# Universidad de las Ciencias Informáticas
## Facultad 4


**Título:** **SISTEMA DE CONTABILIDAD FINANCIERA PARA LA ACTIVIDAD PRESUPUESTADA DE LAS FAR. MÓDULO CAJA**

**Trabajo de Diploma para optar por el título de**

**Ingeniero en ciencias Informáticas**

**Autores:**

Gretter Mora Viera

Yosveni Escalona Escalona

**Tutor:**

Ing. Yunei López Lugo

Ciudad de La Habana, junio de 2007

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a _____ de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

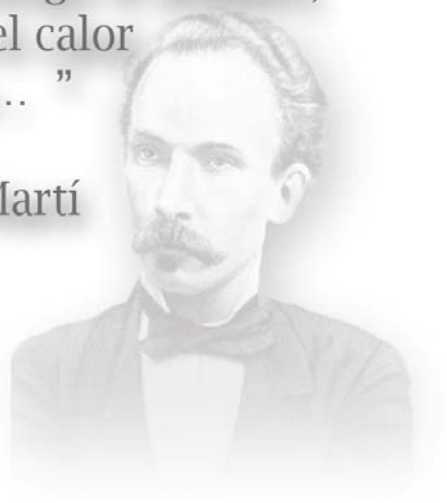
| Gretter Mora Viera | Yosveni Escalona Escalona | Ing. Yunei López Lugo |
|---|---|---|
| _____ | _____ | _____ |
| Firma del Autor | Firma del Autor | Firma del Tutor |

DATOS DE CONTACTO

Tutor: Yunei López Lugo
Profesión: Ingeniera Informática.
Años de experiencia: 3
Años de graduada: 3.

"Día llegará en que pueda llevar consigo el hombre,
como el tiempo en un reloj, la luz, el calor
y la fuerza en un aparato diminuto… "

José Martí



**Agradecimientos**

Agradecemos a todas las personas que nos han tendido la mano a lo largo de este extenso camino, así como a las que nos han permito estar aquí. Gracias a todo el que creyó en nosotros, por transmitirnos la fuerza y voluntad necesarias para acrecentarnos cuando todo se tornó oscuro. Para todos ellos nuestra más sincera Gratitud.

**Dedicatoria**

Dedico este trabajo a mis padres, mi hermana y mi abuela, por forjarme desde pequeña e inculcarme todo lo que soy. Este es el fruto de lo que ustedes sembraron en mí, el resultado de su trabajo, confianza y sacrificio de años.

Gretter

Dedico esta tesis a mis padres y a mis abuelos por darme todo el amor del mundo, por apoyarme, sacrificarse y forjar lo que soy hoy, gracias a ellos de corazón por creer en mí.

Este trabajo es también para mi hermano, porque me dio todo su amor cada día de su efímera vida. Esto es para ti Yoel

Yosveni

**Resumen**

El desarrollo de la informática en Cuba ha experimentado enormes pasos de avance en los últimos tiempos, haciendo extensivo este conocimiento a todos los rincones del país. La incorporación de dicha rama en las empresas e instituciones cubanas hace evidente tal afirmación.

El Ministerio de las Fuerzas Armadas Revolucionarias, es una entidad que no se ha quedado atrás con este progreso, pues se ha propuesto construir un software que gestione todos sus procesos, el cual estará compuesto por otros subsistemas. La presente investigación consiste en realizar una parte de este enorme sistema, es decir, se propone realizar el módulo caja perteneciente al subsistema de contabilidad financiera de forma tal que se automaticen estos procesos, contribuyendo así, a construir una parte de este gran software de gestión.

Se realizará una aplicación Web que automatice los procesos de caja, utilizando los lenguajes de programación PHP y Javascript, como metodología de desarrollo de software RUP, como gestor de base de datos PostgreSql y será utilizado el Visual Paradigm como herramienta Case.

**Palabras Clave**

Sistema de contabilidad Financiera, Caja, aplicación Web.

## Índice

Agradecimientos ..... I

Dedicatoria ..... II

Resumen ..... III

Introducción ..... 1

Capítulo 1. Fundamentación Teórica. ..... 4

- Introducción ..... 4
- Contabilidad Financiera ..... 4
- ¿Qué es un ERP? ..... 4
- Herramientas y Tecnologías. ..... 5
- ¿Qué es Internet? ..... 5
- Composición y Servicios de Internet ..... 5
- La Web ..... 6
- Las Aplicaciones Web ..... 6
- Lenguajes de programación para la Web ..... 6
- Javascript ..... 7
- Personal Home Page (PHP) ..... 7
- Navegador ..... 8
- Metodologías de Trabajo ..... 9
- El Proceso Unificado de Modelado. RUP ..... 9
- Lenguaje Unificado de Modelado (UML) ..... 11
- Sistemas gestores de bases de datos. ..... 12
- SQL ..... 12
- PostgreSQL ..... 12
- Herramientas Case ..... 14
- Visual Paradigm ..... 14
- Sistemas Automatizados vinculados al campo de acción ..... 14
- SABIC (Sistema Automatizado para la Banca Internacional de Comercio) ..... 14
- SIC (Sistema de información contable) ..... 15
- CREPREC (Sistema automatizado de la gestión municipal en Paraguay) ..... 15
- Conclusiones: ..... 16

Capítulo 2. Características del sistema ..... 17

- Introducción ..... 17
- Objeto de estudio ..... 17
- Problema y Situación problémica ..... 17
- Objeto de automatización. ..... 17
- Sistemas automatizados existentes ..... 18

Información que se maneja. 19
Modelo de negocio. 19
Actores y trabajadores del negocio 20
Actores del Negocio 20
Trabajadores del negocio 20
Casos de uso del negocio 20
Modelo de casos de uso del negocio 21
Diagramas de actividades por cada caso de uso del negocio 22
Descripción de los casos de uso del negocio 26
Especificación de los requisitos de software 30
Requerimientos funcionales 30
Requerimientos no Funcionales 31
Modelo del sistema. 32
Definición de los actores del sistema 32
Descripción de los casos de uso del sistema 33
Diagrama de casos de Uso 35
Casos de uso expandidos 35
Conclusiones: 60
Capitulo 3. Análisis y Diseño del sistema. 61

Introducción. 61
Análisis. 61
Modelo de clases del análisis 62
Diseño. 66
Diagramas de interacción 67
Diagramas de clases 71
Descripción de las clases 75
Diseño de la Base de Datos 89
Descripción de las tablas 90
Mecanismos de diseño 94
Mecanismo de Diseño de Acceso a Datos 94
Mecanismo de seguridad 97
Concepción de la ayuda 98
Tratamiento de errores 98
Conclusiones 99
Capitulo 4. Implementación y Prueba 100

Introducción 100
Modelo de Implementación 100
Diagrama de despliegue 100
Diagrama de componentes 101
Modelo de prueba 103
Casos de Prueba 103
Conclusiones. 107
Conclusiones 108

Recomendaciones ..... 109

Bibliografía ..... 110

Glosario ..... 112

**Introducción.**

La contabilidad es el método para organizar los movimientos económicos y financieros de las empresas y personas naturales ofreciendo una visión detallada y precisa de su situación patrimonial. Se remonta desde tiempos muy antiguos, las primeras civilizaciones que surgieron sobre la tierra tuvieron que hallar la manera de dejar constancia de determinados hechos con proyección aritmética, que se producían con demasiada frecuencia y eran demasiado complejos como para poder ser conservados por la memoria. Se ha demostrado a través de diversos historiadores que en épocas bien remotas se empleaban técnicas contables que se derivaban del intercambio comercial.

Es una técnica que se ocupa de registrar, clasificar y resumir las operaciones mercantiles de un negocio con el fin de interpretar sus resultados. Por consiguiente, los gerentes o directores a través de la contabilidad se orientan sobre el curso que siguen sus negocios mediante datos contables y estadísticos. Estos datos permiten conocer la estabilidad y solvencia de las compañías, la corriente de cobros y pagos, las tendencias de las ventas, costos y gastos generales, entre otros. De manera que se puede conocer la capacidad financiera de las empresas. (TOUZETT)

Dadas sus prestaciones, la contabilidad es una técnica mundialmente utilizada y por supuesto, Cuba no está exenta de ello. A su vez, el Ministerio de las Fuerzas Armadas Revolucionarias (en lo adelante MINFAR) es una de las instituciones cubanas, (y en la que se enfoca la presente investigación) que hace uso de la contabilidad, pero de una forma diferente con respecto al resto de las empresas y entidades nacionales.

El MINFAR es poseedor de un sistema automatizado de contabilidad financiera para gestionar sus procesos contables, sistema implementado en un lenguaje de programación obsoleto (Visual FoxPro), que no responde a todas las necesidades de los usuarios y no están realizados módulos medulares para lograr un detallado y perfecto control de la contabilidad financiera tales como caja, entre otros ( los autores se detienen aquí por ser la realización de este módulo el objetivo de la presente investigación) lo que conlleva a que estos procesos se lleven manualmente, recogiéndose los datos y archivándose en papel duro. Esto trae consigo que los niveles de errores de cálculo y de elaboración de vales y documentos sean considerables y las condiciones laborales en el marco de la gestión contable no sean las ideales.

La presente investigación surge por la necesidad de dar solución a la situación anteriormente expuesta, por lo que el problema a resolver reside en: ¿Cómo erradicar los problemas de los procesos contables asociados a la actividad de Caja así como las dificultades que esto genera para las condiciones de trabajo

en las unidades de las Fuerzas Armadas Revolucionarias (en lo adelante FAR) mediante la utilización de un sistema automatizado para la gestión de dicho proceso?

Para darle solución al problema existente, se propone como objeto de estudio el proceso económico de Caja en las unidades de las FAR. Siendo el objetivo de la presente investigación diseñar e implementar una aplicación Web que automatice dicho proceso.

Los objetivos específicos en función de lo anteriormente planteado son:

- ✓ Modelar el negocio.
- ✓ Modelar el sistema.
- ✓ Realizar modelos de análisis y diseño.
- ✓ Implementar la propuesta de solución.
- ✓ Realizar las pruebas pertinentes que garanticen la calidad del sistema resultante.

De acuerdo a los objetivos que persigue la presente investigación, el campo de acción queda definido como: Sistema automatizado para la gestión de los procesos financieros de Caja en las unidades de las FAR.

La hipótesis que guiará a los autores en el desarrollo de la investigación es la siguiente: el desarrollo del sistema automatizado para la gestión de los procesos financieros de Caja en las FAR permitirá que se logre una mayor rapidez en las entregas y recibos de efectivos, gestión de documentos, emisión de vales, es decir de todos los procesos que se realizan en caja, haciéndose estos de forma automatizada y lográndose una mayor consistencia y seguridad de los datos así como un mejor control de los medios financieros.

