tarjeta. (2)

Cuando el usuario poseedor de una *smartcard* desea autenticarse, necesita introducir la tarjeta en un *hardware* lector; los dos dispositivos se identifican entre sí con un protocolo a dos bandas en el que es necesario que ambos conozcan la misma clave (*Company Key* o *Chipcard Communication Key*), lo que elimina la posibilidad de utilizar tarjetas de terceros para autenticarse ante el lector de una determinada compañía; además esta clave puede utilizarse para asegurar la comunicación entre la tarjeta y el dispositivo lector. Tras identificarse las dos partes, se lee la identificación personal de la tarjeta, y el usuario teclea su PIN. Se inicia entonces un protocolo desafío-respuesta: se envía la identificación personal a la máquina y ésta desafía a la tarjeta, que responde al desafío utilizando una clave personal del

usuario (*Personal Key*). Si la respuesta es correcta, el *host* ha identificado la tarjeta y el usuario obtiene acceso al recurso deseado. (2)

Las ventajas de utilizar tarjetas inteligentes como medio para autenticar usuarios son muchas frente a las desventajas, pues se trata de un modelo ampliamente aceptado entre los usuarios, rápido y que incorpora *hardware* de alta seguridad tanto para almacenar datos, así como para realizar funciones de cifrado.
Además, su uso es factible tanto para controles de acceso físico como para controles de acceso lógico a los *hosts*, y se integra fácilmente con otros mecanismos de autenticación como las contraseñas. Como principal inconveniente de las *smartcards*, podemos citar el coste adicional que supone para una organización el comprar y configurar la infraestructura de dispositivos lectores y las propias tarjetas. Además, que un usuario pierda su tarjeta es bastante fácil, y durante el tiempo que no disponga de ella no podrá acceder al sistema. (2)

1.2.5 Sistemas de autenticación biométrica

A pesar de la importancia de la criptología en cualquiera de los sistemas de identificación de usuarios, existen otra clase de sistemas en los que no se aplica esta ciencia, o al menos su aplicación es secundaria. Es más, parece que en un futuro no muy lejano, estos serán los sistemas que se van a imponer en la mayoría de situaciones en las que se haga necesario autenticar un usuario, pues resultan más amigables para el usuario, ya que no necesitará recordar *passwords* o números de identificación complejos y, como se suele decir, el usuario puede olvidar una tarjeta de identificación en casa, pero nunca se olvidará de su mano o su ojo. Igualmente son mucho más difíciles de falsificar que una simple contraseña o una tarjeta magnética y la principal razón por la que no se han impuesto ya en nuestros días es su elevado precio, fuera del alcance de muchas organizaciones, y su dificultad de mantenimiento. (2)

Estos sistemas son los denominados **biométricos**, basados en características físicas o conductuales del usuario a identificar. El reconocimiento de formas, la inteligencia artificial y el aprendizaje son las ramas de la informática que desempeñan el papel más importante en los sistemas de identificación biométricos. (2)

Aunque la autenticación de usuarios mediante métodos biométricos es posible utilizando cualquier característica única y mesurable del individuo, tradicionalmente ha estado basada en cinco grandes grupos:

- Verificación de voz.

- Verificación de escritura.
- Verificación de huellas.
- Verificación de patrones oculares.
- Verificación de la geometría de la mano.

1.2.6 ¿Qué es un pago?

El pago es uno de los modos de extinguir las obligaciones que consisten en el cumplimiento efectivo de la prestación debida (3); ejemplo de esto: el usuario efectúa el pago por el servicio brindado. Actualmente existen diferentes formas de pago, gracias al avance de las tecnologías ya casi no es necesario para efectuar un pago contar con el dinero en efectivo o dirigirse al lugar. Ahora usted puede realizar las compras desde su casa, pagar la cuenta de electricidad desde su casa, así como otras acciones, un ejemplo de estas nuevas formas, es el pago electrónico.

1.2.7 Sistemas de pago electrónico

El comercio electrónico, también conocido como **e-commerce** (*electroniccommerce* en inglés), consiste en la compra y venta de productos o de servicios a través de medios electrónicos, tales como Internet[4] y otras redes informáticas. Originalmente el término se aplicaba a la realización de transacciones mediante medios electrónicos tales como el intercambio electrónico de datos. Sin embargo con el advenimiento de la Internet a mediados de los años 90, comenzó a referirse principalmente a la venta de bienes y servicios a través de Internet, usando como forma de pago medios electrónicos tales como las tarjetas de crédito.

El comercio electrónico por Internet se ofrece como un nuevo canal de distribución sencillo, económico y con alcance mundial las 24 horas del día todos los días del año, y esto sin los gastos y limitaciones de una tienda clásica: personal, local, horario, infraestructura, etc.

Los sistemas de pago electrónico realizan la transferencia de dinero entre comprador y vendedor en una compra-venta electrónica. Es por ello, una pieza fundamental en el proceso de compra-venta dentro del comercio electrónico. Como ejemplos de sistemas de pago electrónico se pueden citar las pasarelas de

[4] Es una red de redes que permite la interconexión descentralizada de computadoras a través de un conjunto de protocolos denominado **TCP/IP**.

pago o TPV-virtual para el pago con tarjeta y los sistemas de monedero electrónico. A continuación se brinda una breve explicación sobre cada uno de ellos.

TPV: Terminal Punto de Venta virtual, cumple en Internet la misma función que los sistemas tradicionales de cobro mediante tarjeta de crédito (TPV físico), permitiendo que los clientes puedan pagar su compra a través de Internet utilizando una tarjeta de crédito. (4)

La tarjeta monedero: llamado también monedero electrónico, ha sido el sistema de micropagos multipropósito más eficiente. Ha servido para aquellas transacciones de bajo monto y alto volumen que requieren gran velocidad y seguridad. El sistema ha permitido a los usuarios pagar más rápido que con efectivo y las transacciones se llevan en menos tiempo. Algunas de sus principales ventajas son:

- Se evita las filas y la congestión en los puntos de pagos.
- Se gana más tiempo.
- Es más fácil, rápido y práctico.
- Da mayor seguridad y control de los pequeños gastos de todos los días.
- Se elimina los problemas del cambio y la falta de monedas.

La tarjeta monedero contiene un chip electrónico que se recarga con dinero y permite pagar productos y servicios deslizando la tarjeta en el lector cuyo importe exacto se descargará o acreditará del chip. Se puede recargar en cualquier establecimiento que tenga una terminal para recibir el pago. También se puede recargar con tarjeta de débito o crédito.

1.2.8 Solicitudes

Por lo general, una solicitud (o instancia) es un documento escrito que va dirigido a un organismo público o a una autoridad, a los que se pide algo o ante los que se plantea una reclamación con la exposición de los motivos en los que se basan. La mayor parte de los organismos públicos disponen de impresos destinados a este fin. Se debe preguntar por su existencia y, si no los hay, se puede hacer la instancia por uno mismo. (5)

1.2.9 ¿Qué es un servicio?

Es la relación que se establece entre dos partes, en la que una (proveedor) organizada sistémicamente, ofrece a la otra (usuario) los resultados de la también relación que se produce entre los recursos con que cuenta este proveedor, tanto financieros como físicos o tecnológicos pero, fundamentalmente, los de información y los humanos, estos últimos, adiestrados previamente en la gestión de todos los anteriores. En todas estas relaciones median procesos de comunicación e informacionales en general y se llevan a cabo con el objetivo de satisfacer las necesidades de información tanto implícitas como explícitas de la parte receptora cuyo conocimiento por el proveedor también está implícito en estas relaciones las que pudiendo ser lucrativas o no, en última instancia están encaminadas a, con sus resultados, elevar el nivel de aptitud de la parte usuaria, individual o colectiva, para abordar problemas tanto personales como sociales. (6)

1.2.10 Solicitudes de servicio

Es aquella petición que se establece entre dos partes, proveedor y usuario, donde la primera parte le brinda al usuario los servicios con los que cuenta para que este interactúe con ellos solicitando los que desee. Con el avance de las tecnologías, ha cambiado la forma de realizarse estas solicitudes, brindándole más comodidad al solicitante, en este caso el usuario, ya que a través de Internet se pueden realizar estas solicitudes a larga distancia sin tener que ver directamente al proveedor.

Las aplicaciones web soportan un gran peso en todo este negocio, ya que interactúa con ambas partes (usuario y proveedor) como la interfaz visual o intermediario, mostrándole al usuario los servicios que se brindan para que este solicite el o los servicios deseados.

1.2.11 Sistemas similares

Se realizó un estudio de sistemas similares existentes, con el objetivo de analizar la forma en que se realiza el proceso de solicitudes y pagos en determinados sitios o empresas. Evaluando sus características fundamentales para a partir de este análisis identificar posibles funcionalidades que se deseen incluir o evitar en el sistema a desarrollar.

Sisproweb

Sisproweb (Sistema y Proyectos Web). Sitio venezolano, el cual ofrece servicios de Web Hosting, Diseño y Programación de Páginas Web, elaboración de Multimedia para Empresas y entidades particulares entre otras.

Para la obtención de estos servicios la aplicación brinda un sistema de solicitud de servicios y registro de pagos en el cual el usuario para adquirir el servicio deseado, debe seleccionar de una lista de opciones el servicio que desea. Una vez realizada esta operación, para registrar el pago del mismo introduce los datos especificados en el formulario “Registrar Pagos” siguiendo las instrucciones reflejadas para alcanzar la máxima seguridad.

Figura 1: Formulario "Registrar Pagos" de Sisproweb (fuente: Web oficial Siproweb).

ANTAG

Sistema que se encarga de buscar todo tipo de compras en internet y mostrarlas al usuario, en este caso se encuentra la oferta de muebles y artículos de oficina. Para realizar la compra de cualquier artículo deseado por el usuario, este sistema brinda entre sus sistema de pago el monedero electrónico, actualmente la tecnología más segura en transacciones financieras. Esta tecnología a diferencia de la tarjeta de crédito, no requiere que se verifique con el banco si existen fondos o no, pues el chip incorporado al monedero establece directamente esta información, haciendo más ágil y seguro el trámite.

Figura 2: Web oficial sistema ANTAG (fuente: Web oficial ANTAG).

Ultra Gas

El sistema de tarjetas de Ultra Gas, permite administrar y controlar el combustible a profesionales y a grandes, medianas y pequeñas empresas. Este sistema se encarga de brindarle la información al cliente de todos sus saldos y movimientos, editar las restricciones y los servicios de su tarjetas, así como también consultar sus estados de cuenta, realizando además el pago del servicio brindado a través de cheques o transferencias bancarias donde una vez identificado el pago realizado, se le aplica el saldo a la tarjeta *Ultra Gas Control Card*, y se le entregará la factura solo de lo que haya consumido del combustible.

Figura 3: Sistema de Ultra Gas (fuente: Web oficial Ultra Gas).

Estos sistemas anteriormente estudiados, permitieron entender la lógica del negocio de los sistemas de gestión de solicitudes de servicio y registro de pagos. La mayoría de estos sistemas, están elaborados con herramientas y tecnologías propietarias, lo que provocan algunas desventajas ya que elevaría el costo de la aplicación por la utilización de estos *software* propietarios. También algunos de los sistemas de pago utilizados en estos sistemas son relativamente caros, haciendo que solo una cierta cantidad de personas con posibilidades tenga acceso a este servicio, lo cual también se convierte en una desventaja ya que se quiere que el sistema sea utilizado por la mayoría de los ciudadanos venezolanos, lográndose un alto impacto social.

1.2.12 Aplicaciones web

Son aplicaciones de software que permiten al usuario acceder a un servidor web a través de internet, se codifica en un lenguaje soportado por los navegadores web que son los encargados de ejecutarlas. 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.

Algunas de las ventajas que ofrecen las aplicaciones web sobre las aplicaciones de escritorio por lo que se recomienda su uso son, que no es necesario descargar o instalar algún programa para realizar tareas sencillas por lo que ahorra tiempo, pueden ejecutarse en cualquier navegador eliminado los problemas de compatibilidad, 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 y ofrecen una gran colaboración entre las organizaciones ya que es sencillo el acceso y compartición de los datos.

1.3 Metodologías y estándares para el desarrollo de software

Existen dos tipos de metodologías para el desarrollo de software, las metodologías 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 cuyos 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 del Rational (RUP) como metodología tradicional para el desarrollo de la aplicación.

1.3.1 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, también posee cinco 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*. (6)

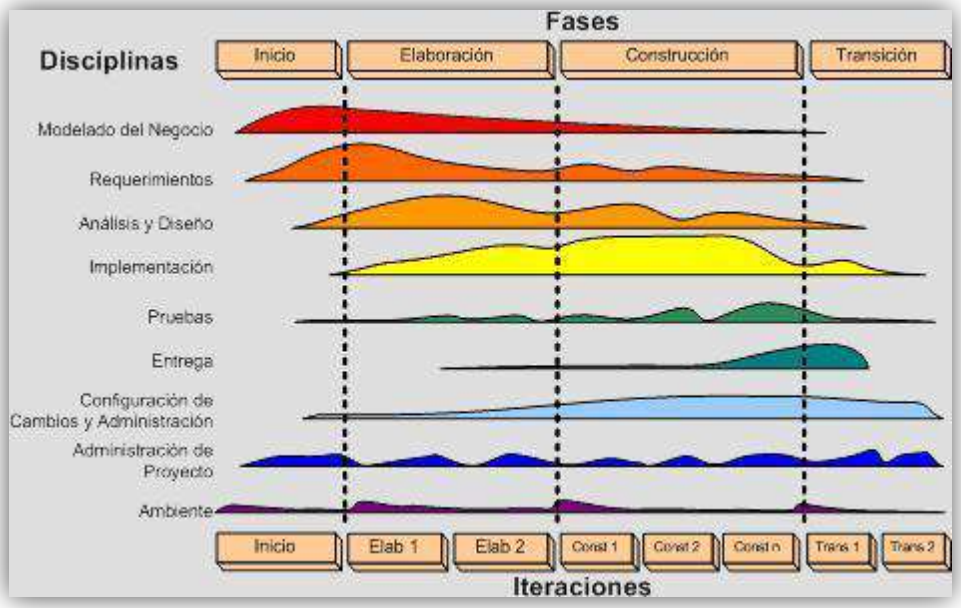


Figura 4: RUP en dos dimensiones (fuente: EVA).

1.3.2 El Lenguaje Unificado de Modelado (UML)

UML (del inglés *Unified Modeling Language)*, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y

documentar un sistema informático. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso utilizar.

1.4 Selección de las herramientas y lenguajes de desarrollo

A continuación de describen las herramientas de desarrollo y lenguajes de de programación a utilizar en la solución que se desea obtener, haciéndose énfasis en las principales características de los mismos, en aras de fundamentar su selección.

1.4.1 Lenguaje de programación del lado del servidor

PHP[5]: lenguaje de programación utilizado para la creación de sitios o aplicaciones web. PHP es un lenguaje de programación interpretado en el lado del servidor, utilizado para la generación de páginas web dinámicas y ejecutadas en un servidor web. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o *Internet Information Server* (IIS) con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Sus principales ventajas son: (7)

- Soporta una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, Microsoft SQL Server[6], Informix[7], entre otras.
- Ofrece una solución simple y universal para las paginaciones dinámicas de la web.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Con PHP se puede hacer cualquier cosa que podamos realizar con un *script* de Interfaz de Entrada Común (CGI), como el procesamiento de información en formularios, foros de discusión, manipulación de *cookies* y páginas dinámicas.

[5] Inicialmente se llamó *Personal Home Page*. Surgió en 1995, desarrollado por PHP Group.
[6] Es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional.
[7] Fue concebido y diseñado por Roger Sippl a finales de los años 1970. La compañía Informix fue fundada en 1980 y durante parte de los años 1990 fue el segundo sistema de bases de datos más popular después de Oracle.

- PHP corre en cualquier plataforma utilizando el mismo código fuente.
- Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo Apache, IIS y AOLServer[8].
- Con respecto a la rapidez, PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.
- Es capaz de leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Tiene la capacidad de expandir su potencial utilizando su enorme cantidad de módulos.
- Posee una amplia documentación en su web oficial de Internet.
- Pertenece a la alternativa de código abierto.
- Permite las técnicas de programación orientada a objetos.

1.4.2 Marcos de trabajo

En el desarrollo de software, un marco de trabajo es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un marco de trabajo puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (8)

Un marco de trabajo simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un marco de trabajo proporciona estructura al código fuente, donde el desarrollador es obligado a crear código más legible y más fácil de mantener, lo cual permite la programación de aplicaciones más robustas, ya que encapsula operaciones complejas en instrucciones sencillas.

1.4.2.1 Symfony 1.4.6

Symfony es un marco de trabajo que tiene como objetivo fundamental automatizar los patrones más utilizados en la elaboración de sistemas web; además de obligar a clarificar a los programadores el código, establece un estándar de código legible, encapsula operaciones complejas en simples líneas de código, ahorrando mucho más tiempo a la hora de mostrar datos directamente de la base de datos. (9)

[8] Servidor web de código abierto.

Este marco de trabajo está implementado bajo el lenguaje PHP 5, y ha sido utilizado en varias aplicaciones web obteniéndose resultados satisfactorios. Es compatible con la mayoría de los gestores de base de datos existentes, dígase MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. También otorga facilidades para su utilización en diferentes plataformas. (9)

Symfony provee una abstracción de la base de datos, que permite una mayor facilidad de obtención de los datos, haciendo a la vista y a las acciones independientes del gestor de base de datos. La vista posee *helpers*[9]que facilitan el trabajo para los diseñadores con el código HTML de la aplicación. (9)

Características particulares de Symfony:

- Escalable: Symfony es infinitamente escalable si se disponen de los recursos necesarios. Yahoo lo utiliza para programar aplicaciones con 20 millones de usuarios y 12 idiomas, además posee un centenar de complementos desarrollados por la comunidad.
- Probado y testeado: Symfony ha sido probado con éxito durante varios años en aplicaciones muy diferentes. Desde sitios web con millones de usuarios hasta otros miles de sitios pequeños y medianos, obteniendo en cada uno de ellos excelentes resultados.
- Soporte: Symfony sigue una política de tipo LTS[10]. Las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de los errores conocidos.
- Licencia: Symfony utiliza una licencia MIT, con la que puedes hacer aplicaciones web comerciales, gratuitas y/o de software libre.
- Compromiso: la empresa que ha creado Symfony (*SensioLabs*) no genera ingresos del marco de trabajo, sino de las aplicaciones que hacen con él.
- Código: desde su primera versión Symfony ha sido creado exclusivamente para PHP 5, desechando la versión PHP 4 (que ha sido declarada obsoleta recientemente).
- Seguro: se puede controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS[11] y CSRF[12].