Para llevar a cabo la presente investigación con éxito y cumplir con los objetivos propuestos, se proponen las siguientes tareas de investigación:

- ✓ Estudiar los principios que rigen las operaciones de Caja, así como la transferencia de recursos financieros para la actividad presupuestada en las Fuerzas Armadas Revolucionarias.
- ✓ Investigar acorde a las condiciones y los recursos con que cuentan las unidades militares cubanas las posibilidades que permitan la implantación del sistema informático en el menor tiempo y costo posible para la automatización de los procesos contables.
- ✓ Identificar los procesos que se realizan en caja y describirlos de forma detallada.
- ✓ Establecer las funcionalidades del sistema a construir.
- ✓ Establecer las propiedades o cualidades que el futuro software debe poseer.

- ✓ Establecer la realización del sistema a partir de las funcionalidades previstas.
- ✓ Definir la implementación del sistema en términos de componentes y subsistemas de implementación.
- ✓ Definir casos de prueba.

El presente documento consta de cuatro capítulos:

En el capitulo 1, se enmarca el entorno en el que se desarrolla el problema, así como el estado del arte del tema en cuestión incluyendo un análisis de las herramientas, técnicas, metodologías que serán usados para construir la solución al problema planteado.

El capítulo 2, se refiere a las características del sistema, describiendo los procesos de negocio y derivando de los mismos los requerimientos del software. Se modela además el sistema con sus casos de uso bien detallados.

En el capítulo 3 se realiza el análisis y diseño del software, determinando así cómo el sistema será realizado a partir de las funcionalidades previstas, es donde se indica con precisión lo que se va a programar.

En el capítulo 4 se obtiene el modelo de implementación, que incluye diagramas de componentes y de despliegue, definiendo cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes. Se obtiene además el modelo de pruebas, el cual se ocupa de buscar los defectos al software.

## Capítulo 1. Fundamentación Teórica.

### Introducción

En el presente capítulo se realiza un análisis de los temas vinculados con el área en que el futuro software se debe desempeñar, puntualizando conceptos claves antes de comenzar con el desarrollo del mismo. Incluye además un breve resumen de los sistemas automatizados existentes asociados al campo de acción, así como un análisis de las técnicas, tecnologías, metodologías y software a utilizar para realizar el sistema propuesto.

### Contabilidad Financiera

Es el arte de usar ciertos principios al registrar, clasificar y sumarizar en términos monetarios datos financieros y económicos, para informar en forma oportuna y fehaciente de las operaciones de la vida de una empresa. Los registros de cifras pasadas sirven para tomar decisiones que beneficien al presente y al futuro. También proporciona estados financieros que son sujetos al análisis e interpretación, informando a los administradores, a terceras personas y a oficinas gubernamentales del desarrollo de las operaciones de la empresa.(*¿Qué es la contabilidad financiera?*)

### ¿Qué es un ERP?

Los sistemas de planificación de recursos de la empresa (ERP, enterprise resource planning) son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación. Estas partes son de diferente uso, por ejemplo: producción, ventas, compras, logística, contabilidad (de varios tipos), gestión de proyectos, inventarios y control de almacenes, pedidos, nóminas, etc. Solo es posible definir un ERP como la integración de todas estas partes. El ERP integra todo lo necesario para el funcionamiento de los procesos de negocio de la empresa. No es posible hablar de ERP en el momento que tan sólo se integra uno o una pequeña parte de los procesos de negocio. La propia definición indica la necesidad de "Disponibilidad de toda la información para todo el mundo todo el tiempo".(SAAVEDRA)

El propósito fundamental de un ERP es otorgar apoyo a los clientes del negocio, tiempos rápidos de respuesta a sus problemas así como un eficiente manejo de información que permita la toma oportuna de decisiones.

Teniendo en sus características y prestaciones, El MINFAR se ha propuesto realizar un ERP (ERP-FAR), de modo que el sistema que se plantea realizar para gestionar los procesos contables que se realizan en Caja, será solo una parte de un sistema para la gestión de la contabilidad financiera, que se integrará entonces al ERP-FAR contribuyendo así a que se lleve un mejor control de los costos y recursos, es decir un mejor control de toda la institución.

**Herramientas y Tecnologías.**

La información se ha convertido en la herramienta principal para la toma de decisiones en los diferentes niveles organizacionales por ser el motor que confiere dinamismo y apoyo a las estrategias corporativas. Toda organización, sin distinciones, debe poseer sistemas eficientes que se ajusten a las necesidades de los diferentes usuarios, evolucionando permanentemente ante los cambios de su entorno. Las entidades financieras siempre han estado cerca de las tecnologías informáticas y de las comunicaciones, conscientes de las ventajas de incorporarlas con rapidez a sus circuitos y procesos.

A la hora de desarrollar un software informático es necesario tener en cuenta un grupo de tecnologías, herramientas y metodologías disponibles para llevar a cabo dicha empresa. La decisión de la idónea a utilizar depende de las características de la entidad donde se va a implantar el sistema resultante así como de lo que se desea del producto final conjugado con lo que ofrece cada tecnología. Partiendo de esto, seguidamente se realizará un análisis justificando la selección de las herramientas y metodologías a utilizar para elaborar el producto final.

**¿Qué es Internet?**

Es una combinación de hardware (ordenadores interconectados por vía telefónica o digital) y software (protocolos y lenguajes que hacen que todo funcione). Es una infraestructura de redes a escala mundial (grandes redes principales y redes más pequeñas que conectan con ellas) que conecta a la vez a todos los tipos de ordenadores. Fue desarrollado originariamente para los militares de Estados Unidos, y después se utilizó para el gobierno, investigación académica y comercial y para comunicaciones. Es conocida también como la red de redes.

**Composición y Servicios de Internet**

Como se ha mencionado anteriormente, Internet es una red mundial de computadoras interconectadas con un conjunto de protocolos. Internet no es sinónimo de World Wide Web, ésta es parte de Internet, siendo la World Wide Web uno de los muchos servicios ofertados en la red de Internet. La Web es un

sistema de información mucho más reciente (1995) que emplea Internet como medio de transmisión. Otros servicios disponibles en Internet aparte de la Web son el acceso remoto a otras máquinas, transferencia de archivos, correo electrónico, boletines electrónicos, conversaciones en línea, mensajería instantánea, transmisión de archivos, etc.

**La Web**

La palabra "Web" se utiliza para denominar el servicio más importante de la red de Internet. Como se ha dicho, el nombre completo es "World Wide Web" y significa "Telaraña Mundial". Son páginas que utilizan un lenguaje especial llamado HTML que permiten presentar en pantalla textos y gráficos en el formato deseado. Estas páginas contienen referencias o enlaces que permiten acceder a otras páginas. Existen millones de páginas Web con gran cantidad de información sobre todo tipo de temas.

**Las Aplicaciones Web**

Una aplicación Web es un sitio Web donde la navegación a través del sitio, y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio.

Las aplicaciones Web utilizan las tecnologías existentes para generar contenidos dinámicos y permitir a los usuarios del sistema modificar la lógica del negocio en el servidor. Si no existe lógica de negocios en el servidor, el sistema no puede ser considerado una aplicación Web, este es el caso de un sitio Web.

Las aplicaciones Web están comúnmente estructuradas como aplicaciones de tres capas. En su forma más común, el navegador Web es la primera capa, un motor usando alguna tecnología Web dinámica es la capa del medio, y una base de datos como última capa. El navegador Web manda peticiones a la capa media, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario.

**Lenguajes de programación para la Web**

Existe una multitud de lenguajes concebidos para la Web. Cada uno de ellos explota más a fondo ciertas características que lo hacen más o menos útil para desarrollar distintas aplicaciones. Se pueden clasificar de dos formas diferentes: de acuerdo a la arquitectura Cliente/Servidor, los lenguajes del lado del servidor son aquellos reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él (entre estos lenguajes podemos encontrar (PHP, ASP, JSP, Perl, Python); por otro lado, los lenguajes del lado del cliente (entre los cuales no sólo se encuentra el HTML

sino también el VBScript y el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente comprendidos por el navegador y no necesitan un pretratamiento. Cada uno de estos tipos tiene por supuesto sus ventajas y sus inconvenientes, por ejemplo: un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Inversamente, un lenguaje del lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo; por otra parte, los scripts son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente. Este hecho puede resultar a todas luces una forma legítima de proteger el trabajo intelectual realizado.

Se decidió utilizar como lenguajes de programación para la implementación del software el PHP y el Javascript.

**Javascript.**

El Javascript es un lenguaje de programación que se puede utilizar para construir sitios Web y para hacerlos más interactivos. Es un lenguaje interpretado, es decir, que no requiere compilación. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje Javascript puede interactuar con el código HTML, permitiendo a los programadores Web utilizar contenido dinámico. Por ejemplo, hace fácil responder a los acontecimientos iniciados por usuarios (como introducción de datos en formularios) sin tener que utilizar CGI.

**Personal Home Page (PHP).**

PHP es un lenguaje de programación interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. El fácil uso y la similitud con los lenguajes más comunes de programación estructurada, permiten a la mayoría de los programadores experimentados crear aplicaciones complejas con una curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

Su interpretación y ejecución se da en el servidor y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página Web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual

se encarga de regresárselo al cliente. Además es posible utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos, entre otras cosas.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, Microsoft SQL Server, SQLite, entre otros; lo cual permite la creación de Aplicaciones Web muy robustas.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como Linux), Windows y Mac OS X, y puede interactuar con los servidores Web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Entre las ventajas que nos brinda su uso se pueden citar:

- Lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Posee una potente variedad de extensiones para el acceso a la mayoría de los sistemas de gestión de bases de datos, por lo que una migración a otro sistema de gestión es mucho menos costosa que en otras plataformas.
- Permite leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear formularios para la Web.
- Posee una biblioteca nativa de funciones sumamente amplia.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

**Navegador**

Se decidió utilizar como navegador para correr la aplicación el Mozilla Firefox 1.5 o superior, el mismo presenta múltiples ventajas con respecto a otros navegadores existentes, pues ofrece mayor seguridad al usuario, es multiplataforma, cuenta con soporte multilingüe, es extensible y adaptable proveyendo de diversos métodos para cambiar la funcionalidad y la apariencia.

**Metodologías de Trabajo**

Todo desarrollo de software se torna riesgoso y difícil de controlar, pero si no se utiliza una metodología, existe una mayor exposición a obtener clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Muchas veces no se toma en cuenta la utilización de una metodología adecuada. En ocasiones, ocurre que se realiza el diseño del software de manera rígida con los requerimientos que el cliente solicitó, de tal manera que cuando el cliente en la etapa final (etapa de prueba), solicita un cambio se hace muy difícil realizarlo, pues si se hace, altera muchas cosas que no se han previsto, y es justo éste, uno de los factores que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido. Obviamente para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique el desarrollo del mismo, así como escoger una metodología adecuada para el desarrollo del software que sirva como guía para realizar de forma disciplinada y eficiente el producto deseado.

Luego de consultar la bibliografía necesaria, se decidió utilizar la metodología de desarrollo de software RUP acompañado del lenguaje de modelado UML.

**El Proceso Unificado de Modelado. RUP**

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- ✓ Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
- ✓ Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ Transición, El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

El ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

Disciplina de Desarrollo

- ✓ Ingeniería de Negocios: Entendiendo las necesidades del negocio.

- ✓ Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- ✓ Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- ✓ Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- ✓ Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte

- ✓ Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- ✓ Administrando el proyecto: Administrando horarios y recursos.
- ✓ Ambiente: Administrando el ambiente de desarrollo.
- ✓ Distribución: Hacer todo lo necesario para la salida del proyecto

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- ✓ Actividades, Son los procesos que se llegan a determinar en cada iteración.
- ✓ Trabajadores, Vienen hacer las personas o gentes involucradas en cada proceso.
- ✓ Artefactos, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

El ciclo de vida de RUP se caracteriza por:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan, lo cual se capta cuando se modela el negocio y se representa a través de requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla por iteraciones comenzando por los casos de uso más relevantes desde el punto de vista arquitectónico.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente unos más que otros.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

**Lenguaje Unificado de Modelado (UML)**

(UML, por sus siglas en 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 de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante remarcar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas:

- ✓ Diagramas de Casos de Uso para modelar los procesos del negocio.
- ✓ Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- ✓ Diagramas de Colaboración para modelar interacciones entre objetos.
- ✓ Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.

- ✓ Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- ✓ Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- ✓ Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- ✓ Diagramas de Componentes para modelar componentes.
- ✓ Diagramas de Implementación para modelar la distribución del sistema.

**Sistemas gestores de bases de datos.**

Los Sistemas gestores de bases de datos (SGBD o DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de información.

Después de analizar las ventajas, posibilidades y limitantes que ofrecen cada uno de los sistemas gestores de bases de datos candidatos para el manejo de los datos en la realización del sistema propuesto, se concluye que el más apropiado para obtener un producto satisfactorio que se ajuste a las necesidades del cliente es el PostgreSQL. Como lenguaje de acceso a datos se utilizará el SQL.

**SQL**

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...). Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

**PostgreSQL**

PostgreSQL es un motor de base de datos, un servidor de base de datos relacional libre. Está considerado como la base de datos de código abierto más avanzada del mundo. Ofrece muchas ventajas respecto a otros sistemas de bases de datos como son:

- ✓ Instalación ilimitada

Es frecuente que las bases de datos comerciales sean instaladas en más servidores de los que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

✓ Mejor soporte que los proveedores comerciales

Existe una importante comunidad de profesionales y entusiastas de PostgreSQL de los que se puede obtener beneficios y contribuir.

✓ Ahorros considerables en costos de operación

Este software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

✓ Estabilidad y confiabilidad legendarias

En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad.

✓ Extensible

El código fuente está disponible para todos sin costo.

✓ Multiplataforma

PostgreSQL está disponible en casi cualquier sistema operativo.

✓ Diseñado para ambientes de alto volumen

PostgreSQL usa una estrategia de almacenamiento de filas para conseguir una mejor respuesta en ambientes de grandes volúmenes.

✓ Herramientas gráficas de diseño y administración de bases de datos

Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora , Data Architect).

**Herramientas Case**

Las Herramientas case son la mejor base para el proceso de análisis y desarrollo de software, estas herramientas son el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software, es también definido como el conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. La introducción de las herramientas CASE para ayudar en este proceso ha permitido que los diagramas puedan ser fácilmente creados y modificados, mejorando la calidad de los diseños de software.

Como herramienta case en la construcción de la solución propuesta será utilizado el Visual Paradigm.

**Visual Paradigm**

El Visual Paradigm es una herramienta CASE que ofrece un entorno de creación de diagramas para UML, un diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; hace uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de ingeniería directa (versión profesional) e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; disponibilidad de múltiples versiones, para cada necesidad; disponibilidad en múltiples plataformas, y es muy útil para la generación de código fuente en PHP.

**Sistemas Automatizados vinculados al campo de acción**

El desarrollo acelerado del mundo actual hace casi imposible que cualquier entidad o institución pueda controlar de forma manual su actividad contable. Por tal motivo, la mayoría de las empresas tanto en el ámbito nacional como internacional, emplean sistemas automatizados que permiten agilizar el trabajo de modo óptimo, eficiente y con el personal estrictamente necesario. A continuación se ofrecen algunos ejemplos de sistemas automatizados utilizados en diferentes lugares para la gestión contable.

**SABIC (Sistema Automatizado para la Banca Internacional de Comercio)**

El SABIC es un sistema diseñado y desarrollado para satisfacer las necesidades de procesamiento de datos de bancos e instituciones financieras no bancarias, utilizando los medios técnicos de computación disponibles en el mercado. Este sistema ha sido adaptado a los requerimientos de las operaciones propias

del Banco Central de Cuba y ha sido enriquecido para que los empleados que hagan uso de él puedan tramitar sus operaciones y realizar sus consultas sin necesidad de acudir a los archivos, ni a la actividad manual. De esta forma, aumenta la confiabilidad de la información y la seguridad y eficiencia del trabajo.
Este sistema permite contabilizar en tiempo real, lo cual se traduce en que mantiene actualizados los ficheros contables en todo momento. Soporta una contabilidad multimoneda que hace innecesario depender del tipo de cambio para registrar los activos y pasivos, y evita las conversiones de monedas a la hora de contabilizar. Estas características permiten disponer en todo momento de la información, lo más exacta posible, sobre la posición financiera de la institución. (TAMARGO and SANABRIA)

**SIC (Sistema de información contable)**

El sistema de información contable (SIC) utilizado desde comienzos de 1999 en Colombia, es una herramienta que permite manejar en forma integral todo el proceso contable con criterios de oportunidad, consistencia, seguridad y flexibilidad. A través de este sistema se implementó su contabilidad electrónica. Este sistema dispone de una herramienta dinámica y flexible, denominada “Codificador Inteligente” que permite la integración electrónica a la contabilidad de los hechos económicos provenientes de las demás aplicaciones corporativas. Este sistema convierte automáticamente los datos de entrada definidos para cada tipo de transacción en registros contables electrónicos. El SIC contempla las siguientes etapas básicas para la generación de la información contable:

- ✓ Reporte de hechos económicos a la contabilidad (entradas)
- ✓ Procesamiento de datos.
- ✓ Utilización de la información (salidas).
- ✓ Contabilidad matricial o contabilidad multidimensional.
- ✓ El esquema multimonedas

**CREPREC (Sistema automatizado de la gestión municipal en Paraguay)**

CREPREC es un sistema informático integrado para la modernización de la Administración Financiera Municipal, que abarca los Módulos de Catastro Tributario, Recaudación de Tributos, Ejecución Presupuestaria y Contabilidad Financiera y Patrimonial.
El mencionado sistema, responde y satisface a las más exigentes condiciones determinadas por las Normas Legales Municipales y los Organismos de Fiscalización del Sector Público, en cuanto a la

seguridad, claridad y rapidez en el control, procesamiento e información de datos que requiere una Administración Comunal actualizada y transparente.

El modulo de Contabilidad Financiera y Patrimonial Facilita el registro, control e informe de la Contabilidad Financiera y Patrimonial Municipal y que está dispuesta conforme a las disposiciones legales vigentes y aplicables según las normas y procedimientos establecidos para la Contabilidad del Sector Público, en razón de ser la Institución Comunal, un nivel descentralizado del Estado determinado por la Constitución Nacional.

Este módulo provee:

- ✓ Cuadro de Balance de Sumas y Saldos.
- ✓ Balance de Situación analítico y/o sintético.
- ✓ Balance General analítico y/o sintético.
- ✓ Estado del Cuadro de Resultados.
- ✓ Detalle de los Registros en el Diario.
- ✓ Detalle de las Cuentas con sus respectivas mayorizaciones.
- ✓ Cuadro demostrativo de Revalúo y Depreciación de Bienes del Activo Fijo.(*Sistema automatizado de la gestión municipal del área de la administración financiera*)

**Conclusiones:**

En este capítulo se puntualizaron conocimientos importantes, y se tomaron decisiones claves para el comienzo del software, se decidió utilizar como lenguajes de programación PHP y JavaScript, como gestor de base de datos el PostgreSQL y como lenguaje de acceso a datos el SQL, como herramienta case fue elegido el Visual Paradigm y como lenguaje de modelado UML, será usado como navegador el Mozilla Firefox 1.5 o superior y se utilizará la metodología RUP para guiar el desarrollo de la ingeniería de software.

## Capítulo 2. Características del sistema

### Introducción

Con el presente capítulo se pretende comenzar la fase de inicio del desarrollo del software, es decir, se planea hacer un análisis crítico de los procesos que se ejecutan actualmente en caja, de forma tal que se pueda definir un modelo del negocio a partir del cual se deriven los requerimientos necesarios del software a construir así como el modelamiento del sistema con la descripción correcta y detallada de cada caso de uso identificado.

### Objeto de estudio

El MINFAR, es la principal institución cubana encargada de la defensa nacional. Funciona económicamente como una empresa más del país con características específicas que la diferencia del resto de las entidades. Tal como se ha planteado anteriormente, se pretende automatizar la actividad de Caja en las FAR, siendo dicha actividad el objeto de estudio de la presente investigación.

### Problema y Situación problémica.

En las FAR actualmente existe un sistema de gestión contable implementado en el lenguaje de programación Visual FoxPro, dicho sistema no es totalmente factible, pues carece de funcionalidades por lo que no satisface a plenitud la necesidad que existe en la mencionada institución de gestionar y controlar la contabilidad financiera, dicho sistema no tiene implementado módulos medulares para lograr una perfecta gestión de esta contabilidad tales como Caja, es decir que esta operación se lleva a cabo de forma manual recogiéndose los datos y archivándose solamente en papel duro, lo cual constituye un riesgo para el correcto desempeño y control de la gestión contable pues los niveles de errores de cálculo y de elaboración de vales aumentan en forma considerable, además de que las condiciones de trabajo no son las ideales.

### Objeto de automatización.

En consecuencia de lo anteriormente explicado, se define como objeto de automatización los procesos que se realizan en caja. El proceso de Caja garantiza el control de la entrada y salida del dinero en efectivo mediante la entrega y recibo de vales. Las transacciones o servicios que se pueden realizar y que se necesitan automatizar son las siguientes:

Recibo de efectivo: Tiene lugar cuando un cliente viene a hacer un depósito de efectivo por ejemplo: cobro de vivienda o crédito bancario no descontado por nómina, importe recibido de la especialidad por la venta de “formas valiosas”, cobro de responsabilidad material, etcétera.