[9] Un helper es una función, definida por Symfony, que puede tener parámetros y devolver código HTML. La mayoría de las veces, los helpers te ahorran tiempo, ellos empaquetan código en snippets (porciones de código) utilizados con frecuencia en las plantillas. El helper`include_stylesheets ()` genera una etiqueta `<link>` para las hojas de estilo.

[10] Brinda soporte durante más tiempo del habitual, 3 años para la versión de escritorio y 5 para la de servidores.

[11] Tipo de inseguridad informática o agujero de seguridad basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.

- Documentado: Se trata del marco de trabajo PHP mejor documentado: tutoriales de hasta 250 páginas y libros gratuitos de casi 500 páginas. Además, los libros están completamente traducidos al español.
- Calidad: su código fuente incluye más de 8.000 pruebas unitarias y funcionales.
- Internacionalización: se pueden crear aplicaciones en varios idiomas. La internacionalización está integrada en el marco de trabajo, funciona bien, es muy completa y está probada en aplicaciones reales.

1.4.2.2 JQuery

JQuery es una biblioteca o marco de trabajo de Java Script, creada inicialmente por John Resig, y que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM[13], manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

JQuery es libre y de código abierto, y al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en Java Script que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

1.4.2.3 Doctrine como mapeo objeto-relacional (ORM[14])

Librería para PHP que permite trabajar con un esquema de base de datos como si fuera un conjunto de objetos, y no de tablas. Doctrine está inspirado en *Hibernate*, que es uno de los ORM más populares y grandes que existe, capaz de brindar una capa de abstracción de la base de datos muy completa. Su característica más importante es que nos posibilita escribir consultas de base de dato en un lenguaje propio llamado *Doctrine Query Lenguaje* (DQL), que por demás resulta de fácil aprendizaje para los programadores.

[12] Fragmento de datos malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

[13] Interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

[14] Es el encargado de realizar la conversión de los objetos en registro y para recuperar los datos realiza el proceso inverso simulando que la base de datos es orientada a objeto.

1.4.3 Gestor de base de datos

Los sistemas de gestión de bases de datos (SGBD) surgieron por la necesidad que existía de mejorar los sistemas tradicionales de procesamiento de archivos. Estos sistemas convencionales traían consigo una serie de problemas como son la redundancia e inconsistencia de los datos, la dificultad para acceder a estos y su aislamiento, la existencia de múltiples usuarios y problemas de seguridad e integridad. Por lo que 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, Oracle y Sybase ASE. (10)

1.4.3.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, es una derivación libre del proyecto Postgres, que incluye características de la orientación a objetos como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional; pero a pesar de esto PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Una de sus principales características es que soporta distintos tipos de datos. Además de los del tipo base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits[15] y permite la creación de tipos propios. (11)

1.4.4 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, del inglés Integrated Development Environment) es un paquete de software diseñado para la creación y ejecución de un programa. Son varias las compañías de software importantes, las que cuentan con estos entornos de desarrollo, los cuales combinan las funciones de editor, compilador, enlazador y visor de *applets*. Un entorno de desarrollo integrado ofrece funcionalidades para facilitar tanto como sea posible la creación de un programa informático.

Por lo anterior expuesto se puede concluir que un entorno de desarrollo integrado, es un conjunto de herramientas que permiten realizar la programación de un software. Está compuesto por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, pudiendo dedicarse solamente a un lenguaje de programación o a varios, tales como, Eclipse o NetBeans que pueden usar plugins para soportar diversos lenguajes como son C / C ++, Ada, Perl, Python, Ruby y PHP.

[15] Acrónimo de *Binarydigit* (dígito binario). Un bit es un dígito del sistema de numeración binario.

1.4.4.1 NetBeans IDE

Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre distintas plataformas, haciendo uso de la tecnología Java. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco, sus funcionalidades son ampliables mediante la instalación de paquetes. Además, soporta PHP 5.3 y Symfony. (12)

1.4.5 Herramienta CASE

Las herramientas CASE (del inglés Computer Aided Software Engineering), son un conjunto de programas que ayudan a aumentar la productividad en el desarrollo de *software*, reduciendo costo 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, dígase la realización del diseño del sistema y la implementación automática de partes de este, cálculo de costes, compilación automática, documentación y detección de errores.

1.4.5.1 Visual Paradigm 6.4

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar todos los tipos de diagramas de clases y generar códigos desde estos, también nos permite generar la documentación. Con VisualParading 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 VisualParading se puede generar la base de datos ya que realiza la transformación de diagramas de Entidad Relación en tablas de la base de datos, también permite la realización de la ingeniería inversa de esta ya que puede llevar Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación. Esta herramienta CASE permite organizar los diagramas automáticamente reorganizando las figuras y conectores. Soporta el mapeo de objeto relacional (ORM) y está ampliamente integrada con Netbeans. (13)

1.5 Conclusiones parciales

Con el estudio realizado en este primer capítulo se logró identificar las principales características de los sistemas estudiados. Se realizó un análisis de la metodologías, herramientas y lenguajes definidos por el proyecto para el desarrollo de la aplicación, seleccionando como metodología de desarrollo RUP, para realizar el modelado la herramienta CASE Visual Paradigm y como marco de trabajo el *framework* Symfony. Finalmente, se arribó a la conclusión de que utilizando correctamente dichas herramientas y cumpliendo estrictamente todas las normas y procedimientos de la implementación, se obtendrá como resultado el desarrollo de los módulos de Autenticación, Solicitudes y Pagos del SIB.

Capitulo 2: Propuesta de solución

2.1 Introducción

En este capítulo se realizará la propuesta de solución donde se identificarán los casos de uso del sistema, los requerimientos funcionales y no funcionales, los cuales determinarán el funcionamiento del sistema a construir, y sucesivamente avanzar hacia el modelado del mismo, de modo que se satisfagan las necesidades de los usuarios finales del sistema. Se especifican además los patrones de diseño utilizados así como la descripción del modelo de datos del sistema en cuestión.

2.2 Proceso a informatizar

Se desea informatizar los procesos de gestión de solicitudes de servicios y registro de pagos de los clientes venezolanos en el SIB, además de brindarle al mismo un sistema de autenticación para una mayor seguridad en la realización de estos procesos. En este sentido, se gestionarán las solicitudes de los servicios contratados por los clientes donde el mismo podrá crearlas, modificarlas o eliminarlas. Además, se le brindara al usuario la información requerida sobre las mismas, así como la cantidad de solitudes que tiene realizada el cliente y los servicios contratados en cada solicitud realizada.

En cuanto al registro de pagos, los clientes tendrán la posibilidad de registrar el pago de las solicitudes realizadas. Ya que el cliente deposita una cantidad de dinero en el banco de donde se descuenta a medida que valla registrando las solicitudes que desea pagar.

Igualmente, los clientes tendrán la posibilidad de autenticarse en el sistema mediante la utilización de su usuario registrado con anterioridad y la contraseña correspondiente al mismo, brindando así una mayor seguridad en las operaciones.

2.3 Modelo de dominio

La investigación se basará en un modelo de dominio, que permite mostrar al usuario los principales conceptos que se operan en el dominio de la aplicación en desarrollo. De esta forma los usuarios utilizarán un vocabulario común, para lograr entender el contenido en que se enmarca la aplicación. Este modelo contribuye a describir las clases más importantes dentro del contexto de la aplicación. A continuación se identifican los conceptos más significativos que se utilizarán en el modelo de dominio:

Cliente, es el usuario que ya está autenticado, que puede gestionar una solicitud ya sea crearla, modificarla o eliminarla. Operador de oficina es el usuario que registra los pagos ya sean ordinarios o extraordinario.

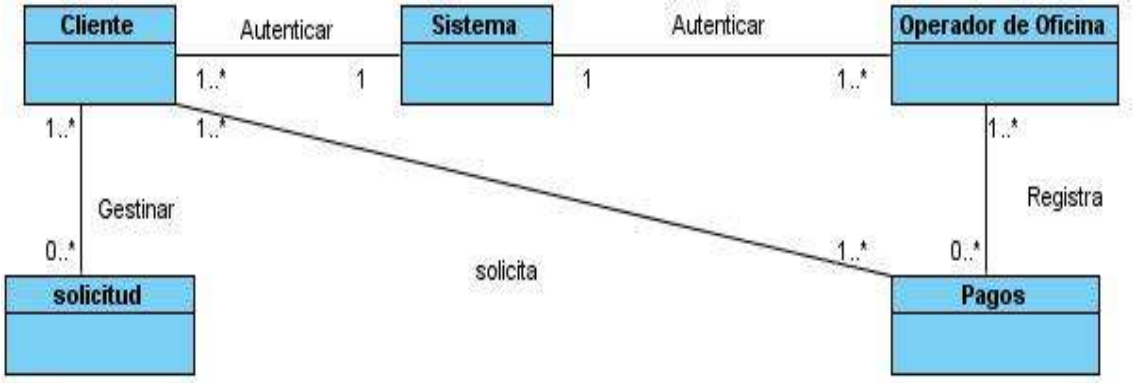


Figura 5 : Modelo de dominio (elaboración propia).

2.4 Propuesta de solución

Para lograr darle solución a la problemática planteada, se propone realizar la implementación de una aplicación web que gestione la recaudación de pagos de SAIME, compuesta por tres módulos que se encarguen de la autenticación, las solicitudes y los pagos que realizan los clientes. Para el desarrollo de estos tres módulos se utilizará el marco de trabajo Symfony con Doctrine como ORM y lenguaje de programación PHP, permitiendo la elaboración de un aplicación web que brinde la posibilidad de realizar las actividades que se encuentran incluidas en estos módulos.

Módulo Autenticación: este módulo se encargará de permitir la seguridad de los clientes en el SIB, el cual está basado en la categoría de identificación número uno: algo que el usuario sabe o conoce y el resto de las personas no, que no es más que autenticación basado en algo conocido (contraseñas). Esta información se almacenará en la base de datos y cuenta con un sistema de seguridad basado en usuarios, los cuales tendrán roles definidos y estos a su vez tendrán permisos sobre acciones específicas. La aplicación exigirá un nivel alto en cuanto a la complejidad de las contraseñas de los usuarios. Las mismas deben ser mayores de 7 caracteres y deben contener letras, números y caracteres especiales. El sistema validará automáticamente la fortaleza de la contraseña.

Módulo Solicitudes: este módulo se encargará de gestionar las solicitudes realizadas por el cliente, ya sea crear una nueva solicitud, modificar una ya creada o eliminar la solicitud deseada; brindándole a su vez el

estado de dichas solicitudes así como la cantidad de solicitudes activas que posee. En cada una de estas se mostrará al cliente, la cantidad de servicios contratados que contiene dicha solicitud.

Módulo Pagos: este módulo se encargará de permitir que los clientes realicen el registro de los pagos para cada solicitud de servicio realizada, mostrándole una interfaz para que introduzca todos los datos requerido, para que posteriormente seleccione de una lista de solicitudes realizadas por el, las que desee registrar como pagada, después de haber ingresado en el banco una cierta cantidad de dinero como respaldo para la compra de cada servicio deseado.

2.5 Especificación de los requisitos del sistema

La definición de los requisitos del sistema es una tarea sumamente importante y que tiene una repercusión directa en 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 denotara dolencias funcionales y de usabilidad y, lo que suele ser peor, repercutirá negativamente en el coste económico del sistema.

2.5.1 Requisitos funcionales

Dentro de los requisitos del sistema se encuentran los requisitos funcionales, a través de los cuales se puede especificar de forma más detallada de las responsabilidades del sistema. Los requisitos funcionales permiten determinar, de manera clara y concisa, lo que debe hacer el sistema y los mismos se mantienen invariables sin importar con que propiedades o cualidades se relacionen. (14)

Requisitos funcionales definidos para darle solución a la problemática planteada

RF-1. Crear solicitud de pago de servicios.

- RF-1.1. Mostrar los proveedores de servicio.
- RF-1.2. Ordenar los proveedores de servicios por orden alfabético ascendentemente.
- RF-1.3. Seleccionar un proveedor de servicio.
  - RF - 1.3.1. Mostrar los servicios que brinda el proveedor seleccionado por orden alfabético ascendentemente.
- RF-1.4. Comprar uno o varios servicios.
  - RF - 1.4.1. Comprar varias unidades de un servicio dado.
  - RF - 1.4.2. Crear la nueva solicitud con los servicios seleccionados.

RF - 1.4.3. Calcular el monto de la solicitud de pago sumando el monto total de cada unidad de servicio comprado.

RF - 1.4.4. Mostrar el monto total de la solicitud creada.

RF-1.5. Guardar la solicitud de pago con las unidades de servicios compradas y el monto total calculado.

RF-2. Modificar solicitudes activas.

RF-2.1. Modificar servicios solicitados en solicitud activa.

RF-2.1.1. Seleccionar la solicitud que se desea modificar.

RF-2.1.2. Mostrar los servicios que tienen el mismo costo que el servicio seleccionado y que no está en la solicitud actual.

RF-2.1.3. Seleccionar un nuevo servicio para intercambiar con el de dentro de la solicitud.

RF-2.1.4. Verificar que el servicio dentro de la solicitud no está pagado todavía.

RF-2.1.4.1. Cancelar la opción de intercambio de servicios en este caso.

RF-2.1.4.2. Notificar al usuario del resultado de la acción de verificación.

RF-2.1.5. Intercambiar servicios, el nuevo seleccionado con el ya existente dentro de la solicitud.

RF-2.2. Guardar los cambios efectuados a la solicitud.

RF-2.3. Eliminar servicios dentro de una solicitud.

RF-2.3.1. Listar las solicitudes activas.

RF-2.3.2. Seleccionar solicitud.

RF-2.3.3. Listar los servicios de la solicitud.

RF-2.3.4. Seleccionar varios servicios.

RF-2.3.5. Eliminar los servicios seleccionados.

RF-2.3.6. Validar que a la solicitud le quede al menos un servicio.

RF-2.3.7. Re-calcular el monto total de la solicitud.

RF-3. Eliminar solicitudes de pago activas.

RF-3.1. Listar las solicitudes de pago de servicios existentes en el estado que se encuentran.

RF-3.2. Ordenar alfabéticamente las solicitudes de pago de servicios de manera ascendente.

RF-3.3. Seleccionar una o varias solicitudes de pago.

RF-3.3.1. Validar que la solicitud no tenga algún servicio pagado.

RF-3.3.1.1. Cancelar eliminación en caso de que haya algún servicio pagado.

RF-3.3.1.2. Notificar al usuario del resultado de la validación.

RF-8. Registrar pago ordinario de solicitud activa.

RF-8.1. Recibir los datos de la ráfaga bancaria:

- BCO – Banco y Sucursal donde se realizó el depósito
- CSS – Código de Seguridad del SAIME
- NTR – Número de transacción del día
- FEC – Fecha de pago
- HRM – Hora militar y minutos del pago
- CID – Cédula del depositante
- CIB – Cédula del beneficiario
- MRT – Monto recaudado total

RF-8.2. Obtener el "CSS".

RF-8.2.1. Determinar los datos esenciales.

RF-8.2.2. Obtener la cadena de seguridad vigente de acuerdo a la FEC para la sucursal especificada en BCO.

RF-8.2.3. Mezclar las cadenas de los datos esenciales y la cadena de seguridad.

RF-8.2.4. Obtener función B64.

RF-8.2.5. Obtener función B32.

RF-8.3. Comparar el CSS de la ráfaga con el CSS' obtenido.

RF-8.3.1. Notificar al usuario si la ráfaga no fue validada correctamente.

RF-8.4. Buscar las solicitudes efectuadas según la cédula especificada en CIB.

RF-8.4.1. Mostrar las solicitudes encontradas.

RF-8.4.1.1. Seleccionar la solicitud activa especificada por el usuario.

RF-8.4.1.2. Especificar el monto.

RF-8.4.1.3. Comprobar el monto pagado.

RF-8.4.1.4. Pagar la solicitud.

RF-8.4.2. Crear nueva solicitud para el cliente con cédula descrita en CIB.

RF-8.4.2.1. Capturar datos CIB y MRT.

RF-8.4.2.2. Mostrar los servicios que brinda el proveedor por orden alfabético ascendentemente.

RF-8.4.2.3. Seleccionar uno o varios servicios.

RF-8.4.2.4. Seleccionar varias unidades de un servicio dado.

R-8.4.2.5. Crear la nueva solicitud con los servicios seleccionados.

RF-8.4.2.6. Calcular el monto de la solicitud de pago sumando el monto total de cada unidad de servicio comprado.

RF-8.4.2.7. Chequear que el monto total calculado es inferior al MRT capturado.

RF-8.4.2.8. Guardar la solicitud de pago con las unidades de servicios seleccionadas y el monto total calculado.

RF-8.5. Guardar metadatos del usuario proveedor de servicios que registra la acción.

RF-8.6. Crear automáticamente pago extraordinario si MRT es menor que monto total de la solicitud seleccionada.

RF-8.6.1. Generar PIdeS asociada a las solicitudes detenidas.

- Número de la planilla
- Fecha de emisión
- Monto total a cancelar
- Monto total en letras
- CI/RIF/Pasaporte del cliente
- Nombre y apellidos
- Identificador de la solicitud detenida
- Cadena de seguridad de la planilla (CSP)

RF-8.6.2. Generar CSP.

RF-8.6.2.1. Determinar los datos esenciales.

RF-8.6.2.2. Obtener la cadena de seguridad vigente para la oficina que emite la PIdeS.

RF-8.6.2.3. Mezclar las cadenas de los datos esenciales y la cadena de seguridad.

RF-8.6.2.4. Obtener función B32.

RF-8.6.2.5. Obtener función B8.

RF-8.6.3. Imprimir PIdeS.

RF-8.6.4. Guardar metadatos del usuario proveedor de servicios que registra la acción.

RF-8.7. Crear automáticamente pago excedente si MRT es mayor que monto total de la solicitud seleccionada.

RF-8.7.1. Registrar pago excedente a la solicitud seleccionada.

RF-8.7.2. Notificar al cliente por SMS y correo electrónico de que tiene pago excedente.

RF-10. Registrar pago extraordinario a solicitud detenida.

RF-10.1. Recibir los datos de la ráfaga bancaria:

- BCO – Banco y Sucursal donde se realizó el depósito
- CSS – Código de Seguridad del SAIME
- NTR – Numero de transacción del día
- FEC – Fecha de pago
- HRM – Hora militar y minutos del pago
- CID – Cédula del depositante
- CIB – Cédula del beneficiario
- MRT – Monto recaudado total
- CSP – Código de seguridad de la PIdeS

RF-10.2. Obtener el “CSS”.