Anticipo y liquidación de gastos: Tiene lugar cuando a un cliente se le autorizan anticipos por asignación de misiones que conlleven pago de gastos de alojamiento y alimentación en el lugar de destino; así como de los gastos de alimentación por duración de los viajes, otros gastos menores y de transporte incurridos en estos casos, cuando el cliente regresa de dichas misiones se liquidan estos anticipos.

Pagos que se efectúan por caja: Tiene lugar cuando un cliente solicita un pago en efectivo por caja. Estos pagos no pueden exceder el límite autorizado ($500) y se solicitan en casos de compras de materiales y servicios en efectivo, etc.

Nómina para pagos especiales: Tiene lugar cuando se realizan pagos al personal por una sola vez, tales como: subvenciones, gratificaciones, ayudas económicas, diferencias en el pago de Haberes y Salarios, subsidios a trabajadores civiles, dietas al personal en los casos establecidos como gastos directos, así como para aplicar otras retribuciones en determinadas circunstancias y efectuar pagos de salarios no reclamados de meses anteriores.

Todas estas operaciones se registran en distintos modelos para garantizar un correcto control del efectivo en caja teniendo así constancia de cada operación que se realiza en la misma.

Existen además modelos que se utilizan para registrar en forma de resumen las mencionadas operaciones, ellos son:

Registro Control de Anticipos: Se registran cronológicamente y de forma consecutiva la entrega de anticipos así como su liquidación, con el fin de conocer los anticipos entregados que se encuentran pendientes de liquidar, su antigüedad y su monto, además de servir de base para las reclamaciones que correspondan y evitar su liquidación fuera de los términos establecidos.

Movimiento de Efectivo para Pagos Eventuales: Con este modelo se controla cronológicamente el movimiento de efectivo en caja.

**Sistemas automatizados existentes**

Como se ha explicado anteriormente, en el MINFAR existe un sistema de gestión de la contabilidad financiera, el cual no se utiliza en todas las unidades pertenecientes a dicha institución sino en solo una pequeña parte de estas, el mismo no tiene implementado módulos como caja, además de presentar disímiles dificultades:

- ✓ No está concebido para ser multiplataforma.
- ✓ No permite el trabajo de múltiples usuarios.
- ✓ No cuenta con un entorno visual usuario-sistema. Al ser implementado en un software de programación bastante obsoleto, la calidad de la interfaz con el usuario no es la mejor y brinda muy pocas facilidades, lo que hace más engorroso el trabajo.
- ✓ Algunos de los formatos de salida de los reportes no están acorde con los modelos definidos en el Manuel de sistema de contabilidad financiera.

**Información que se maneja.**

En la realización de los procesos de caja, se manejan una serie de documentos legalmente certificados, estos documentos permiten registrar y documentar todas las operaciones que se llevan a cabo en la misma, dichos documentos son:

- ✓ RECIBO DE EFECTIVO SCF-01
- ✓ VALE PARA PAGOS MENORES SCF-02
- ✓ ANTICIPO Y LIQUIDACIÓN DE GASTOS SCF-03
- ✓ MOVIMIENTO DE EFECTIVO PARA PAGOS EVENTUALES SCF-04
- ✓ REGISTRO CONTROL DE ANTICIPOS SCF-05
- ✓ NÓMINA PARA PAGOS ESPECIALES SCF-29

**Modelo de negocio.**

De acuerdo a los objetivos propuestos y a lo que se desea obtener como producto final, teniendo en cuenta lo que indica la metodología de desarrollo utilizada, se ha realizado un estudio minucioso de los procesos de negocio como primer y necesario paso en el desarrollo del software, en consecuencia se ha obtenido el modelo del negocio.

El modelado del negocio es una técnica para comprender los procesos del negocio de la organización. Los propósitos que se persiguen al realizarse el modelado del negocio, son:

- ✓ Entender la estructura y la dinámica de la organización.
- ✓ Comprender los problemas actuales e identificar mejoras potenciales.
- ✓ Asegurarse de que los clientes, usuarios finales y desarrolladores tienen una idea común de la organización.

- ✓ Derivar los requerimientos del sistema a partir del modelo de negocio que se obtenga.

**Actores y trabajadores del negocio**.

Luego de haber realizado un análisis del flujo de los procesos que se realizan en caja, se concluye que los actores y trabajadores del negocio son los que se listan a continuación:

**Actores del Negocio**

| Actores | Descripción |
|---|---|
| Cliente | Persona que solicita los diferentes servicios que se realizan por caja. |

**Trabajadores del negocio**

| Trabajadores | Descripción. |
|---|---|
| Cajero | Persona de la unidad militar que brinda los servicios de la caja. |
| Contador del área de contabilidad | Persona de la unidad militar que recibe los modelos que se envían desde caja para el área de contabilidad. |

**Casos de uso del negocio:**

Recibir Efectivo: Registrar solicitud y operación de depósito de efectivo hecha por el cliente.

Pagar por caja: Registrar operación de pagos en efectivo realizados por caja.

Entregar Anticipo: Registrar solicitud y operación de anticipo hecha por el cliente.

Liquidar Anticipo: Registrar operación de liquidación de anticipos pendientes.