RF-10.2.1. Determinar los datos esenciales.

RF-10.2.2. Obtener la cadena de seguridad vigente de acuerdo a la FEC para la sucursal especificada en BCO.

RF-10.2.3. Mezclar las cadenas de los datos esenciales y la cadena de seguridad.

RF-10.2.4. Obtener función B64.

RF-10.2.5. Obtener función B32.

RF-10.3. Comparar el CSS de la ráfaga con el CSS' obtenido.

RF-10.3.1. Notificar al usuario si la ráfaga no fue validada correctamente.

RF-10.4. Buscar las solicitudes efectuadas según la cédula especificada en CIB.

RF-10.4.1. Mostrar las solicitudes encontradas.

RF-10.4.1.1. Seleccionar la solicitud activa detenida cuyo monto se corresponda con el especificado en MRT.

RF-10.4.1.2. Especificar el monto.

RF-10.4.1.3. Comprobar el monto pagado.

RF-10.4.1.4. Pagar la solicitud.

RF-10.4.1.5. Guardar metadatos del usuario proveedor de servicios que registra la acción.

RF-12. Crear pago excedente.

RF-12.1. Mostrar las solicitudes cuyo pago es mayor al necesario para cubrir el monto total.

RF-12.2. Seleccionar las solicitudes a las que se les desea crear pago excedente.

RF-12.3. Generar pago excedente asociado a la solicitud.

2.5.2 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (14)

Requisitos de hardware

Para la explotación del sistema se requiere una computadora que haga función de un servidor y una computadora que haga función de cliente, las cuales deben cumplir con las características siguientes:

Tabla 2: Requisitos de hardware (elaboración propia).

Cantidad de PC	1 PC Servidor	1 PC Cliente
Microprocesador	Pentium IV (mínimo)	Pentium IV (mínimo)
Memoria RAM	0.99 GB (mínimo)	0.99 GB (mínimo)
Capacidad disco duro	80 GB (mínimo)	80 GB (mínimo)
Sistema operativo	El sistema propuesto es multiplataforma por lo que se puede tener cualquier sistema operativo	El sistema propuesto es multiplataforma por lo que se puede tener cualquier sistema operativo

Requisitos de apariencia o interfaz externa

- La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización.
- El diseño 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.
- Para tareas similares, el entorno se mantendrá estable utilizando nomenclaturas, elementos estructurales y tipografías iguales.
- 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.
- Se tendrán en cuenta todas las acciones que tienen impacto dentro del sistema para manejar las informaciones y confirmaciones que debe dar el mismo al usuario.
- La ayuda siempre estará en una zona accesible y visible en todo momento sin que interrumpa la tarea que se lleva a cabo.

Restricciones en el diseño y la implementación

- Las herramientas de desarrollo de la aplicación serán, Netbeans 6.9, Visual Paradigm y PgAdmin III.
- El lenguaje de programación será PHP 5.3.
- El marco de trabajo será el *framework* Symfony.
- Como mapeo de objeto relacional Doctrine.
- Se utilizará la Programación Orientada a Objetos.

Requisitos de usabilidad

- Debe existir un mapa de navegación consistente que garantice la orientación del usuario.

- La ayuda estará organizada de forma que el usuario entienda donde buscar lo que necesita y que las explicaciones sean lo más concisas y explícitas posibles.
- 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.
- El sistema deberá visualizarse correctamente en los principales navegadores que existen en la actualidad.

Requisitos de seguridad

- Se realizarán las validaciones que garantizan el sentido de los datos con respecto al negocio. Se debe discriminar la entrada de caracteres especiales principalmente: comillas simples (‘ ‘), comillas dobles (“ “), paréntesis angulares (<>), asterisco (*) y cualquier otro que no tenga sentido para la lógica de la aplicación.
- Se validará la entrada de palabras reservadas SQL para mitigar algún tipo de inyección SQL.
- Disponibilidad: la información generada en el sistema por cada usuario, estará centralizada y en servidores seguros desde donde el sistema podrá generar diferentes reportes en dependencia de los usuarios del sistema y sus respectivos roles o niveles de acceso.
- Integridad: garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario para evitar entradas inadecuadas.

Rendimiento

- El sistema operará con grandes volúmenes de información, por tanto, se hacen necesarios tiempos de respuestas cortos, al igual que la velocidad de procesamiento de la información.

Funcionalidad

- Mínima cantidad de páginas para ejecutar todas las funciones posibles, es decir, agrupar funciones afines en las mismas páginas.

2.6 Actores del sistema

Los actores de un sistema son agentes externos que interactúan con el sistema, éste puede ser una persona, una organización, otro sistema o empresa. Los actores realizan una labor frente al sistema

intercambian información con éste. En la Tabla 3 se muestran los actores del sistema y sus respectivas descripciones.

Tabla 3: Descripción de los Actores del sistema (elaboración propia).

Actor	Descripción
Cliente	Cualquier usuario que desee registrarse en el sistema, y se encargue de modificar sus preferencias y recuperar su contraseña.
Operador de oficina	Persona encargada de realizar el registro de los pagos ya sea ordinario o extraordinario además de crear el pago excedente.

2.7 Modelo de 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. Estos representan las acciones que un usuario (actor) realiza sobre el sistema (caso de uso) y modelan el sistema desde el punto de vista del usuario. Son muy fáciles de interpretar por lo que son muy útiles en la comunicación con los clientes. El diagrama de casos de uso del sistema que se desea implementar se muestra en la Figura 6.

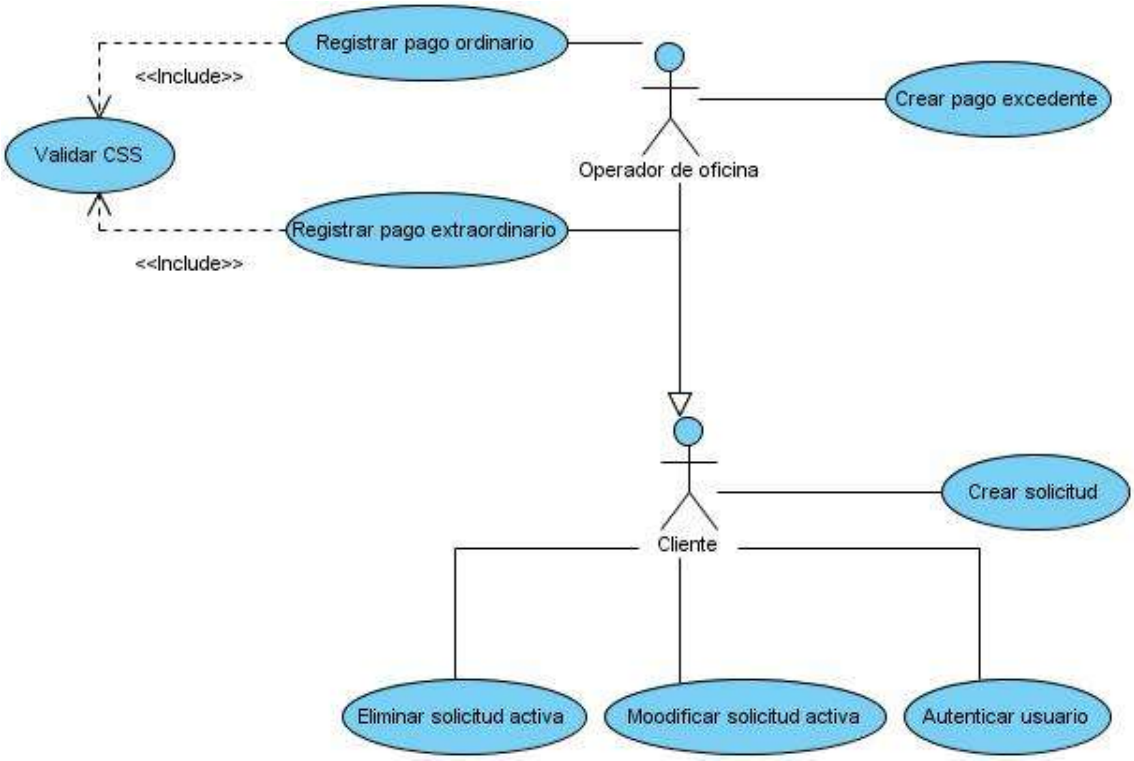


Figura 6: Diagrama de casos de uso (elaboración propia).

2.7.1 Especificación de los casos de uso del sistema

Para entender la funcionalidad asociada a los casos de uso, no es suficiente con la representación gráfica del diagrama de casos de uso. La descripción de los casos de usos describe paso a paso, qué es lo que el sistema debe hacer, separando las responsabilidades del sistema y la de los actores.

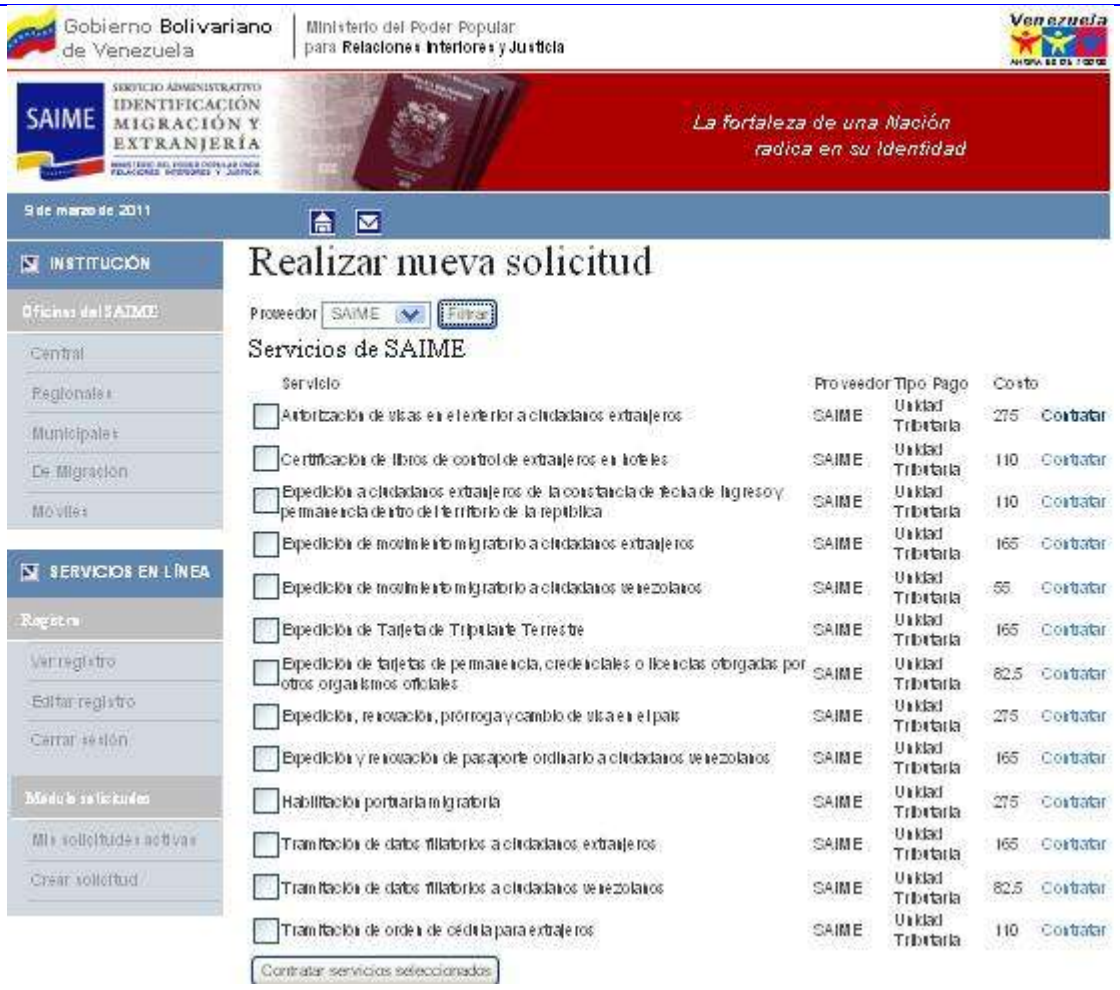
A continuación se presenta la descripción del caso de uso **Crear Solicitud** del módulo Solicitudes del SIB. Las restantes descripciones de casos de uso se pueden consultar en el anexo 1.

Tabla 4: Descripción del caso de uso "Crear solicitud" (elaboración propia).

Caso de Uso:	**Crear Solicitud**
Actores:	Solicitante
Resumen:	El caso de uso inicia cuando el Solicitante ejecuta la opción **Crear Solicitud** donde compra los servicios del proveedor seleccionado, finalizando así el caso de uso.

Precondiciones:	El Solicitante debe estar autenticado en el sistema.
Referencias	**RF1**
Prioridad	**Critico**

Flujo Normal de Eventos	
"Crear Solicitud"	
Acción del Actor	**Respuesta del Sistema**
1- El Solicitante selecciona la opción **Crear Solicitud**.	1.1- El sistema muestra una interfaz con los proveedores de servicio ordenados por orden alfabético ascendentemente.
2- El Solicitante selecciona un proveedor de servicio.	2.1- El sistema muestra los servicios que brinda el proveedor seleccionado, por orden alfabético ascendentemente, el campo unidades y el botón: • **Contratar servicios seleccionados**
3- El Solicitante selecciona el o los servicios que desea comprar y las unidades del mismo.	3.1- Lista las unidades de servicios seleccionados y calcula el monto total de la solicitud. Muestra los botones: • **Guardar solicitud** • **Cancelar** • **Mis solicitudes activas**
4- El solicitante selecciona: 4a) **Guardar solicitud,** continúa con el flujo normal de eventos en la parte 4.1. 4b) **Cancelar,** ver **Flujo Alterno 1 "Cancelar Crear Solicitud"** 4c) **Mis solicitudes activas,** ver **Flujo Alterno 2 "Mis solicitudes activas"**	4.1- El sistema guarda la solicitud de pago.
Prototipo de Interfaz	



Flujos Alternos	
Flujo Alterno 1 "Cancelar Crear Solicitud"	
Acción del Actor	**Respuesta del Sistema**
	• El sistema muestra un mensaje indicando que será cancelada la Solicitud.
• El Solicitante selecciona: 2a) Cancelar la solicitud, se reinician las variables de la creación de la solicitud y se muestra la interfaz principal.	
Flujo Alterno 2 "Mis solicitudes activas"	
Acción del Actor	**Respuesta del Sistema**

	1- Se muestra la interfaz principal

2.8 Diseño

La realización del diseño de un sistema se hace con el objetivo de refinar el análisis y teniendo 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.

2.8.1 Descripción de la arquitectura

La arquitectura es el esqueleto o base de una aplicación y representa la organización fundamental de un sistema. Desde los pequeños programas hasta los sistemas más grandes, poseen una estructura y un comportamiento que los hace clasificables según su “arquitectura”. En la web es muy común la utilización de la arquitectura “3-capas”, “n-capas”, “MVC”, entre otras.

El *framework Symfony*, seleccionado para desarrollar la aplicación, implementa el patrón arquitectónico Modelo – Vista – Controlador (MVC), y es por ello que se adopta esta arquitectura para el desarrollo de la propuesta de solución.

Modelo Vista Controlador (MVC)

Este patrón de arquitectura de *software* separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos, quedando como sigue:

- Modelo: representa la información con la que trabaja la aplicación, es decir, su lógica del negocio.
- Vista: presenta el modelo en un formato adecuado, como en una página web que le permite al usuario interactuar con ella, usualmente un elemento de interfaz de usuario.
- Controlador: responde a eventos, usualmente acciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

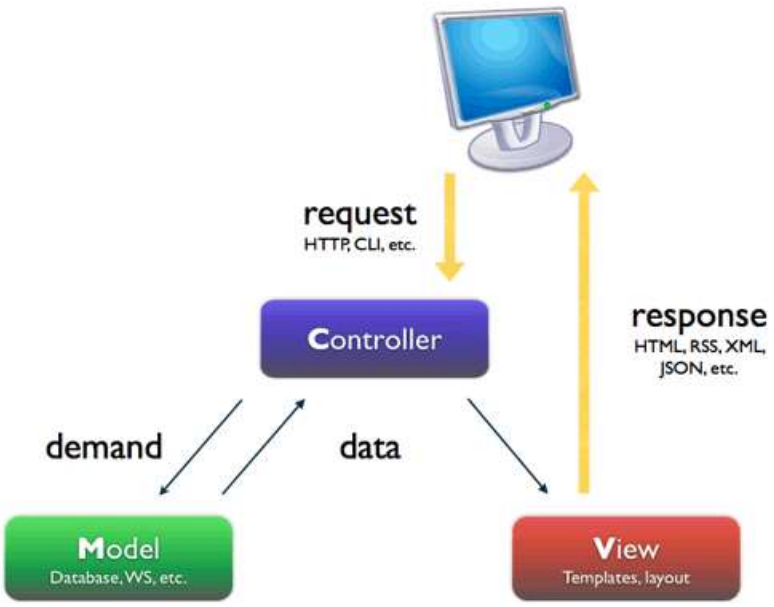


Figura 7: Patrón arquitectónico Modelo - Vista - Controlador (fuente: Jobeet symfony 1.4).

2.8.1.1 Patrones de diseño

La utilización del framework *Symfony* para el desarrollo del sistema proporciona ventajas significativas para los desarrolladores del software. Este framework mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones arquitectónicos y de diseño más utilizados en la actualidad, ya que Symfony los implementa por defecto.

2.8.1.1.1 Patrones GRASP

- Creador: en la clase Acción de cada uno de los módulos se encuentran definidas las acciones del sistema y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Acción es "creador" de dichas entidades.
- Experto: este es uno de los más utilizados, puesto que doctrine es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo. *Symfony* divide el modelo en una capa de abstracción a datos y otra de acceso a datos en las cuales se encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

- Alta Cohesión: *Symfony* permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Acción tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.
- Controlador: todas las peticiones son manejadas por un solo controlador frontal (sfAcción), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal 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: este patrón se manifiesta en cada uno de los módulos del sistema. La clase Acción hereda solamente de la clase sfAcción para lograr un bajo acoplamiento de clases.

2.8.1.1.2 Patrones GOF

En la categoría 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): permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo 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.

En la categoría estructurales:

- Decorator (Envoltorio): añade funcionalidad a una clase dinámicamente. El archivo **layout.php**, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. La Figura 8 ilustra lo anteriormente descrito.