**Modelo de casos de uso del negocio**:

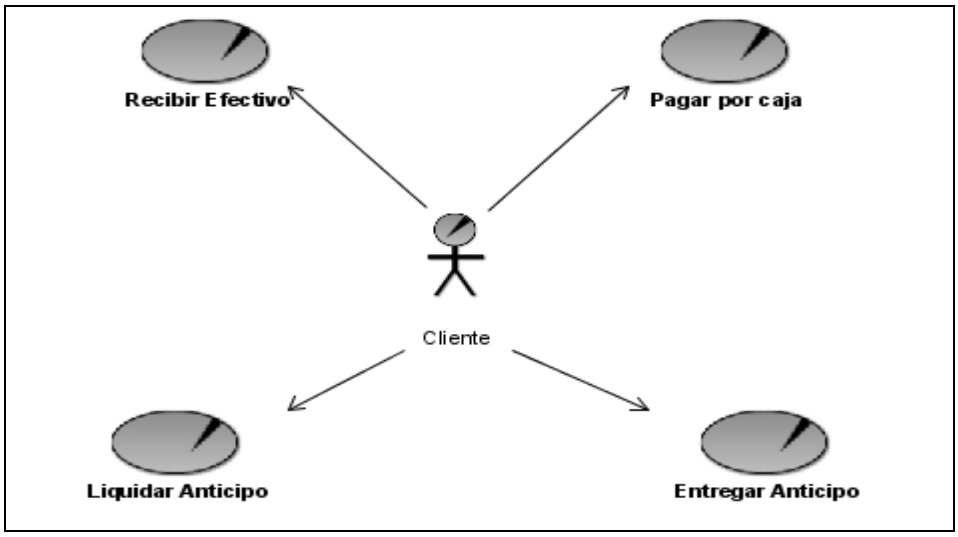



**Fig.1 Modelo de Casos de uso del Negocio.**

**Diagramas de actividades por cada caso de uso del negocio.**

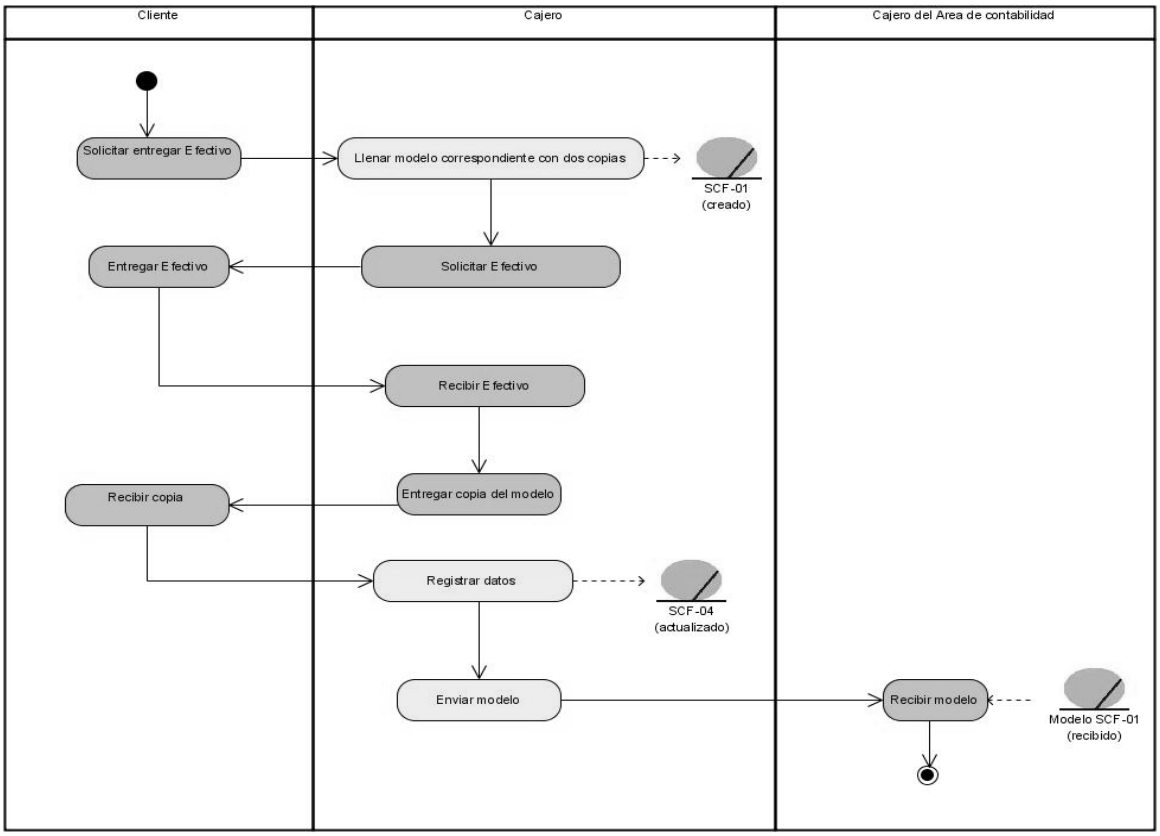



**Fig.2 Diagrama de actividades. Caso de uso Recibir Efectivo.**

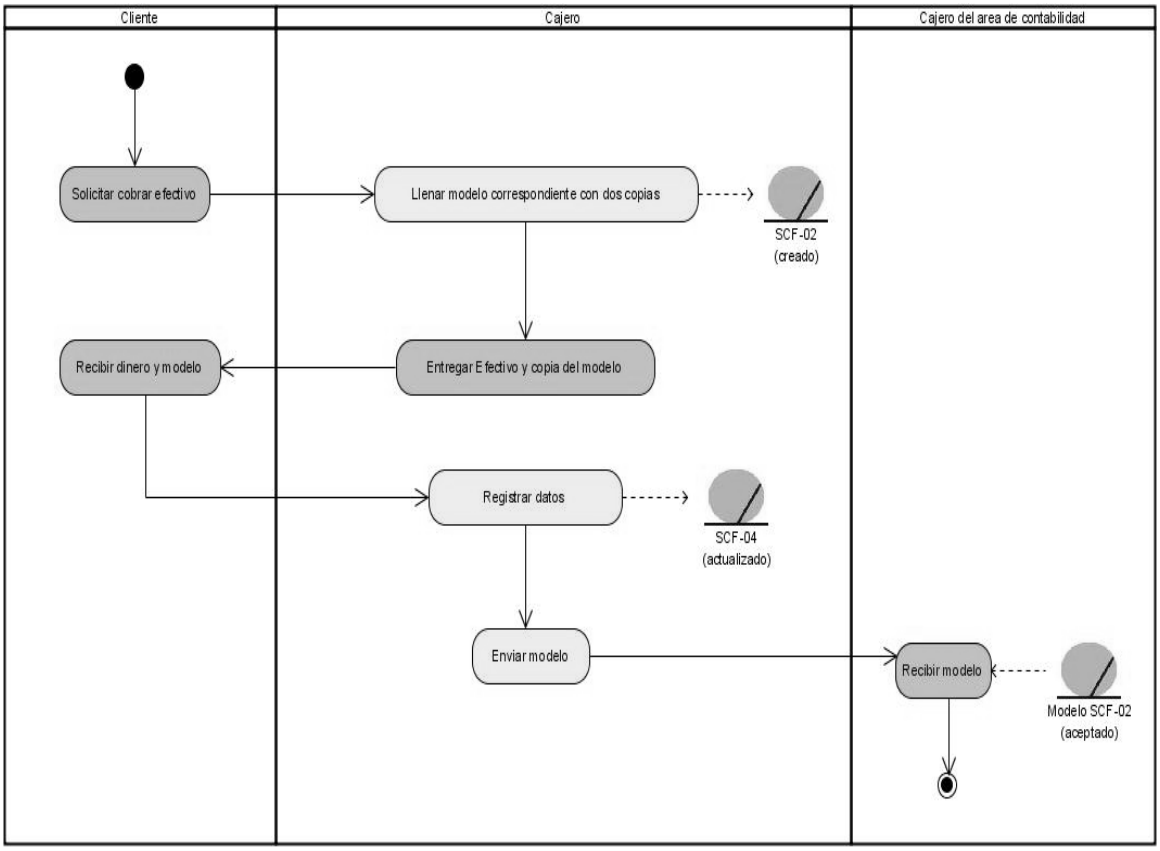



**Fig.3 Diagrama de actividades. Caso de uso Pagar por Caja.**

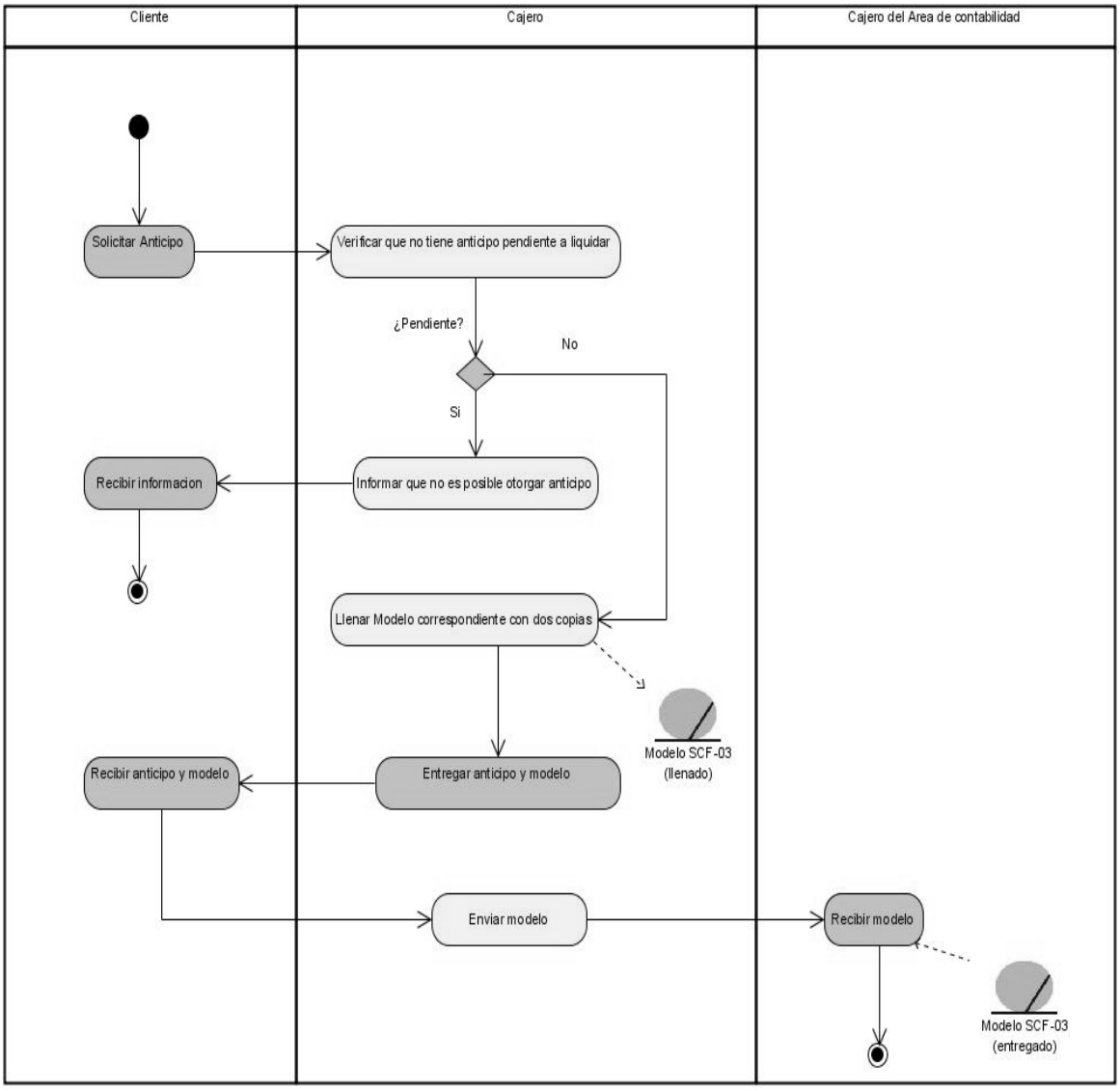



**Fig.4 Diagrama de actividades. Caso de uso Entregar Anticipo.**

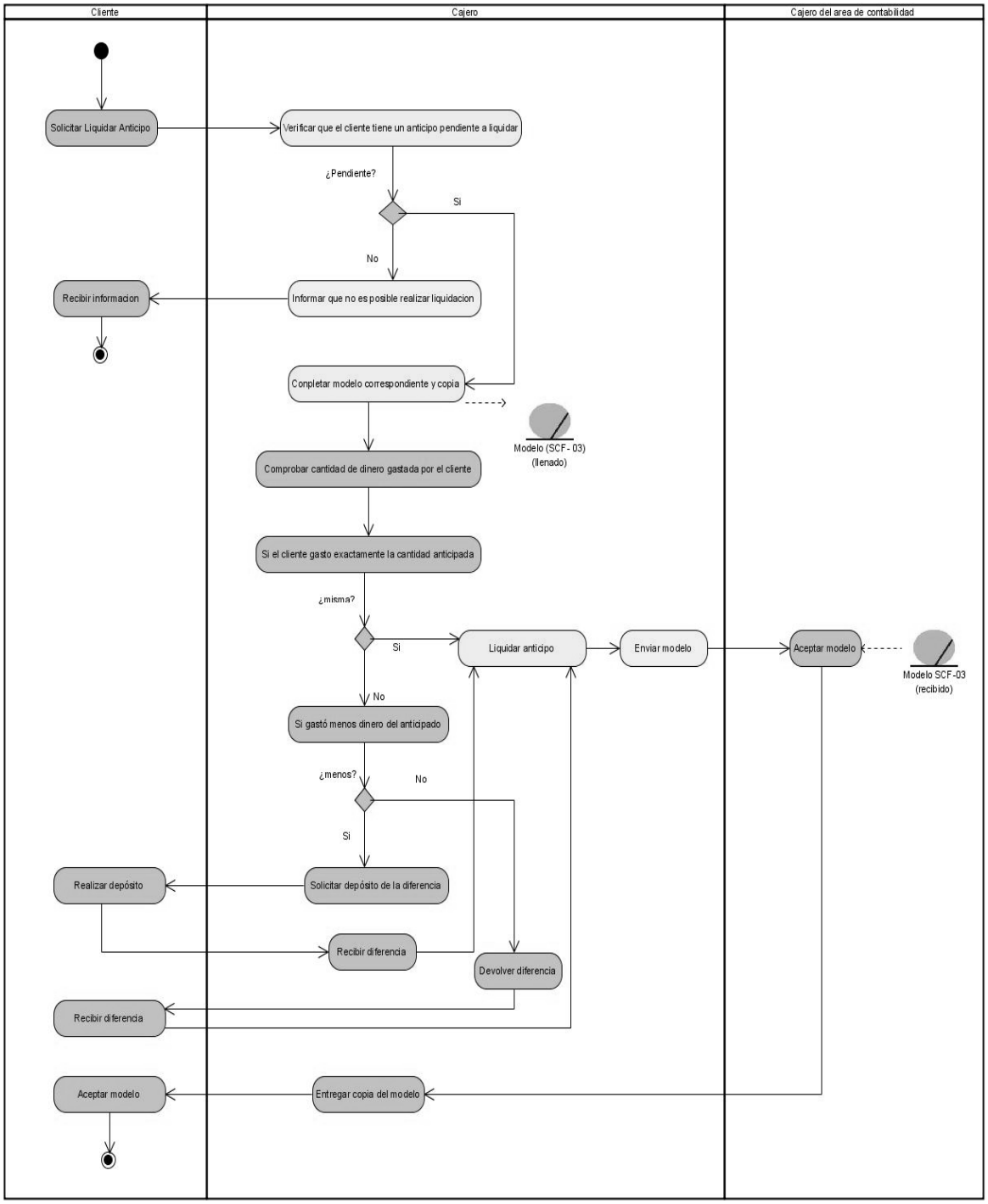



**Fig.5 Diagrama de actividades. Caso de uso Liquidar Anticipo.**

**Descripción de los casos de uso del negocio.**

Para comprender mejor los procesos de negocio, a continuación se hace una descripción textual de la realización de cada caso de uso.