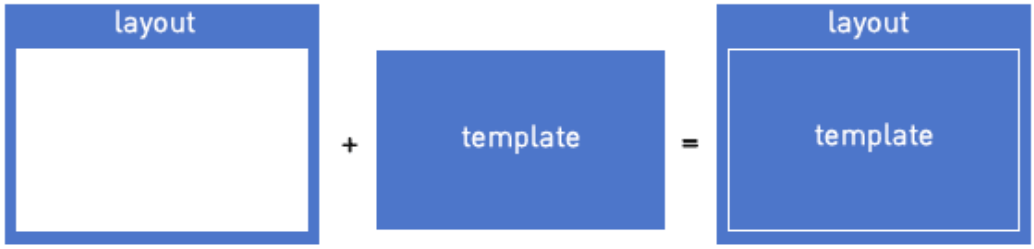


Figura 8: Funcionamiento del patrón Envoltorio (fuente: Jobeet symfony 1.4).

- Composite (Objeto compuesto): permite tratar objetos compuestos como si de un objeto simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

En la categoría comportamiento:

- Command (Acción): permite que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación.

2.8.2 Diagrama de clases del diseño

Los diagramas clases del diseño muestran una visión que describe las especificaciones de las clases de software y de las interfaces en una aplicación. Se utilizan principalmente para modelar la vista de 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 "**Crear solicitud**" se muestra en la Figura 9. Los restantes diagramas referentes al resto de los casos de uso del sistema se pueden ver en el Anexo 4.

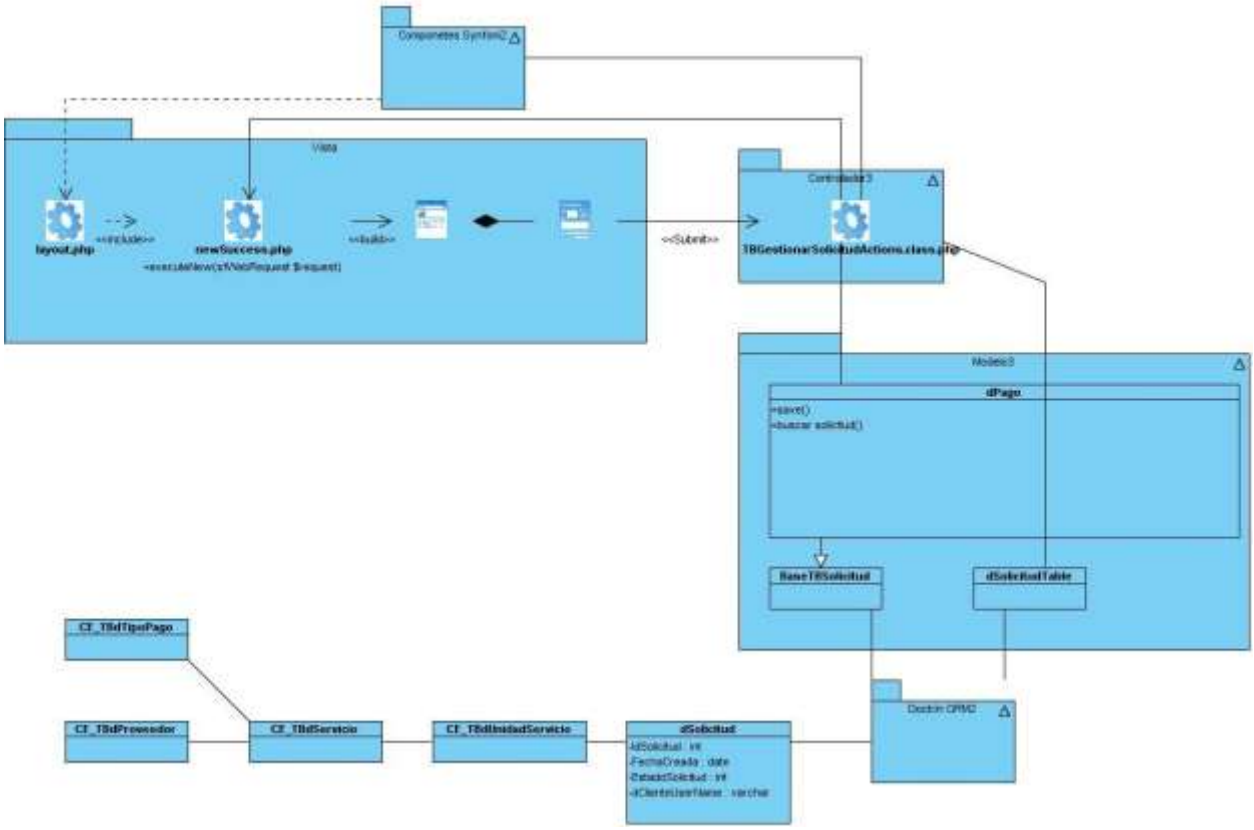


Figura 9: Diagrama de clases del diseño del caso de uso "Crear solicitud" (Elaboración propia).

2.8.3 Diagrama de interacción (Secuencia)

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 de la aplicación se representará mediante un grupo de diagramas de secuencia, cada uno de ellos vinculados generalmente a una sub función del sistema.

Los diagramas de secuencia del sistema en cuestión, describen las interacciones de un grupo de objetos mostrando de forma secuencial los envíos de mensajes entre los mismos. Igualmente se muestran los flujos de datos intercambiados durante el envío de mensajes.

A continuación se presenta el diagrama de secuencia “Crear solicitud”, donde el cliente interactúa con la aplicación web seleccionando la solicitud y el proveedor de la misma, seleccionando contratar, para así dejar por realizada la solicitud mostrándosele el precio de la misma como se muestra en la Figura 10.

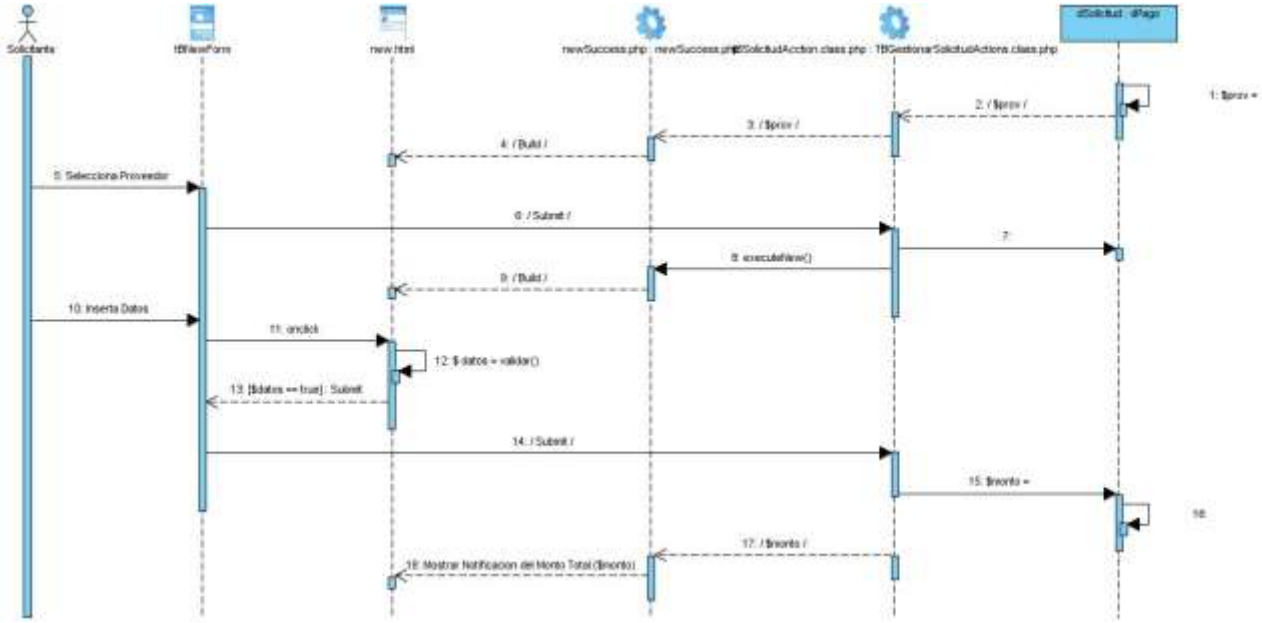


Figura 10: Diagrama de secuencia "Crear solicitud" (Elaboración propia).

2.8.4 Diseño de la base de datos

La persistencia es la capacidad de un objeto de mantener su valor en el tiempo y el espacio. En el proceso de abstracción que conduce a la creación de una base de datos, desempeña una función prioritaria el modelo de datos.

El modelo de datos, es el enfoque utilizado para la representación de las entidades y sus características dentro de la base de datos. Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones. La propuesta de base de datos que se expone a continuación (Figura 11), satisface las necesidades de persistencia de los datos que el sistema requiere, en cumplimiento de sus requerimientos funcionales.

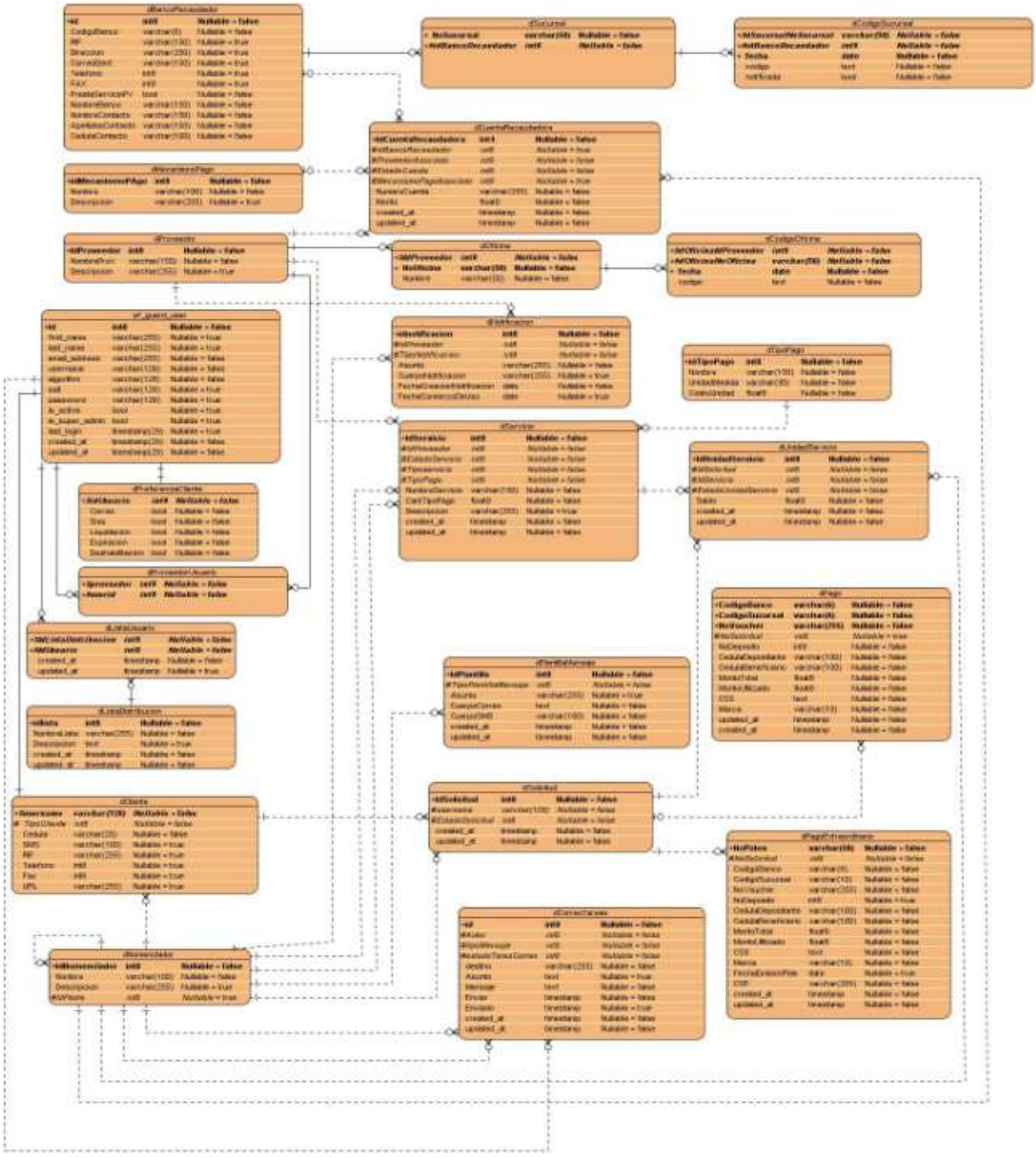


Figura 11: Modelo de datos (Elaboración propia).

2.8.4.1 Descripción del modelo de datos

Tabla 5: Descripción del Modelo de datos (Elaboración propia).

Tablas	Descripción
sf_guard_user	Guarda todos los usuarios del sistema.
sf_guard_permission	Guarda los permisos de que tendrán los usuarios.
Sf_guard_user_permission	Guarda los datos de la relación de las tablas ***Sf_guard_user*** y ***sf_guard_permission*** producto de la relación de muchos a muchos que existe entre ellas.
dcliente	Guarda todos los datos de los clientes del sistema.
dNomenclador	Guarda todos los campos multievaluados del sistema.
dsolicitud	Guarda todas las solicitudes realizadas por un cliente.
dservicio	Guarda todos los servicios que brinda el SAIME.
dBancoRecaudador	Guarda los datos de los bancos recaudadores.
dPago	Guarda los datos de los pagos realizados por los clientes.
dSucursal	Guarda los datos de las sucursales de los bancos.
dCodigoSucursal	Guarda los códigos que se generan para cada sucursal.
dPagoExtraordinario	Guarda los datos de los pagos registrados de este tipo por los clientes.
dProveedor_dServidor	Guarda los datos que permiten acceder a un proveedor a un servidor.

dProveedor	Guarda los datos de los proveedores.

2.8.5 Definiciones de diseño a aplicar

Desarrollo de prototipos de interfaz de usuario

La interfaz ha de ser lo más uniforme posible utilizando una apariencia profesional. Se usará el estándar de los sistemas anteriormente elaborados, que pertenecen a esta organización, para garantizar una homogeneidad entre todas las aplicaciones. Se utilizará colores iguales o similares en todas las páginas, con textos concisos y claros sin mezclar muchos tipos de letra y tamaños en cada una. La navegación debe ser ágil, por lo que la presencia de imágenes innecesarias y todo lo que atente contra una navegación rápida y eficiente deben ser eliminadas.

Para obtener mejor entendimiento y desarrollo del sistema se comienza a definir y 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 este. En la Figura 12 se muestra el prototipo desarrollado para la interfaz principal.

Figura 12: Prototipo de interfaz principal (Elaboración propia).

Tratamiento de errores

Para que la aplicación funcione de manera óptima, es importante comunicarle al usuario cuando es incorrecta la operación que realiza. 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 y 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, como se muestra en la Figura 13.

Figura 13: Tratamiento de errores en la interfaz de usuario (Elaboración propia).

2.9 Conclusiones parciales

Con la correcta realización de los tres primeros flujos de la metodología utilizada se logra realizar una buena comprensión del negocio y la interpretación correcta de las necesidades del sistema. La descripción detallada de los procesos permitió obtener una mejor comprensión de los requisitos especificados por el cliente. Se pudieron precisar los casos de uso a automatizar, los trabajadores del sistema y una arquitectura robusta, sirviendo de base a los pasos posteriores de implementación y pruebas.

La realización del análisis y el diseño, mediante la confección de diferentes diagramas que describen como se realizarán los procesos, que 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 Autenticación, Solicitudes y Pagos, permitiendo estar preparado para realizar la implementación de éstos.

Capítulo 3: Implementación del sistema

3.1 Introducción

En el siguiente capítulo se abordará sobre la estructura de la aplicación y los componentes que son reutilizados, así como una representación física de como se implementó la solución a través de representaciones gráficas, enfocándose en el diagrama de componentes y diagrama de despliegue de la solución desarrollada.

3.2 Análisis de complementos reutilizados

Los complementos permiten agrupar todo el código diseminado por diferentes archivos y reutilizar este código en otros proyectos. Además, permiten encapsular clases, filtros, helpers, archivos de configuración, tareas, módulos, esquemas y extensiones para el modelo y archivos estáticos. Dentro de los complementos utilizados se cuenta con:

- SfDoctrineGuardPlugin: es un complemento de Symfony que proporciona características de autenticación y autorización por encima de la función de seguridad estándar de Symfony. Brinda el modelo (usuarios, grupos y objetos de permiso) y los módulos (sfGuardAuth, sfGuardGroup, sfGuardPermission y sfGuardUser) facilitando el control de la seguridad de la aplicación.
- SfJqueryReloadedPlugin: resulta de gran ayuda mucho en el manejo de la librería de JavaScript *JQuery* en la aplicación.

3.3 Diagrama de despliegue

El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto al *hardware* y el *software.* El diagrama de despliegue se utiliza en el diseño y la implementación, pudiéndose distinguir componentes y nodos así como las relaciones entre todos estos.

El sistema se encontrará desplegado en el centro de datos SAIME, donde se hospeda la aplicación web, que estará de cara a Internet. Existirá comunicación con los bancos a través de servidores FTP donde se colocarán y leerán los archivos XML. El sistema notificará de eventualidades a sus clientes y a los bancos recaudadores por medio de correo electrónico haciendo uso del servidor de correo. Se necesitarán dos (2) servidores para la base datos (uno principal y uno de respaldo), como se muestra en la Figura 14.