| **Nombre del Caso de Uso** | Recibir Efectivo |
|---|---|
| **Actores** | Cliente |
| **Trabajadores** | Contador |
| **Propósito** | Registrar solicitud y operación de depósito de efectivo hecha por el cliente. |
| **Resumen** | El caso de uso se inicia cuando el cliente solicita hacer un depósito de efectivo. El mismo es atendido por el cajero, el cual llena el modelo pertinente lo registra en el modelo movimiento de efectivo para pagos eventuales, el modelo original es enviado al área de contabilidad, y se le entrega una copia del mismo al cliente. El caso de uso termina cuando ocurre dicha entrega. |
| **Curso Normal de los eventos** | |
| **Acciones del actor** | **Respuesta del proceso del negocio** |
| 1. El cliente solicita hacer un depósito de efectivo. | 2. El cajero llena el modelo correspondiente “Recibo de efectivo” (SCF- 01) con dos copias. |
| 3. El cliente realiza el depósito | 4. El cajero recibe el dinero. |
| | 5. El cajero le entrega al cliente una copia del modelo y guarda una para si. |
| 6. El cliente recibe el modelo. | 7. El cajero registra los datos en el modelo modelo “Movimiento de Efectivo para Pagos Eventuales” (SCF-04). |
| | 8. El cajero envía el modelo “Recibo de efectivo” (SCF- 01) para el área de contabilidad. |
| | 9. Se termina el caso de uso. |

| **Nombre del Caso de Uso** | Entregar anticipo. |
|---|---|
| **Actores** | Cliente |
| **Trabajadores** | Cajero |
| **Propósito** | Registrar solicitud y operación de anticipo hecha por el cliente |
| **Resumen** | El caso de uso se inicia cuando el cliente solicita un anticipo. El cajero llena el modelo pertinente con dos copias y envía una para el área de contabilidad donde será procesada, o imprime dos copias, se queda con una para si y le entrega una al cliente como constancia de que ha recibido un anticipo. El caso de uso termina cuando ocurre dicha entrega. |

| **Curso Normal de los eventos** | |
|---|---|
| **Acciones del actor** | **Respuesta del proceso del negocio** |
| 1. El cliente solicita un anticipo | 2. El cajero verifica que el cliente no tenga un anticipo pendiente a liquidar. |
| | 3. El cajero llena el modelo correspondiente "Anticipo y Liquidación de gastos" (SCF- 03) con dos copias. |
| | 4. El cajero entrega el anticipo y una copia del modelo al cliente. Guarda una para si. |
| 5. El cliente recibe el anticipo y el modelo. | 6. El cajero envía el modelo para el área de contabilidad. |
| | 7. Se termina el caso de uso. |

Flujos alternos.

| **Flujos Alternos** | |
|---|---|
| **Acción del Actor** | **Respuesta del proceso de negocio** |
| 2. Si el cliente tiene un anticipo pendiente a liquidar. | 2.1 El cajero le informa al cliente que no es posible entregar el anticipo. |
| | 2.2 Se termina el caso de uso. |

| **Nombre del Caso de Uso** | Pagar por caja |
|---|---|
| **Actores** | Cliente |
| **Trabajadores** | Cajero |
| **Propósito** | Registrar operación de pagos en efectivo realizados por caja. |
| **Resumen** | El caso de uso se inicia cuando el cliente solicita cobrar un efectivo por caja. El cajero llena el modelo pertinente con dos copias, entrega una copia al cliente y lo envía otra para el área de contabilidad. |
| **Curso Normal de los eventos** | |
| **Acciones del actor** | **Respuesta del proceso del negocio** |
| 1. El cliente solicita cobrar efectivo por caja. | 2. El cajero llena el modelo correspondiente ”Vale para pagos menores” (SCF-02) con dos copias. |
| | 3. El cajero entrega el efectivo y copia del modelo. |
| 4. El cliente recibe el dinero y modelo. | 5. El cajero registra los datos en el modelo “Movimiento de Efectivo para Pagos Eventuales” (SCF-04) |
| | 6. Se envía el modelo (SCF-02) para el área de contabilidad. |
| | Se termina el caso de uso. |

| **Nombre del Caso de Uso** | Liquidar Anticipo |
|---|---|
| **Actores** | Cliente |
| **Trabajadores** | Cajero |
| **Propósito** | Registrar operación de liquidación de anticipos pendientes. |
| **Resumen** | El caso de uso se inicia cuando el cliente solicita la liquidación del anticipo con el modelo pertinente. El cajero verifica el anticipo correspondiente a liquidar, completando los modelos elaborados anteriormente en la entrega del anticipo. El cajero verifica los gastos del cliente sopesando si gastó más de lo anticipado o si por el contrario utilizó menos de esa cantidad; ejecuta las acciones correspondientes en cada caso. Completa el modelo pertinente y envía una copia al área de contabilidad. Entrega una copia al cliente como |

| | |
|---|---|
| | constancia de que se ha liquidado el anticipo y conserva una para si. |
| **Curso Normal de los eventos** | |
| **Acciones del actor** | **Respuesta del proceso del negocio** |
| 1. El cliente solicita liquidar un anticipo con el modelo correspondiente. | 2. El cajero verifica que el cliente tiene un anticipo pendiente a liquidar. |
| | 3. El cajero completa los datos en el modelo correspondiente "Anticipo y Liquidación de gastos" (SCF- 03) y en la copia del mismo. |
| | 4. El cajero comprueba la cantidad de dinero que el cliente gastó. |
| | 5. Si el cliente gastó exactamente la misma cantidad de dinero que recibió en el anticipo. |
| | 6. El cajero realiza la liquidación del anticipo |
| | 7. Envía el modelo para el área de contabilidad. |
| | 8. El cajero entrega una copia del modelo al cliente. |
| 9. El cliente recibe el modelo. | 10. Se termina el caso de uso. |

Flujos alternos.

Acción 2: Si el cliente no tiene pendiente un anticipo a liquidar.

Acción 5 F1: El cajero comprueba que el cliente gastó menos dinero del que se le facilitó en el anticipo.

Acción 5 F2: El cajero comprueba que el cliente gasto más dinero del que se le facilitó en el anticipo.

| **Flujos Alternos** | |
|---|---|
| **Acción del Actor** | **Respuesta del proceso de negocio** |
| 2. Si el cliente no tiene pendiente un anticipo a liquidar. | 2.1 Se le informa que no es posible realizar la liquidación. |
| | 2.2 Se termina el caso de uso. |

| | |
|---|---|
| 5 F1 Si el cliente gastó menos dinero del que se le facilitó en el anticipo. | 5 F1.1 El cajero solicita al cliente el depósito de la diferencia. |
| 5 F1.2 El cliente realiza el depósito. | 5 F1.3 El cajero recibe el dinero. |
| | 5 F1.4 Pasa a la acción 7. |
| 5 F2 Si el cliente gasto más dinero del que se le facilitó en el anticipo | 5 F2.1 El cajero devuelve al cliente la diferencia. |
| 5 F2.2 El cliente recibe el dinero. | 5 F2.3 Pasa a la acción 7. |

**Especificación de los requisitos de software.**

Como próximo paso, corresponde realizar el flujo de trabajo de levantamiento de requisitos, los mismos se dividen en funcionales y no funcionales.

Los requisitos funcionales constituyen las condiciones o capacidades que tiene que cumplir el sistema para satisfacer un contrato o documento formal, o sea, para satisfacer lo que el cliente ha solicitado, son los que describen el qué debe hacer el software, no el cómo. Los requisitos no funcionales son propiedades o cualidades que el producto debe tener porque representan las características del producto, conforman un grupo de categorías que pueden ser de software, hardware, de usabilidad, de seguridad, etc.; son características que debe cumplir la aplicación informática como complemento de los requisitos funcionales que son los que van a describir qué debe hacer el sistema.

**Requerimientos funcionales**

R1 Recibir efectivo.

R1.1 Registrar UM (unidad militar), día, mes y año, nombre de la persona (que recibe el efectivo), importe, ingreso, grupo, especialidad, partida, cuenta(al debe o al haber).

R1.2 Modificar y eliminar los detalles de un recibo de efectivo.

R1.3 Imprimir el modelo SCF-01 Recibo de Efectivo.

R2 Pagar efectivo por caja.

R2.1 Registrar UM (unidad militar), día, mes y año, nombre de la persona, importe, presupuesto, especialidad, partida, cuenta (al debe o al haber).

R2.2 Modificar y eliminar los detalles de un pago por caja.

R2.3 Imprimir el modelo SCF-02 Vale Para Pagos Menores.

R3 Modificar Datos de efectivos y pagos por caja.

R3.1 Buscar datos de recibos de efectivos o de pagos por caja según lo decida el actor ofreciendo varios criterios para facilitar la búsqueda (fecha, nombre de la persona, órgano).

R3.2 Modificar y eliminar o anular los datos de un modelo seleccionado.

R4 Entregar anticipo.

R4.1 Registrar fecha de entrega, fecha de liquidación (estimada), órgano, nombre y apellidos (de la persona que solicita el anticipo), labor a realizar, importe, cuenta (por el debe o por el haber).

R4.2 Verificar que no existan anticipos pendientes a liquidar.

R4.3 Modificar y eliminar o anular datos de anticipos no liquidados.

R5 Liquidar anticipo.

R5.1 Mostrar anticipos pendientes a liquidar.

R5.2 Liquidar anticipo.

R6 Registrar Pagos menores.

R6.1 Recuperar los datos: UM, Autorizado por, No., Año, Día, Mes, Referencia, Detalle, Co, Debe, Haber, Saldo en un registro con el formato correspondiente (SCF-04).

R6.2 Permitir imprimir el registro recuperado.

R7 Controlar Anticipos:

R7.1 Recuperar los datos: UM, No, Fecha, Números, Co, A nombre de, Debe, Haber, Saldo, Vence, Liquidación, No. Liq, en un registro con el formato correspondiente (SCF-05).

R7.2 Permitir imprimir el registro recuperado.

R8 Cuadrar Caja:

R8.1 Permitir hacer el cuadre diario de caja.

**Requerimientos no Funcionales**

Requerimientos de apariencia o interfaz externa: La interfaz debe ser amigable y fácil de usar.

Requerimientos de Usabilidad: La herramienta propuesta será usada por personas que no necesariamente tienen habilidades con el trabajo en la computadora de manera que no puede ser una dificultad para el usuario el uso de ella.

Requerimientos de Rendimiento: La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible.

Requerimientos de Portabilidad: La herramienta propuesta podrá ser usada bajo cualquier sistema operativo, para su implementación se usará PHP y PostgreSQL que son multiplataforma.

Requerimientos de Seguridad:

Confiabilidad: La información manejada por el sistema estará protegida de accesos no autorizados y divulgación pues se trata de las transacciones de efectivo en las unidades militares de las FAR.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos.

Disponibilidad: Solo tendrá acceso pleno al sistema el cajero o persona facultada para realizar las transacciones.

Requerimientos de Ayuda y Documentación en línea: La herramienta contará con sistema de ayuda donde se esclarecerán dudas sobre su uso. En el sistema debe tener una opción para que el usuario encuentre una explicación de cómo navegar por él, así como las facilidades que le brinda.

Requerimientos de Software: En la computadora que hará función de servidor, independientemente del sistema operativo, se necesitará el lenguaje de programación PHP y el SGBD, Postgres. En las computadoras de los usuarios y del grupo de soporte sólo se requerirá del navegador mozilla firefox 1.5 o superior.

**Modelo del sistema.**

Como parte del flujo de trabajo de levantamiento de requisitos, se debe modelar la solución, para lo cual es preciso construir el modelo del sistema, para ello se debe definir, al igual que en el modelo del negocio actores del sistema y casos de uso del sistema para poder construir entonces los diagramas de casos de uso del sistema además de las descripciones textuales de cada uno de estos casos de uso.

**Definición de los actores del sistema.**

| Actores | Justificación |
|---|---|
| Cajero | Persona de la unidad militar que brinda los servicios de la Caja |

**Descripción de los casos de uso del sistema.**

| CU-1 | Recibir Efectivo |
|---|---|
| Actor | Cajero |
| Descripción | Permite al cajero insertar los datos correspondientes a un Recibo de Efectivo, modificar los mismos e imprimir el modelo correspondiente SCF-01 Recibo de Efectivo. |
| Referencia | R2, R2.1, R2.2, R2.3 |

| CU-2 | Pagar por caja |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero insertar los datos de los pagos en efectivo realizados por caja, así como modificarlos e imprimir el modelo correspondiente SCF-02 Vale para pagos Menores. |
| **Referencia** | R2, R2.1, R2.2, R2.3 |

| CU-3 | Modificar E/S Efectivo |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero buscar los datos de recibos de efectivo o pagos por caja según desee el mismo y modificar los datos de un modelo seleccionado. |
| **Referencia** | R3, R3.1, R3.2 |

| CU-4 | Entregar Anticipo |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero insertar los datos correspondientes a un anticipo, así como modificarlos. |
| **Referencia** | R4, R4.1, R4.2, R4.3 |

| CU-5 | Liquidar Anticipo |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero visualizar los datos de todos los anticipos pendientes a liquidar, seleccionarlos y liquidarlos. |
| **Referencia** | R5, R5.1, R5.2 |

| CU-6 | Registrar Pagos menores |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero obtener un registro de las operaciones realizadas de entrega de efectivo por concepto de anticipo para gastos y pagos menores por caja y de recibo de efectivo, además de ofrecer la opción de imprimir dicho registro. |
| **Referencia** | R6, R6.1, R6.2 |

| CU-7 | Controlar anticipos |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero obtener un registro de los datos de las operaciones de anticipo y liquidación de efectivo, además de dar la opción de imprimir dicho registro. |
| **Referencia** | R7, R7.1, R7.2 |

| CU-8 | Cuadrar Caja |
|---|---|
| **Actor** | Cajero |
| **Descripción** | Permite al cajero hacer el cuadre diario de caja |
| **Referencia** | R8, R8.1 |

**Diagrama de casos de Uso.**

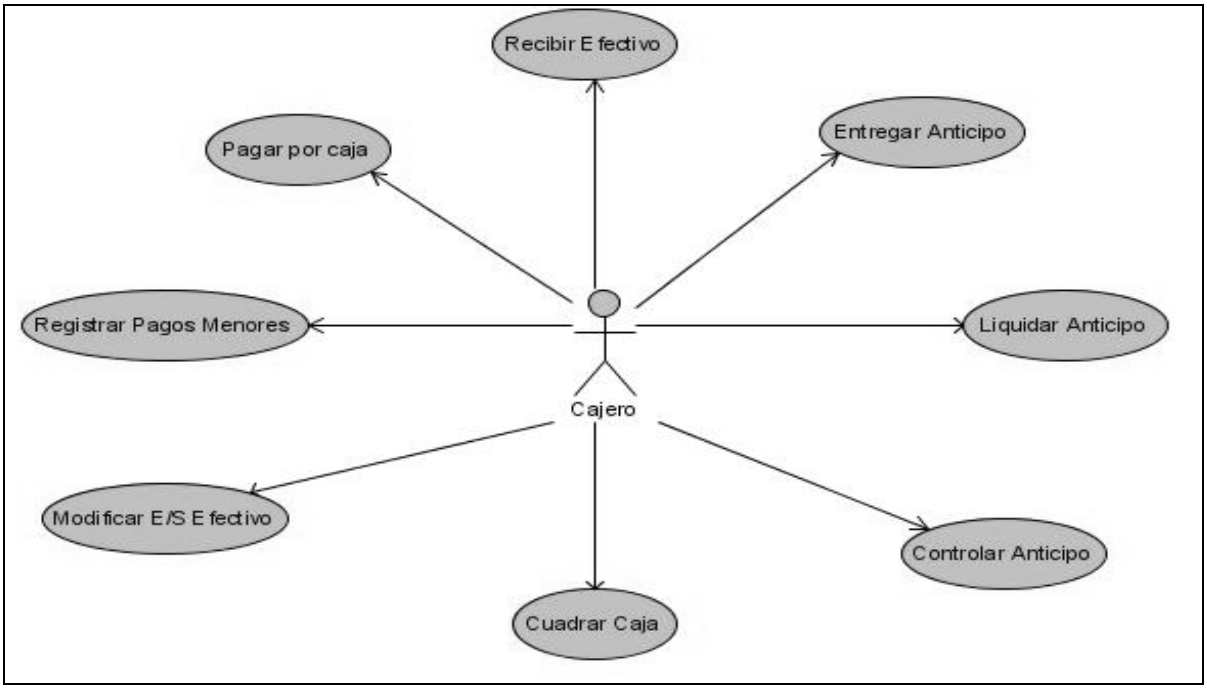
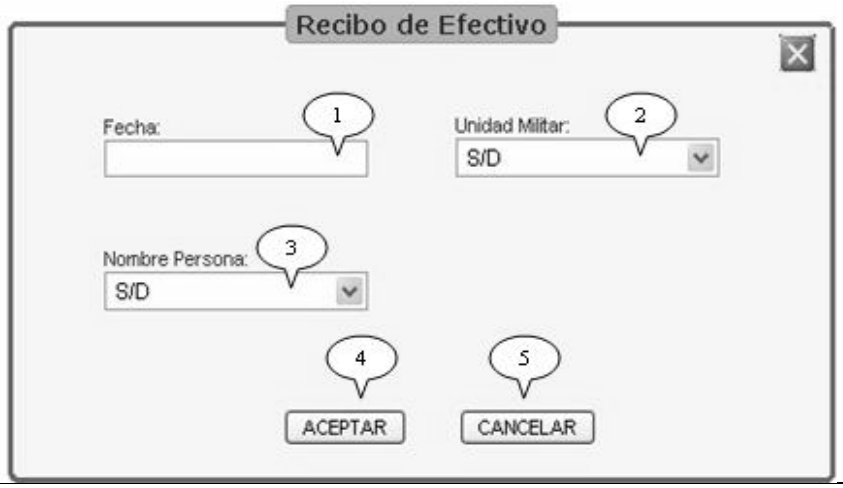
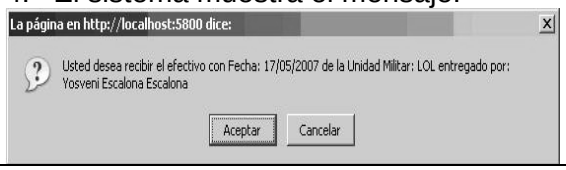
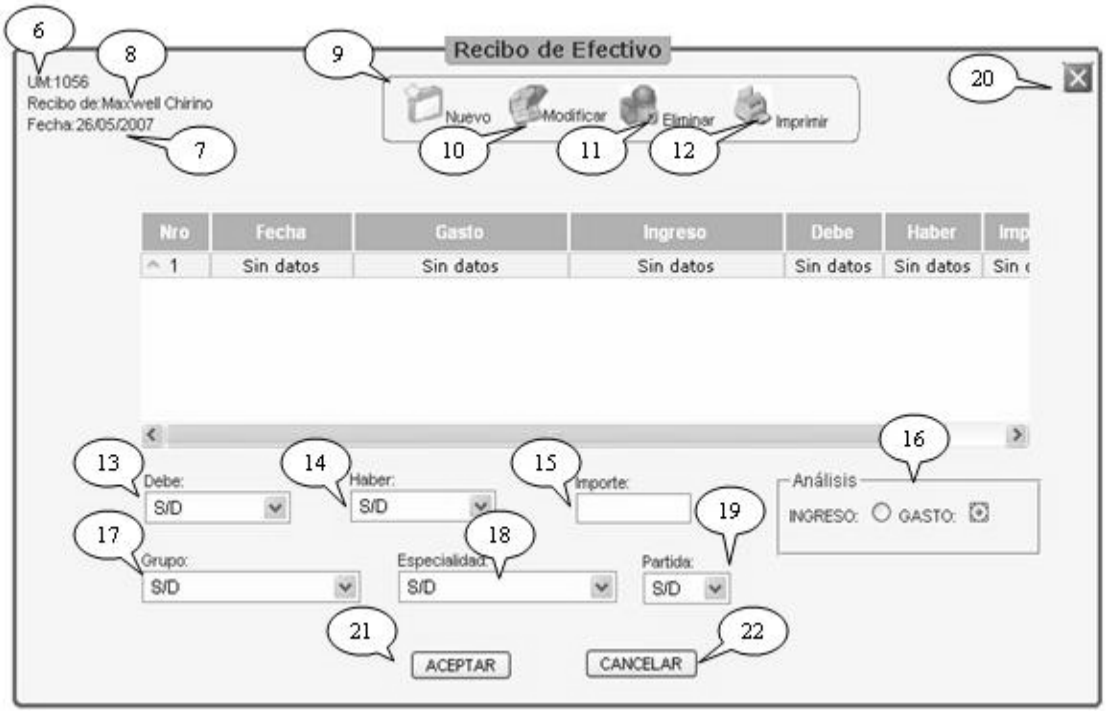



**Fig. 6 Diagrama de Casos de Uso del Sistema.**

**Casos de uso expandidos.**

| **Caso de uso:** | Recibir Efectivo |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Insertar datos de un recibo de efectivo, modificarlos e imprimir el modelo SCF-01 Recibo de Efectivo. |
| **Resumen:** | El cajero inicia este caso de uso cuando decide hacer un recibo de efectivo, tiene las opciones de insertar los datos del modelo correspondiente, modificarlos, eliminarlos e imprimir dicho modelo. Termina el caso de uso con la acción que decida el actor. |
| **Precondiciones:** | |
| **Poscondiciones:** | Se elabora el modelo recibo de efectivo SCF-01. |
| **Referencias:** | R1, R1.1, R1.2, R1.3 |
| **Interfaz I** | |

<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">(1)Fecha en que se recibe el efectivo; Lo selecciona el usuario del control fecha, atributo fecha de la entidad dat_pagosefectivo.<br><br>(2)Unidad Militar a la cual que pertenece el usuario que deposita el efectivo; su valor se toma de una lista que se forma partiendo de la entidad nom_organo, atributo denom.<br><br>(3) Nombre de la persona que entrega el efectivo; su valor se toma de una lista que se forma partiendo de la entidad dat_persona; atributo nombre.</td></tr>
<tr><td colspan="2">Curso normal de eventos para el caso de uso</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El Cajero selecciona del menú principal la opción Recibir Efectivo.</td><td>2. El sistema muestra la interfaz I con las opciones aceptar y cancelar activas.</td></tr>
<tr><td>3. El Cajero introduce los datos solicitados y presiona el botón aceptar (4).</td><td>4. El sistema muestra el mensaje:<br> 
</td></tr>
<tr><td>5. El Cajero acepta el mensaje.</td><td>6. El sistema muestra la interfaz II con las opciones aceptar y cancelar inactivas.</td></tr>
</table>




(6) UM: Número de la unidad militar. El sistema lo muestra de forma automática.