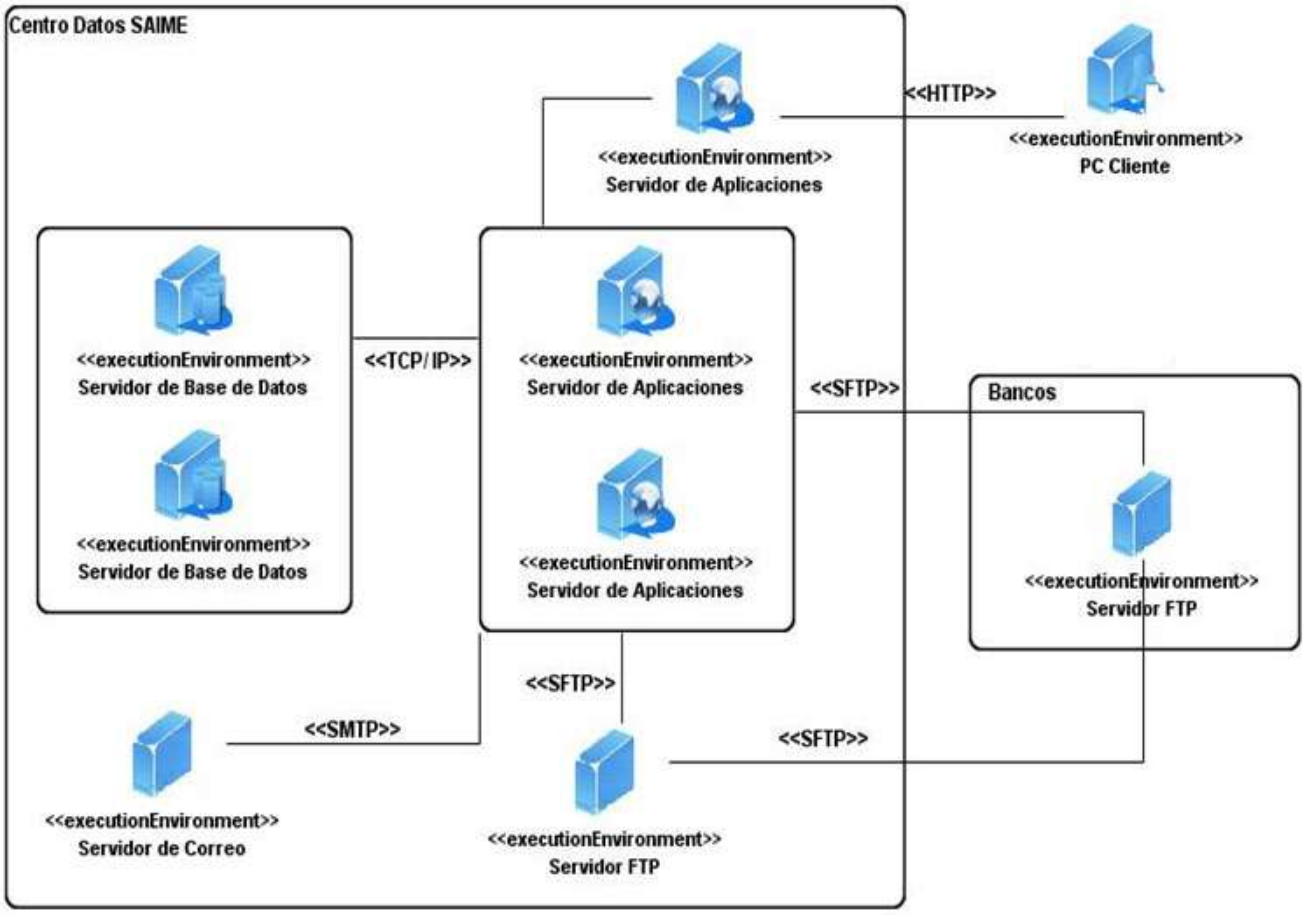


Figura 14: Diagrama de despliegue (Elaboración propia).

3.4 Diagrama de componentes

Es un diagrama tipo del Lenguaje Unificado de Modelado que representa como un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

El mismo muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema pues normalmente se realiza por partes. Cada diagrama describe un apartado del sistema en el cual se situarán las librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. En este caso se dividió el diagrama de

componentes del sistema en tres diagramas, uno por cada módulo implementado (autenticación, solicitudes y pagos).

En las Figuras 15, 16 y 17 se representan los diagramas de componentes para los módulos Autenticación, Solicitudes y Pagos respectivamente, donde 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* que son las clases donde se implementaron las funcionalidades de acceso a datos, utilizando el *framework* Doctrine como ORM para tener acceso a los datos.

El componente *Actions.class.php* que se encuentra en el **Controlador** ejecuta las plantillas que estarán en la **Vista**, la cual utilizará CSS que contendrá las hojas de estilo en cascada, así como el componente JQuery que permitirá 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.

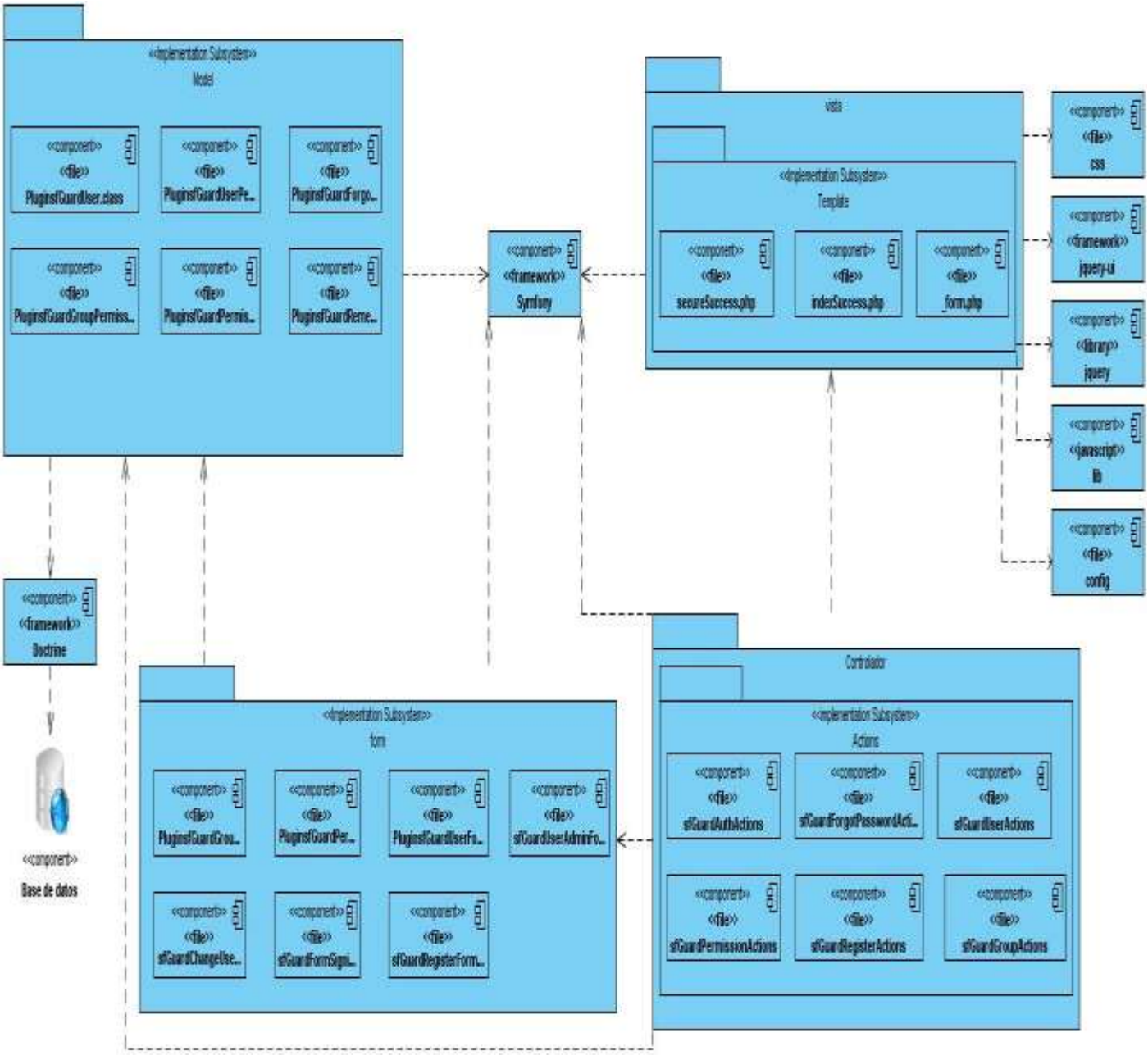


Figura 15: Diagrama de componentes del módulo "Autenticación" (Elaboración propia).

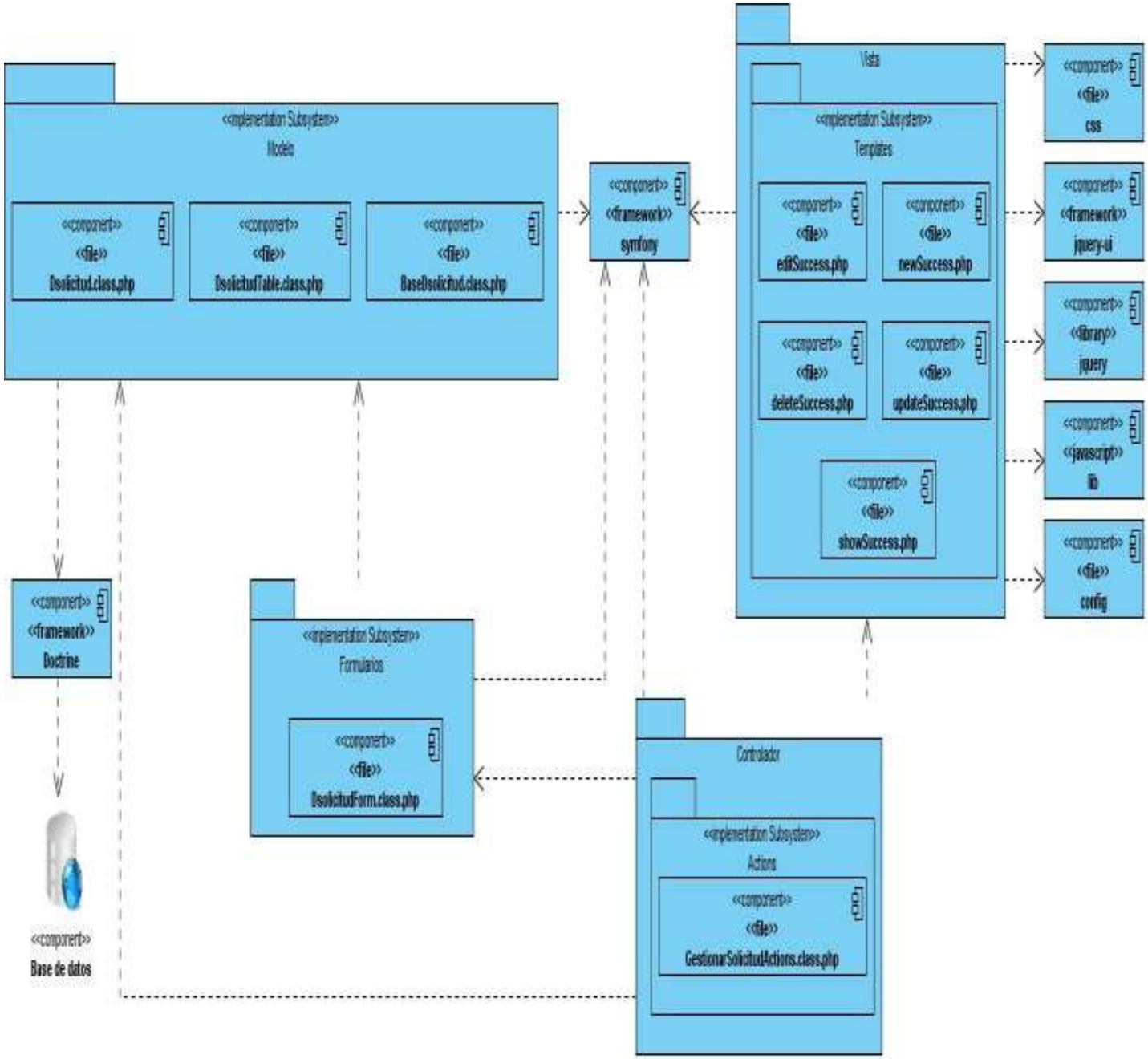


Figura 16: Diagrama de componentes del módulo "Solicitudes" (Elaboración propia).

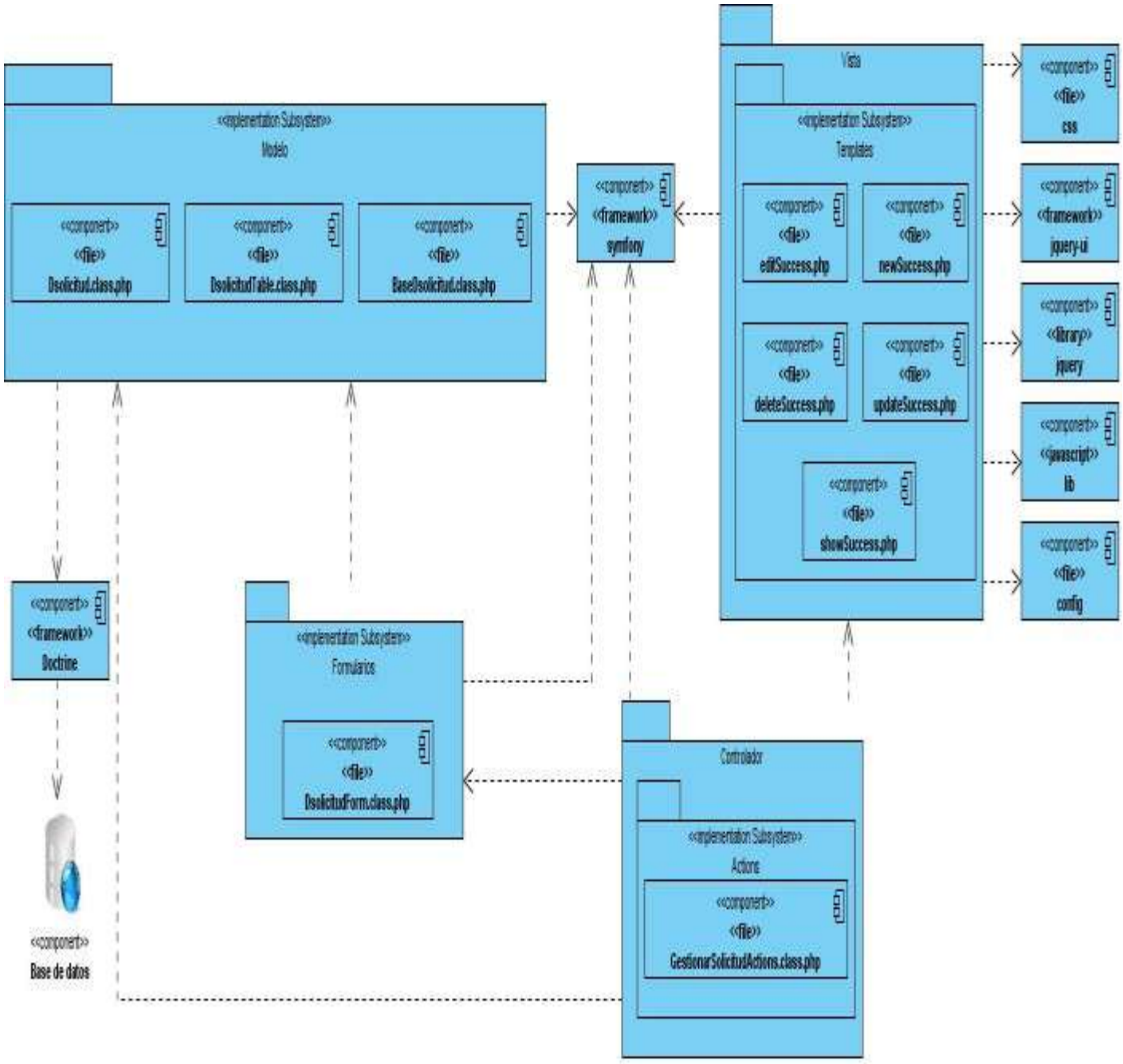


Figura 17: Diagrama de componentes del módulo "Pagos" (Elaboración propia).

3.5 Estándares de codificación

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 conlleven a lograr un código más legible, reutilizable y que guíen el desarrollo y la implementación desde el punto de vista arquitectónico estandarizando el código a implementar. Para la implementación del sistema propuesto se seguirá el siguiente estándar 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 la primera letra en mayúscula y las demás en minúscula y son un sustantivo singular. Se usaron palabras completas para evitar acrónimos y abreviaturas.

Nomenclatura de las funciones

- El nombre a emplear para las funciones se escribe con la primera palabra en minúscula 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 se escribe con la primera palabra en minúscula y se evitan los nombres de un solo carácter.

3.6 Conclusiones parciales

En este capítulo se mostró varias vistas para llevar a cabo el proceso de implementación del sistema, haciendo un análisis de los componentes utilizados. Se utilizó el diagrama de despliegue para identificar

las relaciones físicas de cada uno de los nodos del sistema además de las relaciones de los componentes sobre dichos nodos, así como la realización del diagrama de componentes para cada módulo implementado.

Capítulo 4: Pruebas al sistema

4.1 Introducción

En el proceso de desarrollo de software debido a la cantidad de actividades de producción que existe en el mismo, existe alta probabilidad de que se produzcan ciertos errores humanos. Estos fallos pueden presentarse debido a especificaciones erróneas e imperfectas de los requisitos, uso indebido de las estructuras de datos, errores al integrar módulos, entre otras causas. Debido la imposibilidad humana de ser perfectamente humanos y tener el derecho a equivocarnos el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

Las pruebas es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. En este capítulo se realiza la validación a la solución propuesta, con el objetivo de comprobar la eficiencia operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

4.2 Pruebas de software

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto. Se utilizan para identificar posibles fallos de implementación, calidad o usabilidad. Básicamente es una fase en el desarrollo de software consistente en probar aplicaciones construidas. Estas se integran dentro de las diferentes fases del ciclo de software dentro de la ingeniería de software, ejecutando un programa y mediante técnicas experimentales se trata de descubrir que errores tiene.

Para comprobar las funcionalidades se diseñan casos de prueba que definen cómo proceder. Estos casos de prueba incluyen los juegos de datos a utilizar: los válidos o esperados y los no válidos o no esperados por el programa. Además, se establecen los resultados a alcanzar en correspondencia con la lógica del programa y los datos ingresados. Describen las condiciones generales en las que se deben aplicar las pruebas para obtener los objetivos propuestos. El objetivo de los casos de prueba es forzar al máximo el sistema en los puntos críticos para encontrar fallos y detectar defectos.

Se deben escoger los tipos de prueba que se adapten mejor al sistema que se va a probar. Para esto se debe tener en cuenta el lenguaje de programación, el proceso de desarrollo, las características de los desarrolladores, 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 y si realiza conexiones a bases de datos entre otros.

4.3 Pruebas de Caja Negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Las pruebas de caja negra se centran principalmente en los requisitos funcionales del software.

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados con los métodos de caja blanca, por lo que se puede decir que estas no son excluyentes sino complementarias. Entre las principales características que presentan las pruebas de caja negra se pueden citar:

- Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- Es hecha sin el conocimiento interno del producto.
- No validan funciones ocultas por tanto los errores asociados a ellas no serán encontrados.

4.4 Diseño de Casos de Prueba aplicados

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. La prueba es una actividad en la cual, un sistema o componente, es ejecutado bajo unas condiciones o requerimientos específicos, los

resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

Se exponen a continuación tres (3) casos de prueba del módulo de gestión de solicitudes siendo estos: Crear solicitud, Modificar solicitud y Eliminar solicitud. Los restantes casos de prueba se pueden visualizar en el anexo 2.

4.4.1 Diseño del Caso de prueba Crear Solicitud

Descripción General

El caso de uso inicia cuando el Solicitante ejecuta la opción **Crear Solicitud** donde compra los servicios del proveedor seleccionado, finalizando así el caso de uso.

Condiciones de Ejecución:

El solicitante debe estar autenticado en el sistema.

Secciones a probar en el Caso de Uso: Crear Solicitud

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
*SC 1:*Crear Solicitud	*EC 1.1:* Flujo Normal de Eventos	Se crea una solicitud con los servicios seleccionados	-El solicitante selecciona la opción Crear Solicitud -El sistema muestra los proveedores de servicio ordenados por orden alfabético. -Selecciona un proveedor de servicio. -El sistema muestra los servicios que brinda el proveedor seleccionado, por orden alfabético ascendentemente, el campo unidades y las opciones: • **Guardar** • **Cancelar** -El solicitante selecciona el o los servicios que desea comprar y las unidades del mismo -El sistema crea la nueva solicitud con los servicios seleccionados -Calcula y muestra el monto total a pagar -Selecciona **Guardar** -El sistema guarda la solicitud de pago

	EC 1.2: Flujo alternos de eventos 1**" Cancelar Crear Solicitud"**	Se cancela la creación de la solicitud	-El sistema muestra un mensaje indicando que será cancelada la operación. -El solicitante selecciona cancelar, se reinician las variables y se muestra la interfaz principal.

SC 1: Crear Solicitud

Id del escenario	**Escenario**	**Variable 1** proveedores de servicio	**Variable 2** Servicios que brinda el proveedor	**Variable 3** Guardar	**Variable 4** Cancelar	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1	**Flujo Normal de Eventos**	Uno de los proveedores	Uno o más servicios	Si	_____	-El sistema calcula y muestra el monto total a pagar. -Se guarda la solicitud creada	Se crea una solicitud.
EC 1.2	**Cancelar Crear Solicitud**	Uno de los proveedores	Uno de los servicios	_____	Si	El sistema muestra un mensaje indicando que será cancelada la solicitud.	Se cancela la solicitud y se muestra la interfaz principal.

4.4.2 Diseño del Casos de prueba Modificar Solicitud

Descripción General

El caso de uso inicia cuando el Solicitante ejecuta la opción **Modificar Solicitud** donde se modifican los servicios de la solicitud y guardan los nuevos cambios, finalizando así el caso de uso.

Condiciones de Ejecución:

El solicitante debe estar autenticado en el sistema.

Secciones a probar en el Caso de Uso: Modificar Solicitud

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
*1:*Modificar Solicitud	*EC 1.1:* Flujo Normal de Eventos	Se modifica el servicio seleccionado por el solicitante	-El solicitante selecciona la opción: • **Mis solicitudes activadas** -El sistema muestra un listado de todas las solicitudes realizadas y para cada una de estas una opción: • **Detalle** -Selecciona la opción **Detalle** de la solicitud deseada -El sistema muestra todo los servicios contratados por dicha solicitud y la opción: • **Modificar Solicitud** -Selecciona la opción **Modificar Solicitud** -El sistema muestra un listado con todos los servicios contratados y para cada uno de estos la opción: • **Eliminar** -También se muestra un listado con todos los servicios de SAIME y para cada uno la opción: • **Contratar** -Y las opciones: • **Guardar solicitud** • **Cancelar** -El Cliente selecciona la opción **Guardar Solicitud** -El sistema guarda la solicitud
	EC 1.2: Flujo	Se cancela la acción de modificar solicitud	-El sistema muestra un mensaje de

	Alternativo de eventos **"Cancelar Acción "**		confirmación y las opciones: • **Aceptar** • **Cancelar** -El cliente selecciona la opción **Aceptar** -Se cancela la acción de modificar solicitud
	EC 2.2: Flujo Alternativo de Eventos **"Cancelar Eliminación"**	Se cancela la acción de eliminar servicio	-No se elimina el servicio
3- Contratar Servicio	*EC 3.3*: Flujo Normal de Eventos	Se adiciona el servicio	-El sistema añade el servicio a la lista de servicios contratado por la solicitud

SC 2: Modificar Solicitud

Id del escenario	Escenario	Variable 1 Mis solicitudes activas	Variable 2 Detalles	Variable 3 Modificar Solicitud	Variable 4 Guardar Solicitud	Variable 5 Cancelar	Respuesta del Sistema	Resultado de la Prueba
EC 2.1	Flujo Normal de Eventos	si	si	si	si		El sistema guarda la solicitud	Se guarda la solicitud
	Flujo alterno Cancelar Eliminación	si	si	si		si	Se cancela la acción de modificar solicitud	No se modifica la solicitud

Diseño de Casos de prueba correspondiente al Caso de Uso: Eliminar Solicitud

Descripción General

El caso de uso inicia cuando el Solicitante ejecuta la opción **Eliminar Solicitud** donde se elimina la solicitud y guardan los nuevos cambios, finalizando así el mismo.

Condiciones de Ejecución:

El Solicitante debe estar autenticado en el sistema.

Secciones a probar en el Caso de Uso: Eliminar Solicitud Activa

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1- Eliminar Solicitud Activa	*EC 1.1:* Flujo Normal de Eventos	El Solicitante entra a la aplicación para eliminar una solicitud	-El Solicitante entra a la aplicación y selecciona la opción: • **Mis solicitudes activas** -El sistema muestra una interfaz el listado de todas las solicitudes activas y para cada una de estas la opción: • **Eliminar Solicitud** -El Solicitante selecciona la opción **Eliminar Solicitud** para la solicitud deseada -El sistema muestra un mensaje de confirmación y las opciones: • **Aceptar** • **Cancelar** -El Solicitante selecciona la opción **Aceptar** -El sistema elimina la solicitud
	EC 1.2: Flujo Alternativo de Eventos 2 **Cancelar acción**	Se cancela la acción de eliminar solicitud	No se elimina la solicitud

SC 1: Eliminar Solicitud Activa

Id del escenario	Escenario	**Variable 1** Eliminar Solicitud	**Variable 2** Aceptar	**Variable 3** Cancelar	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	**Flujo Normal de Eventos**	**si**	**si**	_____	Se elimina la solicitud	Se elimina la solicitud del listado de solicitudes realizadas
EC 1.2	**Cancelar acción**	**si**	_____	**si**	Se cancela la acción de eliminar solicitud	No se elimina la solicitud

La aplicación de las pruebas al software se dividió en tres iteraciones. La dos (2) primeras fueron realizadas por el Grupo de Calidad del CISED, donde en la primera se aplicaron pruebas funcionales sobre las interfaces de los casos de uso que forman parte de los módulos de Autenticación y Solicitudes, encontrándose cinco (5) casos de uso con dificultades y veinte (20) no conformidades. La segunda iteración se realizó sobre los módulos Autenticación, Solicitudes y Pagos, donde se encontraron cuatro (4) casos de uso con dificultades y quince (15) no conformidades. La tercera iteración de pruebas fue realizada por Calisoft, y en la misma se detectaron seis (6) no conformidades de impacto bajo, emitiéndose así el acta de liberación del proyecto expuesta en el Anexo 3. En la tercera iteración se le dan cumplimiento a las no conformidades encontradas en las iteraciones anteriores.

Después de todas las pruebas realizadas se puede indicar que la gestión de los procesos de autenticación, solitudes y pagos de los clientes con el proveedor de servicio SAIME se clasifican de bien, ya que se le dieron respuesta a todas las no conformidades encontradas durante las tres (3) iteraciones de pruebas realizadas al sistema. Se realizó un buen control de todas las actividades, dándole seguridad al sistema. Se puede concluir que la creación del software es factible ya que el tiempo de desarrollo fue aproximadamente 7 meses, con un alto esfuerzo, por lo que se puede clasificar de moderado.

4.5 Conclusiones parciales

Después de todas las pruebas realizadas se puede concluir que la gestión de los procesos de autenticación, solitudes y pagos de los clientes con el proveedor de servicio SAIME se clasifican de bien, ya que se le dieron respuesta a todas las no conformidades encontradas durante las dos (2) primeras iteraciones. En este sentido se considera que el sistema se encuentra listo, pues se logró obtener una primera versión funcional del mismo; aunque se debe seguir mejorando en posteriores iteraciones durante el proceso de desarrollo.

Conclusiones generales

Una vez finalizada la investigación científica se concluye que:

- La realización del análisis y diseño de los módulos de Autenticación, Solicitudes y Pagos del Sistema de Integración Bancaria (SIB), permitió identificar correctamente las funcionalidades a implementar para dar solución al problema planteado.
- La implementación del sistema fue realizada satisfactoriamente, utilizando para ello herramientas y tecnologías libres en la mayoría de sus casos, acorde a las estrategias de la Universidad respecto a este tipo de software.
- Las pruebas realizadas a la aplicación obtenida permitió validar el correcto funcionamiento del sistema, logrando cumplir de este modo el objetivo de la investigación.

Recomendaciones

Una vez vencidos los objetivos de esta investigación, y teniendo en cuenta las experiencias obtenidas a lo largo del desarrollo del sistema, se recomienda lo siguiente:

- Desarrollar un sistema de pagos electrónico que le permita al cliente realizar el pago de sus solicitudes de servicio desde la aplicación, sin tener que realizar el pago en el banco u oficina del SAIME.

- Implementar un sistema de autenticación basado en tarjetas inteligentes, dando la posibilidad que en un futuro, los usuarios que cuenten con cédulas electrónicas, puedan utilizar este tipo de autenticación.

Referencias bibliográficas

1. msdn.microsoft.com. [En línea] [Citado el: 5 de Marzo de 2011.] http://www.msdn.microsoft.com/es-es/library/syf5yeat.aspx.

2. rediris.es. [En línea] [Citado el: 4 de Abril de 2011.] http://www.rediris.es/cert/doc/unixsec/node14.html.

3. **Real Academia Española.** *Diccionario de la lengua española.*

4. avalonps. [En línea] [Citado el: 4 de Enero de 2011.] http://www.avalonps.com/serv_ecom_tpv.asp.

5. Solicitud.¿Qué es una solicitud? [En línea] [Citado el: 10 de Enero de 2011.] http://www2.ull.es/docencia/cv/solicitud.htm.

6. **Dante, G. P.** *Gestión de la Información: Dimensiones e Implementación para el éxito organizacional.* Argentina : Nuevo Pharadigma, 2004.

7. EVA. [En línea] [Citado el: 27 de Febrero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=2241.

8. php.net. [En línea] [Citado el: 6 de Marzo de 2011.] http://mx.php.net/manual/es/history.php.php .

9. **Gutierrez, J.** Lenguajes y Sistemas Informáticos . [En línea] [Citado el: 21 de Enero de 2011.] http://www.lsi.us.es/-javierj/investigacion_ficheros/Framework.pdf.

10. **Potencier, Fabien.** *Symfony la guia definitiva.* 2008.

11. **datos, Sistema gestor de base de.** CAVSI. [En línea] [Citado el: 20 de Enero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

12. Linux-es. [En línea] [Citado el: 3 de Febrero de 2011.] http://www.linux-es.org/node/536.

13. netbeans. [En línea] [Citado el: 15 de Febrero de 2011.] http://netbeans.org/community/releases/69/.

14. visual-paradigm. [En línea] [Citado el: 24 de Abril de 2011.] http://www.visual-paradigm.com/.

15. Presentación de Power Point. Clase práctica 3. Ingeniería de Software I . *Rational RequisitePro.* La Habana : Universidad de las Ciencias Informáticas.

Bibliografía

1. msdn.microsoft.com. [En línea] [Citado el: 5 de Marzo de 2011.] http://www.msdn.microsoft.com/es-es/library/syf5yeat.aspx.

2. rediris.es. [En línea] [Citado el: 4 de Abril de 2011.] http://www.rediris.es/cert/doc/unixsec/node14.html.

3. **Real Academia Española.** *Diccionario de la lengua española.*

4. avalonps. [En línea] [Citado el: 4 de Enero de 2011.] http://www.avalonps.com/serv_ecom_tpv.asp.

5. Solicitud.¿Qué es una solicitud? [En línea] [Citado el: 10 de Enero de 2011.] http://www2.ull.es/docencia/cv/solicitud.htm.

6. EVA. [En línea] [Citado el: 27 de Febrero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=2241.

7. php.net. [En línea] [Citado el: 6 de Marzo de 2011.] http://mx.php.net/manual/es/history.php.php .

8. **Gutierrez, J.** Lenguajes y Sistemas Informáticos . [En línea] [Citado el: 21 de Enero de 2011.] http://www.lsi.us.es/-javierj/investigacion_ficheros/Framework.pdf.

9. **Potencier, Fabien.** *Symfony la guia definitiva.* 2008.

10. **datos, Sistema gestor de base de.** CAVSI. [En línea] [Citado el: 20 de Enero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

11. Linux-es. [En línea] [Citado el: 3 de Febrero de 2011.] http://www.linux-es.org/node/536.

12. netbeans. [En línea] [Citado el: 15 de Febrero de 2011.] http://netbeans.org/community/releases/69/.

13. visual-paradigm. [En línea] [Citado el: 24 de Abril de 2011.] http://www.visual-paradigm.com/.

14. Presentación de Power Point. Clase práctica 3. Ingeniería de Software I . *Rational RequisitePro.* La Habana : Universidad de las Ciencias Informáticas.

15. *Conferencias de Ingeniería de Software I.* s.l. : Universidad de las Ciencias Informáticas, 2010.

16. *Conferencias de ingenieria de Software II.* 2010.

17. Desarrollo Web. [En línea] [Citado el: 8 de Enero de 2011.] http://www.desarrolloweb.com/manueales/21/.

18. **Alvarez, Angel Miguel.** Qué es cada tecnología . [En línea] [Citado el: 20 de Enero de 2011.] http://www.desarrolloweb.com/manuales/15/.

19. **Foundation, A. S.** *The Apache Software Foundation.* 2010.

20. **Garcia, J.** IngenieroSoftware.com. [En línea] [Citado el: 2 de Marzo de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php/.

21. **Group, Object Management.** Unified Modeling Language. [En línea] [Citado el: 5 de Enero de 2011.] http://www.uml.org/.

22. **Jacobson, I, Booch, G y Rumbaugh, J.** *"El Proceso Unificado de Desarrollo de software".* s.l. : Addison-Wesley, 2000.

23. **Larman, C.** Introducción al analisis y diseño orientado a objetos. *UML y Patrones.* La Habana : Felix Varela, 2004.

24. **Pressman, R.** *Ingeniería del software. Un enfoque práctico.* 2005.

25. **Potencier, Fabier.** librosweb. [En línea] 2008. [Citado el: 9 de Abril de 2011.] http://www.librosweb.es/symfony_1_1.

26. acm. [En línea] [Citado el: 2 de Abril de 2011.] http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html.

Glosario de términos

Apache: servidor HTTP de código abierto para plataformas UNIX, Windows entre otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

AJAX: acrónimo de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos). Es una técnica de desarrollo web para crear aplicaciones interactivas.

Biometría: es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o físicos intrínsecos.

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

FTP: es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor.

HTML: siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

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

Internet: es una red de redes que permite la interconexión descentralizada de computadoras a través de un conjunto de protocolos denominado TCP/IP.

Java: es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

MySQL: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Glosario de términos

Oracle: es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

Password: es una serie secreta de caracteres que permite a un usuario tener acceso a un archivo, a un ordenador, o a un programa.

PIN: número de identificación personal.

TCP/IP: es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).

Usuario: es la persona que utiliza o trabaja con algún objeto o que es destinataria de algún servicio público, privado, empresarial o profesional.

Web: el sistema de documentos (o páginas web) interconectados por enlaces de hipertexto, disponibles en Internet.

Anexos

Anexo 1: Descripciones de casos de Uso

Tabla 6: Descripción del caso de uso Autenticar usuario (Elaboración propia).

Caso de Uso:	**Autenticar Usuario**
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el Usuario entra a la aplicación para autenticarse, finalizando así el caso de uso
Precondiciones:	-
Referencias	-
Prioridad	**Crítico**
Flujo Normal de Eventos	
Sección "Autenticar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1- El Usuario entra a la aplicación y elije la opción **Autenticarse**	1.1- El sistema muestra la interfaz principal con los siguientes campos a llenar: • **Usuario o E-mail** • **Contraseña** Un campo Opcional para recordar la contraseña: • **Recordar** Y las opciones: • **Entrar** • **¿Desea Registrarse?** • **¿Ha olvidado su contraseña?**
2- El usuario inserta los datos y opción: 2a) **Entrar**, continua en la **Sección "Autenticar Usuario"** parte 2.1. 2b) **¿Desea Registrarse?** ver **Flujo Alterno 1** 2c) **¿Ha olvidado su contraseña?** ver	2.1- El sistema valida que los campos no estén vacios. En caso contrario, ver **Flujo Alterno 3 "Campos Vacíos"** 2.2- El sistema valida también que no existan datos incorrectos. En caso contrario ver **Flujo Alterno 4 "Datos Incorrectos".**

Flujo Alterno 2	2.3- Se muestra la interfaz con todas las opciones que brinda la aplicación según su rol.

Prototipo de Interfaz

Flujos Alternos	
Flujo Alterno 1 "Desea Registrarse"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra una interfaz donde puede registrarse como cliente
Flujo Alterno 2 "Ha olvidado su contraseña"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra la interfaz para realizar la operación de cambiar contraseña
Flujo Alterno 3 "Campos Vacíos"	

Acción del Actor	Respuesta del Sistema
	1- El sistema muestra un mensaje notificando al usuario que ha dejado campos vacíos.
Flujo Alterno 4"Datos Incorrectos"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra un mensaje notificando al usuario que los datos son incorrectos
Poscondiciones	Quedan autenticado en usuario

Tabla 7: Descripción del caso de uso Crear pago excedente (Elaboración propia).

Caso de Uso:	**Crear pago excedente.**
Actores:	Operador de Oficina.
Resumen:	El caso de uso se inicia cuando el operador entra al sistema para **Crear pago excedente**, a partir de las solicitudes cuyo pago es mayor que el que se necesita para cubrir el monto total, se genera el pago excedente, finalizando así el caso de uso.
Precondiciones:	El Operador de Oficina debe estar autenticado en el sistema
Referencias	**RF 12**
Prioridad	
Flujo Normal de Eventos	
Sección "Crear pago excedente"	
Acción del Actor	**Respuesta del Sistema**
1- El Operador selecciona la opción **Crear pago excedente.**	1.1-El sistema muestra una interfaz con las solicitudes cuyo pago es mayor al necesario para cubrir el monto total. Y las opciones: • **Aceptar** • **Cancelar**
2- Selecciona las solicitudes a las que se les desea crea pago excedente y la opción:	2.1- Genera pago excedente asociado a la solicitud.

2a) **Aceptar**, continua en la **Sección "Crear pago excedente"** parte 2.1. 2b) **Cancelar,** ver **Flujo Alterno "Cancelar Acción".**	
Prototipo de Interfaz	
Flujos Alternos" Cancelar Acción"	
Acción del Actor	**Respuesta del Sistema**
	1-El sistema muestra un mensaje que indica que la acción será cancelada
2-El Operador selecciona: 2a) Cancelar Acción, se muestra la interfaz principal.	
Prototipo de Interfaz	
Poscondiciones	Se crea pago excedente

Tabla 8: Descripción del caso de uso Registrar pago ordinario (Elaboración propia).

Caso de Uso:	**Registrar pago ordinario**
Actores:	Operador de oficina
Resumen:	El caso de uso inicia cuando el Operador de oficina registra el pago de la solicitud que realiza el usuario, finalizando así el mismo.
Precondiciones:	El Operador de oficina debe estar autenticado en el sistema.
Referencias	**RF 8, RF 8.1, RF 8.4** al **RF 8.7.2.**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Operador de Oficina selecciona la opción **Registrar Pago Ordinario.**	1.1- El sistema recibe una ráfaga bancaria con los datos: • **BCO** – Banco y Sucursal donde se realizó el depósito • **CSS** – Código de Seguridad del SAIME • **NTR** – Número de transacción del día • **FEC** – Fecha de pago

	• **HRM** – Hora militar y minutos del pago • **CID** – Cédula del depositante • **CIB** – Cédula del beneficiario • **MRT** – Monto recaudado total • **CSP** – Código de seguridad de la PIdeS
	1.2- El sistema verifica que los datos estén correctos, ver caso de uso incluido (**"Validar CSS"**).
	1.3- El sistema busca y muestra una interfaz con todas las solicitudes efectuadas según la cédula especificada en CIB:
3- El Operador de Oficina selecciona la solicitud que desea.	2.1- El sistema muestra la solicitud con el monto a pagar.
4- El Operador de Oficina comprueba el monto pagado y marca la solicitud como pagada y selecciona la opción: 3a) **Aceptar**, continúa con el flujo normal de eventos parte 3.1. 3b) **Cancelar**, ver **Flujo Alterno 1 "Cancelar Registrar Pago Ordinario"**	3.1- El sistema muestra una interfaz con los servicios que brinda el proveedor por orden alfabético ascendentemente, el campo unidades y las opciones: • **Guardar** • **Cancelar**
5- El Operador de Oficina selecciona él o los servicios que desea comprar y las unidades del mismo y la opción. 5a) **Guardar,** continúa con el flujo normal de eventos en la parte 4.1. 4b) **Cancelar,** ver **Flujo Alterno 2 "Cancelar Crear Solicitud"**	4.1- El sistema crea la nueva solicitud con los servicios seleccionados.
	4.2- El sistema calcula el monto de la solicitud de pago.
	4.3- El sistema comprueba que el monto total calculado es inferior al MRT capturado o igual. En caso contrario, ver caso de uso **("Crear Pago Extraordinario")**

	4.4- El sistema guarda la solicitud de pago con las unidades de servicios seleccionadas y el monto total calculado.
	4.5- El sistema guarda los metadatos del usuario proveedor de servicios que registra la acción. • **Fecha del registro.** • **Hora del registro.** • **No. De cédula del usuario proveedor de servicios que registra la acción.**
	4.6- El sistema comprueba que el monto total calculado es inferior al MRT capturado.
6- El Operador de Oficina registra pago excedente a la solicitud.	5.1- El sistema notifica al cliente por sms o correo electrónico que tiene pago excedente.
Prototipo de Interfaz	
Flujos Alternos	
Flujo Alterno 1 "Cancelar Registrar Pago Ordinario".	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra un mensaje indicando que será cancelado Registrar Pago Ordinario.
2- El Operador de Oficina selecciona: 2a) Cancelar Registrar Pago Ordinario y se muestra la interfaz principal.	
Flujo Alterno 2 "Cancelar Crear Solicitud"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra un mensaje indicando que será cancelada la Solicitud.
2- El Operador de Oficina selecciona: 2a) Cancelar la solicitud, se reinician las variables de la creación de la solicitud y se muestra la interfaz principal.	

Prototipo de Interfaz	
Poscondiciones	Queda registrado el pago ordinario.

Tabla 9: Descripción del caso de uso Registrar pago extraordinario (Elaboración propia).

Caso de Uso:	**Registrar Pago Extraordinario**
Actores:	Operador de Oficina
Resumen:	El caso de uso inicia cuando el Operador de Oficina registra el pago extraordinario de la solicitud que realiza el usuario, finalizando así el caso de uso.
Precondiciones:	El Operador de Oficina debe estar autenticado en el sistema y el pago extraordinario tiene que estar creado.
Referencias	**RF10.4; RF10.4.1; RF10.4.1.1; RF10.4.1.2; RF10.4.1.3; RF10.4.1.4; RF10.4.1.5.**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7- El Operador de Oficina selecciona la opción **Registrar Pago Extraordinario.**	1.4- El sistema recibe una ráfaga bancaria con los datos: • **BCO** – Banco y Sucursal donde se realizó el depósito • **CSS** – Código de Seguridad del SAIME • **NTR** – Número de transacción del día • **FEC** – Fecha de pago • **HRM** – Hora militar y minutos del pago • **CID** – Cédula del depositante • **CIB** – Cédula del beneficiario • **MRT** – Monto recaudado total • **CSP** – Código de seguridad de la PIdeS
	1.5- El sistema verifica que los datos estén correctos, ver caso de uso incluido (**"Validar CSS"**).
	1.6- El sistema busca y muestra una interfaz con todas las solicitudes efectuadas según la cédula especificada en CIB.

2- El Operador de Oficina selecciona la solicitud cuyo monto se corresponda con el especificado en MRT.	2.1- El sistema muestra la solicitud con el monto a pagar y las opciones: • **Aceptar** • **Cancelar**
3-El Operador de Oficina comprueba el monto pagado y marca la solicitud como pagada y selecciona la opción: 3a) **Aceptar**, continúa con el flujo normal de eventos en la parte 3.1. 3b) **Cancelar,** ver **Flujo Alterno 1 "Cancelar Registrar Pago Extraordinario"**	3.1 El sistema guarda los metadatos del usuario proveedor de servicios que registra la acción. • **Fecha del registro.** • **Hora del registro.** • **No. De cédula del usuario proveedor de servicios que registra la acción.**
Prototipo de Interfaz	
Flujos Alternos	
Flujo Alterno 1 "Cancelar Registrar Pago Extraordinario"	
Acción del Actor	**Respuesta del Sistema**
	3- El sistema muestra un mensaje indicando que será cancelada el registro de pago extraordinario.
4- El Operador de Oficina selecciona: 2a) Cancelar el registro de pago extraordinario y se muestra la interfaz principal.	
Prototipo de Interfaz	
Poscondiciones	Queda registrado el pago extraordinario.

Tabla 10: Descripción del caso de uso Validar CSS (Elaboración propia).

Caso de Uso:	**Validar CSS**
Actores:	Sistema
Resumen:	El caso de uso inicia cuando el sistema recibe los datos de la ráfaga bancaria y valida el CSS finalizando así el caso de uso.
Precondiciones:	El sistema debe recibir los datos de la ráfaga bancaria.

Referencias	**RF8; RF8.1; RF8.2; RF8.2.1; RF8.2.2; RF8.2.3; RF8.2.4; RF8.2.5; RF8.3; RF8.3.1; RF10; RF10.1; RF10.2; RF10.2.1; RF10.2.2; RF10.2.3; RF10.2.4; RF10.2.5; RF10.3; RF10.3.1**
Prioridad	**Auxiliar**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema determina los datos esenciales: De la Pldes[16]: • **Número de la planilla.** • **Fecha de emisión.** • **Monto total a cancelar.** • **CSP.** Del servicio pagado: • **Identificador del servicio pagado.** Del pago efectuado: • **Número del voucher.** • **Fecha del pago.** • **Nro. De la transacción.** • **Nro. Del banco.** • **Nro. De la sucursal.** • **Cédula del depositante.** • **Cédula del beneficiario.** • **CSS.**
	2- El sistema obtiene la cadena de seguridad vigente de acuerdo a la FEC para la sucursal especificada en BCO.
	3- El sistema mezcla las cadenas de los datos esenciales y la cadena de seguridad.
	4- El sistema obtiene la función B64.
	5- El sistema obtiene la función B32.
	6- El sistema compara el CSS de la ráfaga con el CSS obtenido.
	7- El sistema informa al usuario que la ráfaga es correcta y notifica un **Semi OK**. En caso

[16] **Pldes**: Planilla Informativa de Servicios.

	contrario, ver **Flujo Alterno 1 "Ráfaga incorrecta"**
Flujos Alternos	
Flujo Alterno 1 "Ráfaga incorrecta"	
Acción del Actor	**Respuesta del Sistema**
	a. El sistema notifica al usuario si la ráfaga es incorrecta, notifica un **No OK** y va a la interfaz principal.
Prototipo de Interfaz	
Poscondiciones	Queda validado el CSS.

Tabla 11: Descripción del caso de uso Modificar solicitud activa (Elaboración propia).

Caso de Uso:	**Modificar Solicitud Activa**
Actores:	Solicitante
Resumen:	El caso de uso inicia cuando el Solicitante ejecuta la opción **Modificar solicitud activa** donde se modifican y eliminan los servicios de la solicitud y guardan los nuevos cambios, finalizando así el mismo.
Precondiciones:	El Solicitante debe estar autenticado en el sistema.
Referencias	**RF2**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
2. El Solicitante selecciona la opción **Mis solicitudes activas.**	• El sistema muestra una interfaz con todas las solicitudes activas alfabéticamente de forma ascendente y las opciones: **4. Más detalles** **5. Eliminar solicitud**
3. El Solicitante selecciona la opción: 2a) **Más detalles**, continua el flujo normal de los eventos 2.1**.** 2b) **Eliminar solicitud**, ver **Sección 1 "Eliminar**	2.1- Muestra los detalles de la solicitud: El número de la solicitud, la fecha de creación, el estado de la solicitud. Los servicios contratados en esa solicitud.

solicitud".	
	2.2- Muestra las opciones: • **Modificar solicitud** • **Eliminar solicitud** • **Mis solicitudes activas**
4. El Solicitante selecciona la opción que desee: 3a) **Modificar solicitud**, continua con el flujo normal de eventos en la parte 3.1. 3b) **Eliminar solicitud**, ver **Sección 1 "Eliminar solicitud".** 3c) **Mis solicitudes activas,** retorna al flujo normal de los eventos 1.1	3.1- Muestra la solicitud con los servicios contratados y la lista de los otros servicios. Los botones: **5- Contratar servicios seleccionados** **6- Eliminar servicios seleccionados** **7- Guardar solicitud** Los vínculos: • Mis solicitudes activas • Eliminar solicitud actual • Cancelar
5. El Solicitante selecciona la opción que desee: 4a) Selecciona servicios y presiona **Contratar servicios seleccionados**, continua con el flujo normal de eventos en la parte 4a.1. 4b) Selecciona servicios contratados y presiona **Eliminar servicios seleccionados**, continua con el flujo normal de eventos en la parte 4b.1. 4c) **Guardar solicitud,** continua con el flujo normal de eventos en la parte 4c.1. 4d) **Mis solicitudes activas,** retorna al flujo normal de los eventos 1.1 **4e) Eliminar solicitud actual,** ver **Sección 1 "Eliminar solicitud".** **4f) Cancelar** ver **Sección 2 "Cancelar acción"**	4a.1- Añade los servicios seleccionados a la solicitud y calcula el costo de la solicitud. 4b.1- Elimina los servicios seleccionados de la solicitud y calcula el costo de la solicitud. 4c.1- Guarda la solicitud y retorna a la pantalla de los detalles de la solicitud.
Sección 1"Eliminar solicitud"	
Acción del Actor	**Respuesta del Sistema**
	1.1- Se muestra una ventana preguntándole al

	usuario si desea eliminar la solicitud y los botones: **4.2- Aceptar** **4.3- Cancelar**
1.2- Selecciona la acción 1.2a) **Aceptar**, continua con el flujo normal de eventos en la parte 2a.1. 1.2b) **Cancelar,** retorna al paso 3.1 del flujo básico	1.2a.1-Elimina la solicitud.
Sección 2 "Cancelar acción"	
Acción del Actor	**Respuesta del Sistema**
	2- Se muestra una ventana preguntándole al usuario si desea eliminar la solicitud y los botones: **3. Aceptar** **4. Cancelar**
2.1 El solicitante selecciona: 2.1a) Aceptar 2.1b) Cancelar, retorna al paso 3.1del flujo básico.	2a.1- Se cancela la acción retorna al paso 1.1.
Poscondiciones	Queda modificada la solicitud.

Tabla 12: Descripción del caso de uso Eliminar solicitud activa (Elaboración propia).

Caso de Uso:	**Eliminar Solicitud Activa**
Actores:	Solicitante
Resumen:	El caso de uso inicia cuando el Solicitante ejecuta la opción **Eliminar Solicitud Activa** donde se elimina la solicitud y guardan los nuevos cambios, finalizando así el mismo.
Precondiciones:	El Solicitante debe estar autenticado en el sistema.
Referencias	**RF3**
Prioridad	**Crítico**
Flujo Normal de Eventos	
""	

Acción del Actor	**Respuesta del Sistema**
8- El Solicitante selecciona la opción **Eliminar Solicitud Activa.**	1.2- El sistema muestra una interfaz con todas las solicitudes de pago de servicios en el estado en que se encuentran ordenadas alfabéticamente de forma ascendente y las opciones: • **Aceptar** • **Cancelar**
2- El Solicitante selecciona las solicitudes que desea eliminar y selecciona la opción: 2a) **Aceptar**, continua con el flujo normal de eventos en la parte 2.1. 3b) **Cancelar**, ver **Flujo Alterno 2 "Cancelar Acción"**	2.1- El sistema verifica que la solicitud no tenga ningún servicio pagado. En caso contrario ver **Flujo Alterno 1 "Servicio pagado".**
	2.2- El sistema elimina las solicitudes seleccionadas.
Prototipo de Interfaz	
Flujos Alternos	
Flujo Alterno 1 "Servicio Pagado"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra un mensaje indicando que el cambio no se puede realizar ya que el servicio está pagado y regresa al paso 1.1.
Flujo Alterno 2 "Cancelar Acción"	
Acción del Actor	**Respuesta del Sistema**
	5- El sistema muestra un mensaje indicando que será cancelada la acción.
6- El Solicitante selecciona: 2a) Cancelar la acción y muestra la interfaz principal.	
Prototipo de Interfaz	
Poscondiciones	Queda eliminada la solicitud.

Anexo 2: Pruebas de caja negra

2.1 Caso de Prueba de Autenticar Usuario

Descripción General

El caso de uso inicia cuando el Usuario entra a la aplicación para autenticarse, finalizando así el caso de uso

Condiciones de Ejecución:

Secciones a probar en el Caso de Uso: Autenticar Usuario

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1- Autenticar Usuario	*EC 1.1:* Flujo Normal de Eventos	El usuario entra a la aplicación para autenticarse.	-El usuario entra a la aplicación y selecciona la opción: • **Autenticarse** -El sistema muestra una interfaz con los siguientes campos a llenar: • **Usuario o E-mail** • **Contraseña** Un campo opcional para recordar la contraseña • **Recordar** Y la opción: • **Entrar** -El usuario introduce los datos y selecciona la

			opción **Entrar** -El sistema verifica que no hayan campos vacios ni datos incorrectos. -El sistema de acuerdo a los permisos que tenga el usuario le muestra una interfaz con todas las opciones que tiene la aplicación según su rol.
	EC 1.2: Flujo alternativo de Eventos 1 **Campos vacios**	El usuario no se autentica debido a que existen campos vacios.	-El sistema muestra un mensaje indicando que el usuario ha dejado campos
	EC 1.3: Flujo alternativo de Eventos 2 **Datos incorrectos**	El usuario no se autentica debido a errores en los datos	-El sistema muestra un mensaje indicando que el usuario no ha llenado los campos correctamente.

SC 1: Autenticar Usuario

Id del escenario	**Escenario**	**Variable 1** Usuario o E-mail	**Variable 2** contraseña	**Variable 3** Entrar	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1	**Flujo Normal de Eventos**	mmrondon	**********	si	El usuario se autentica	Se muestra la aplicación con las opciones que esta tiene según el rol que tenga el usuario

EC 1.2	**Campos Vacios**	mmrondon		si	El sistema muestra un mensaje	Se muestra un mensaje indicando que existen campos vacios
EC 1.3	**Datos Incorrectos**	mmrondon	+++++++	si	El sistema muestra un mensaje	Se muestra un mensaje indicando que hay datos incorrectos

2.2 Caso de Prueba del Pago Ordinario

Descripción General

El caso de uso inicia cuando el Operador de Oficina registra el pago de la solicitud que realiza el usuario, finalizando así el caso de uso.

Condiciones de Ejecución:

El Operador de Oficina debe estar autenticado en el sistema.

Secciones a probar en el Caso de Uso: Registrar Pago Ordinario

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1- Registrar pago Ordinario	*EC 1.1:*Flujo Normal de Eventos	El usuario entra los datos necesarios para registrar la solicitud de pago con las unidades de servicios seleccionadas y el monto total calculado.	-El usuario selecciona la opción **Registrar Pago Ordinario** -El sistema muestra una interfaz con los campos a llenar: • **BCO – Banco y Sucursal donde se realizó el depósito** • **CSS – Código de Seguridad del SAIME** • **NTR – Numero de transacción del día** • **FEC – Fecha de pago** • **HRM – Hora militar y minutos del pago** • **CID – Cédula del depositante** • **CIB – Cédula del beneficiario** • **MRT – Monto recaudado total** • **CSP – Código de seguridad de la PIdeS**

			Y las opciones: • **Enviar** • **Cancelar** -El usuario introduce los datos y la opción Enviar -El sistema verifica que no hayan campos vacios ni datos incorrectos. - Se muestra una interfaz con todas las solicitudes efectuadas según la cédula especificada en CIB y las opciones: • **Aceptar** • **Cancelar** -Selecciona la solicitud -Se muestra la solicitud con el monto total -El usuario comprueba el monto pagado y marca la solicitud y selecciona Aceptar -Se muestran los servicios que brinda el proveedor y las opciones: • **Guardar** • **Cancelar** -Selecciona el o los servicios que desea comprar -Se crea la nueva solicitud con los servicios seleccionados -Selecciona **Guardar** -El sistema calcula y comprueba que el monto total sea igual o inferior que el MTR calculado -El sistema guarda la solicitud de pago con los servicios seleccionado y el monto calculado - El sistema guarda los metadatos del usuario proveedor de servicios: • **Fecha del registro.** • **Hora del registro.**

		• **No. De cédula del usuario proveedor de servicios que registra la acción.** - El sistema comprueba que el monto total calculado es inferior al MRT capturado -El usuario registra pago excedente a la solicitud - El sistema notifica al cliente por sms o correo electrónico que tiene pago excedente
*EC 1.2:*Flujo alternativo de Eventos 1 **Cancelar Registrar Pago Ordinario**	Se canela la operación de Registrar Pago Ordinario	-El sistema muestra un mensaje indicando que será cancelado registrar pago ordinario -El usuario selecciona la opción cancelar y se muestra la interfaz principal.
*EC 1.3:*Flujo alternativo de Eventos 2 **Campos Vacios**	No se registra el pago ya que existen campos vacios	-El sistema muestra un mensaje notificando al usuario que ha dejado campos vacios
*EC 1.4:*Flujo alternativo de Eventos 3 **Datos Incorrectos**	No se registra el pago debido a que existen datos incorrectos	- El sistema muestra un mensaje notificando al usuario que los datos son incorrectos

	*EC 1.5:*Flujo alternativo de Eventos 4 **Cancelar Crear Solicitud**	No se crea la solicitud.	-El sistema muestra un mensaje indicando que será cancelada la solicitud - El usuario selecciona la opción cancelar, se reinician las variables y se muestra la interfaz principal.

SC 1: Registrar Pago Ordinario

Id del escenario	Escenario	Variable 1 *BCO*	Variable 2 CSS	Variable 3 NTR	Variable 4 FEC	Variable 5 HRM	Variable 6 CID	Variable 7 CIB	Variable 8 MRT	Variable 9 *CSP*
1.1	**Flujo Normal de Eventos**	*8745 y 6547*	*MWRXZT SPRTOH K*	*140*	*12012011*	*14:24*	*V15363289*	*V15731284*	*1800.00*	*18973564286*
1.2	**Cancelar Registrar Pago Ordinario**	*8745 y 6547*	*MWRXZT SPRTOH K*	*140*	*12012011*	*14:24*	*V15363289*	*V15731284*	*1800.00*	*18973564286*
		8745 y 6547	*MWRXZT SPRTOH*	*140*	*12012011*	*14:24*	*V15363289*	*V15731284*	*1800.00*	*18973564286*
1.3	**Campos Vacios**	*8745 y 6547*		*140*	*12012011*	*14:24*		*V15731284*	*1800.00*	*18973564286*
1.4	**Datos Incorrectos**	*8745 y 6547*	*MWRxZT SPRTOh K*	*140*	*12012011*	*14:24*	*V15363289*	*V35731284*	*1800.00*	*18973564286*

1.5	**Cancelar Crear Solicitud**	*8745 y 6547*	*MWRXZT SPRTOH K*	*140*	*12012011*	*14:24*	*V15363289*	*V15731284*	*1800.00*	*18973564286*

Continuación.

Variable 10 Envía	**Variable 11** Cancelar	**Variable 12** Listado se solicitudes	**Variable 13** Aceptar	**Variable 14** Cancelar	**Variable 15** Listado de servicios	**Variable 16** Guardar	**Variable 17** Cancelar	**Respuesta del Sistema**	**Resultado de la Prueba**
si	____	Una de las solicitudes	si	_____	Selecciona el o los servicios	si	____	Se notifica al cliente que tiene pago excedente	Se le envía un correo al cliente
___	si							Se muestra un mensaje indicando que será cancelara la acción	Se cancela la operación
si	_____	Una de las solicitudes	_____	si					
si	_____	Una de las solicitudes	si	_____	Selecciona el o los servicios	si	____	Se muestra un mensaje	Se muestra un mensaje indicando
si	____	Una de las solicitudes	si	_____	Selecciona el o los servicios	si	____	Se muestra un mensaje	Se muestra un mensaje indicando

si	____	Una de las solicitudes	si	_____	Selecciona el o los servicios	_____	si	Se muestra un mensaje indicando que será cancelada la solicitud	Se reinician las variable y se muestra la interfaz

2.3 CP Registrar Pago Extraordinario

Descripción General

El caso de uso inicia cuando el Operador de Oficina crea el pago extraordinario seleccionando las solicitudes detenidas por falta de pago, y se genera la planilla PIdeS., finalizando así el caso de uso.

Condiciones de Ejecución:

El Operador de Oficina debe estar autenticado en el sistema.

Secciones a probar en el Caso de Uso: Registrar Pago Extraordinario

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

C 1- Registrar pago Extraordinario	*EC 1.1:*Flujo Normal de Eventos	El sistema genera una Planilla Informativa de Servicios asociados a las solicitudes seleccionadas por el operador	-El usuario selecciona la opción **Crear Pago Extraordinario** -El sistema muestra una interfaz con las solicitudes detenida por faltante de pago y las opciones: • **Generar Plantilla** • **Cancelar** -El usuario selecciona la o las solicitudes y la opción • **Generar Plantilla** -El sistema determina los datos esenciales de la PIdeS y del servicio pagado -Obtiene la cadena de seguridad vigente para la oficina que emite la PIdeS. -Mezcla las cadenas de los datos esenciales y la cadena de seguridad. - Obtiene la función B32 y B8 y genera el CSP -Genera la PIdeS asociada a las solicitudes seleccionadas -Guarda los metadatos del usuario proveedor de servicio que registra la acción. -Muestra la PIdeS generada y brinda la posibilidad de imprimirla.

*EC 1.2:*Flujo alternativo de Eventos 1 **"Cancelar Crear Pago Extraordinario"**	Se canela la operación de Crear pago extraordinario	-El sistema muestra un mensaje indicando que será cancelada la creación de pago extraordinario. -El usuario selecciona la opción cancelar y se

SC 1: Registrar Pago Extraordinario

Id del escenario	Escenario	Variable 1 *(Solicitudes)*	Variable 2 *(Generar Plantilla)*	Variable 3 *(Cancelar)*	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	**Flujo Normal de Eventos**	Una o varias de las solicitudes	si	____	El sistema genera la PIdeS	Se imprime la PIdeS
EC 1.2	**Flujo alterno**	Una o varias de las solicitudes mostradas	_____	si	Se muestra un mensaje indicando que será cancelara la acción	Se cancela la acción de registrar pago extraordinario

Anexo 3: 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'

Anexo 4: Diagramas de clases de diseño

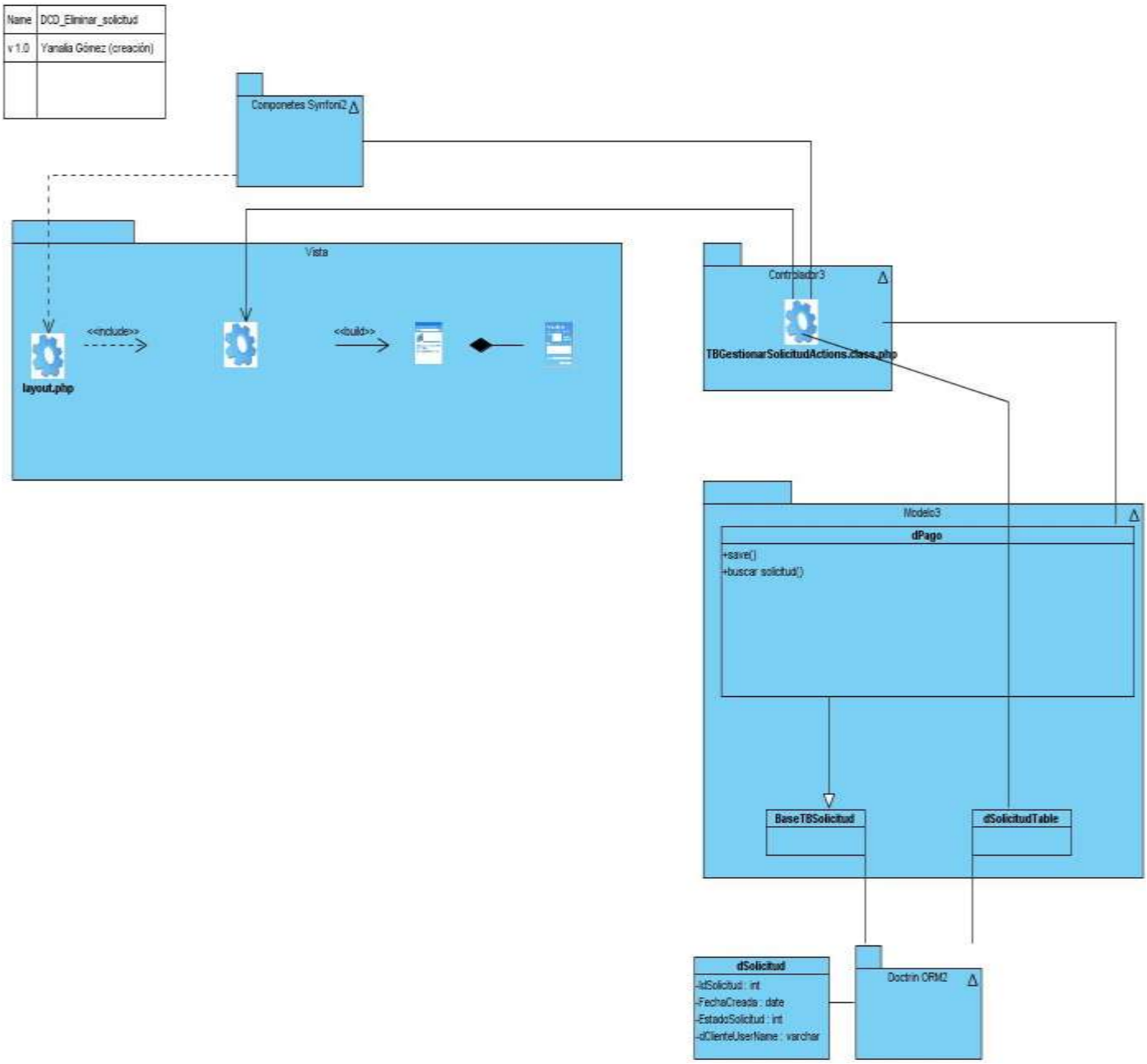


Figura 18: Diagrama de clases del diseño Eliminar solicitud (Elaboración propia).

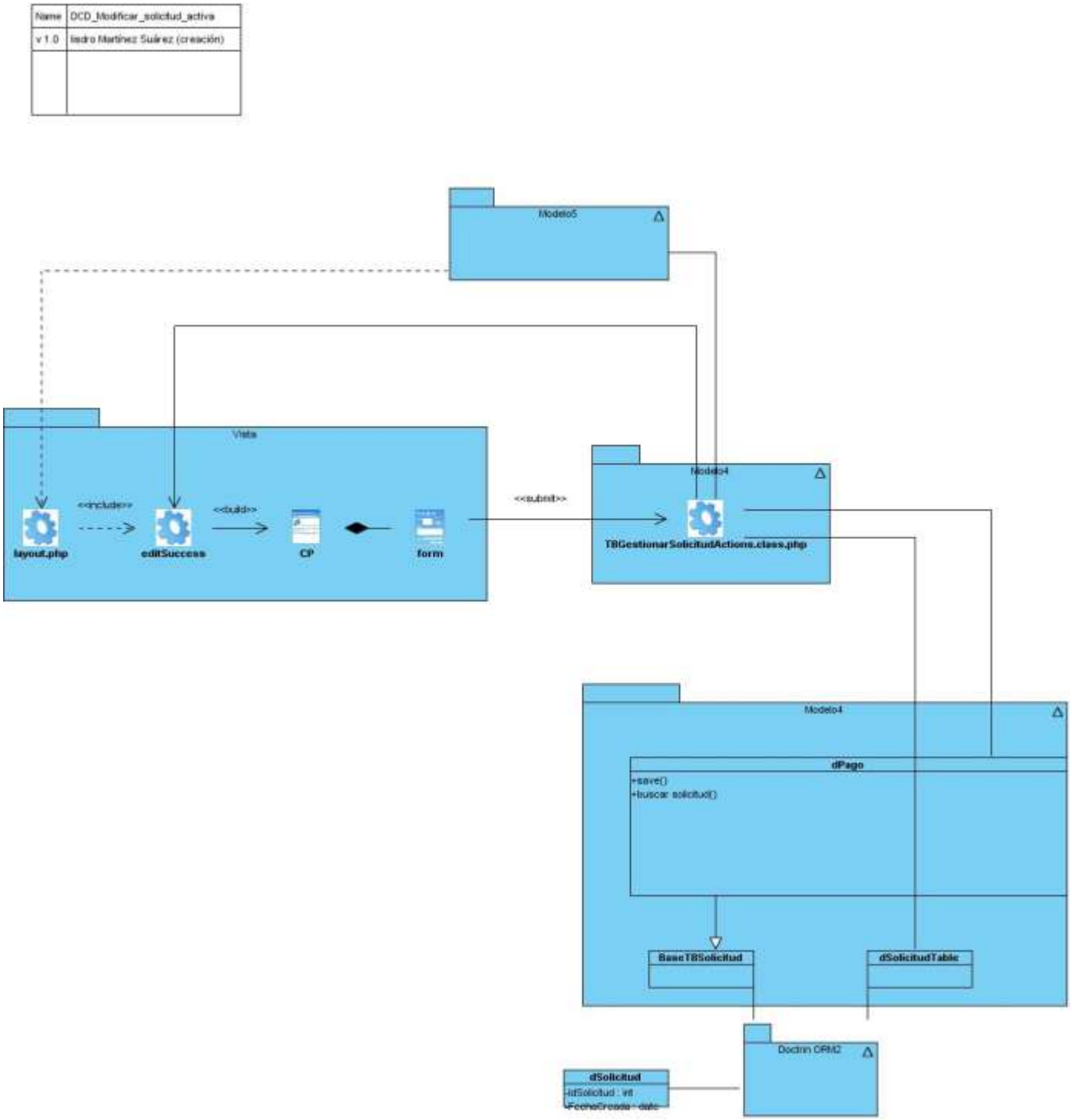


Figura 19: Figura 18: Diagrama de clases del diseño Modificar solicitud (Elaboración propia).

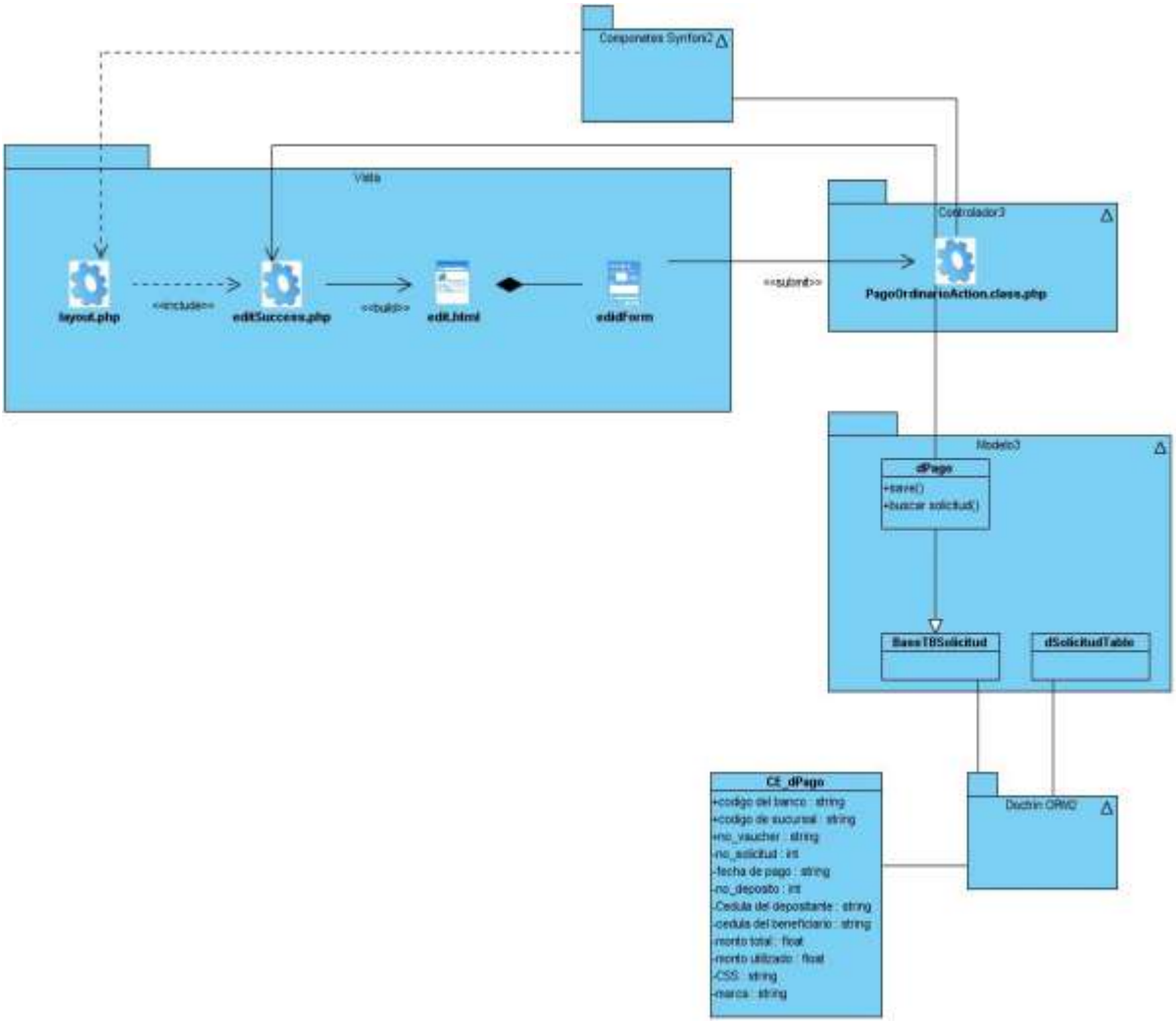


Figura 20: Figura 18: Diagrama de clases del diseño Registrar pago ordinario (Elaboración propia).