(8) Recibo de: Nombre de la persona que recibe el efectivo. El sistema lo muestra de forma automática.

(7) Fecha: Fecha en que se recibe el efectivo. El sistema lo muestra de forma automática.

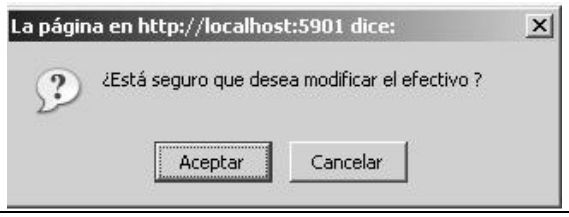
(13) Debe: código de la cuenta por el cual se registra un concepto del efectivo. Atributo idcuenta de la entidad nom_cuenta.

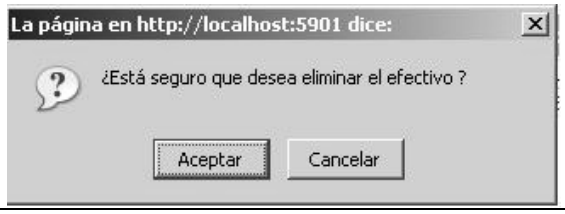
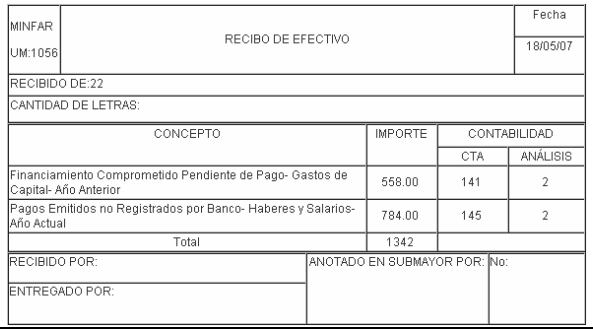
(14) Haber: código de la cuenta por el cual se registra un concepto del efectivo. Atributo idcuenta de la entidad nom_cuenta.

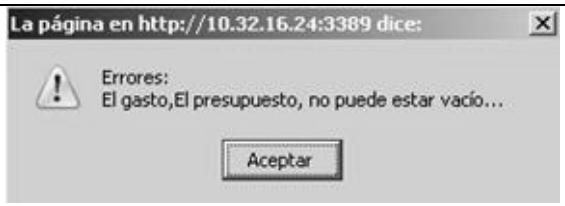
(15) Importe: Importe de un concepto del recibo de efectivo. Atributo importe de la entidad dat_detalle.

(16) Análisis: Si el cajero marca gasto inserta los datos grupo(17), especialidad(18) y partida(19), sino inserta ingreso(20):

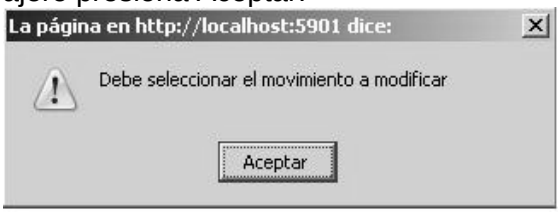
(17) Grupo: su valor se toma de una lista que se forma partiendo de la entidad

<table>
<tr><td colspan="2">nom_presup, atributos idpresup y denom.<br><br>(18) Especialidad: su valor se toma de una lista que se forma partiendo de la entidad nom_entidad, atributos identidad y denom.<br><br>(19) Partida: su valor se toma de una lista que se forma partiendo de la entidad nom_gasto, atributos idgasto y denom.<br><br>(20) Ingreso: su valor se toma de una lista que se forma partiendo de la entidad nom_ingreso, atributos idingreso, denom.</td></tr>
<tr><td>7. El actor decide:<br>• Nuevo (9) (Ver sección nuevo)<br>• Modificar(10) (Ver sección modificar)<br>• Eliminar(11) (Ver sección eliminar)<br>• Imprimir(12) (Ver sección imprimir)</td><td></td></tr>
<tr><td colspan="2">Sección Nuevo</td></tr>
<tr><td>1. El Cajero selecciona la opción Nuevo (9).</td><td>2. El sistema visualiza las opciones aceptar (21) y cancelar (22).</td></tr>
<tr><td>3. El Cajero inserta los datos.</td><td></td></tr>
<tr><td>4. El Cajero presiona Aceptar (21).</td><td>5. El sistema valida los datos.</td></tr>
<tr><td></td><td>6. El sistema muestra la información insertada en el grid (23).</td></tr>
<tr><td colspan="2">Sección Modificar</td></tr>
<tr><td></td><td>1. El sistema le permite al actor desplazarse por el documento y seleccionar el detalle que desea modificar y cambiar los datos que considere necesario.</td></tr>
<tr><td>2. El Cajero selecciona la opción Modificar.</td><td>3. El sistema valida los datos.</td></tr>
<tr><td></td><td>4. El sistema pide confirmación de la acción:<br> 
</td></tr>
<tr><td>5. El Cajero presiona Aceptar.</td><td>6. El sistema muestra actualiza los cambios y muestra los datos en el grid</td></tr>
</table>

<table>
<tr><td></td><td>(23).</td></tr>
<tr><td colspan="2">Sección Eliminar</td></tr>
<tr><td>1. El Cajero selecciona en el grid (23) el detalle a eliminar y presiona la opción Eliminar (11).</td><td>2. El sistema solicita confirmación al actor.<br> 
</td></tr>
<tr><td>3. El Cajero presiona “Aceptar”.</td><td>4. El sistema elimina el detalle seleccionado.</td></tr>
<tr><td colspan="2">Sección Imprimir</td></tr>
<tr><td>1. El Cajero selecciona la opción Imprimir(12)</td><td>1. El sistema muestra una interfaz con el modelo correspondiente y los datos insertados.<br> 
</td></tr>
<tr><td>2. El Cajero presiona Imprimir.</td><td>2. El sistema imprime el modelo.</td></tr>
<tr><td colspan="2">Cursos alternos</td></tr>
<tr><td colspan="2">Acción 3: Si el Cajero decide cancelar (5) se termina el caso de uso.<br>Acción 5: Si el Cajero cancela el mensaje el sistema no ejecuta ninguna acción.<br>Sección Nuevo:<br>Acción 1: Si el Cajero presiona cancelar (20) se termina el caso de uso.<br>Acción 4: Si el Cajero presiona cancelar (20) se termina el caso de uso.<br>Acción 5: Si el Cajero no introduce el importe el sistema muestra el siguiente mensaje. El actor presiona Aceptar.<br> 
<br>Acción 5: Si el Cajero no inserta alguno de los datos el sistema muestra un mensaje informando el dato que necesita insertar. El Cajero presiona Aceptar.</td></tr>
</table>



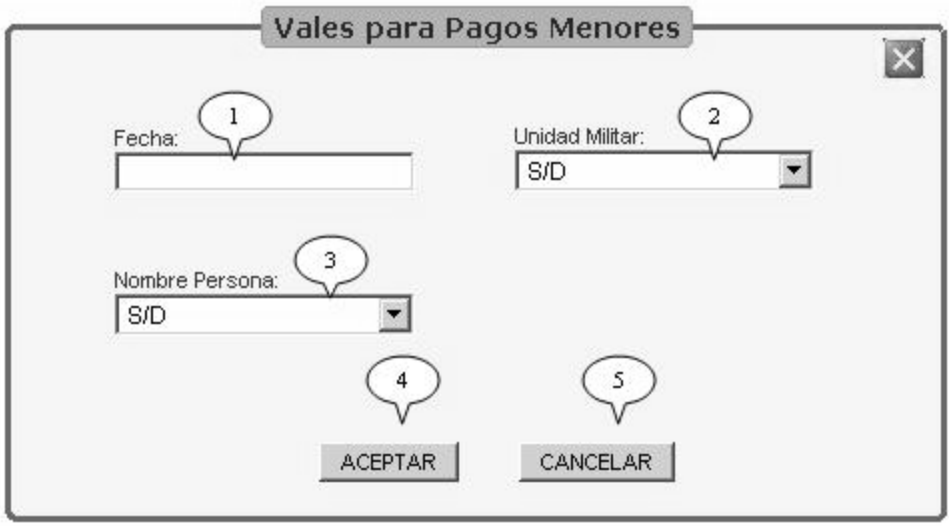

**Sección Modificar:**
**Acción 2: Si** el Cajero decide cancelar (22) ó (20) se termina el caso de uso.
**Acción 3:** Si el Cajero no seleccionó el concepto a modificar el sistema muestra el siguiente mensaje. El Cajero presiona Aceptar.




**Acción 5:** Si el Cajero decide cancelar el sistema no ejecuta ninguna acción.
**Sección Eliminar:**
**Acción 1:** Si el Cajero decide cancelar (22) ó (20) se termina el caso de uso.
**Línea 3:** Si el Cajero decide cancelar eliminar el concepto de efectivo el sistema no ejecuta ninguna acción.

| **Caso de uso:** | Pagar por caja |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Insertar datos de un pago por caja, modificarlos e imprimir el modelo SCF-02 Vale para Pagos Menores. |
| **Resumen:** | El Cajero inicia este caso de uso cuando decide hacer un pago por caja, tiene las opciones de insertar los datos del modelo correspondiente, modificarlos, eliminarlos e imprimir dicho modelo. Termina el caso de uso cuando el Cajero ejecuta las opciones que haya seleccionado. |
| **Precondiciones:** | |
| **Poscondiciones:** | Se elabora el modelo Vale para pagos Menores SCF-02. |
| **Referencias:** | R2, R2.1, R2.2, R2.3 |

**Interfaz I**



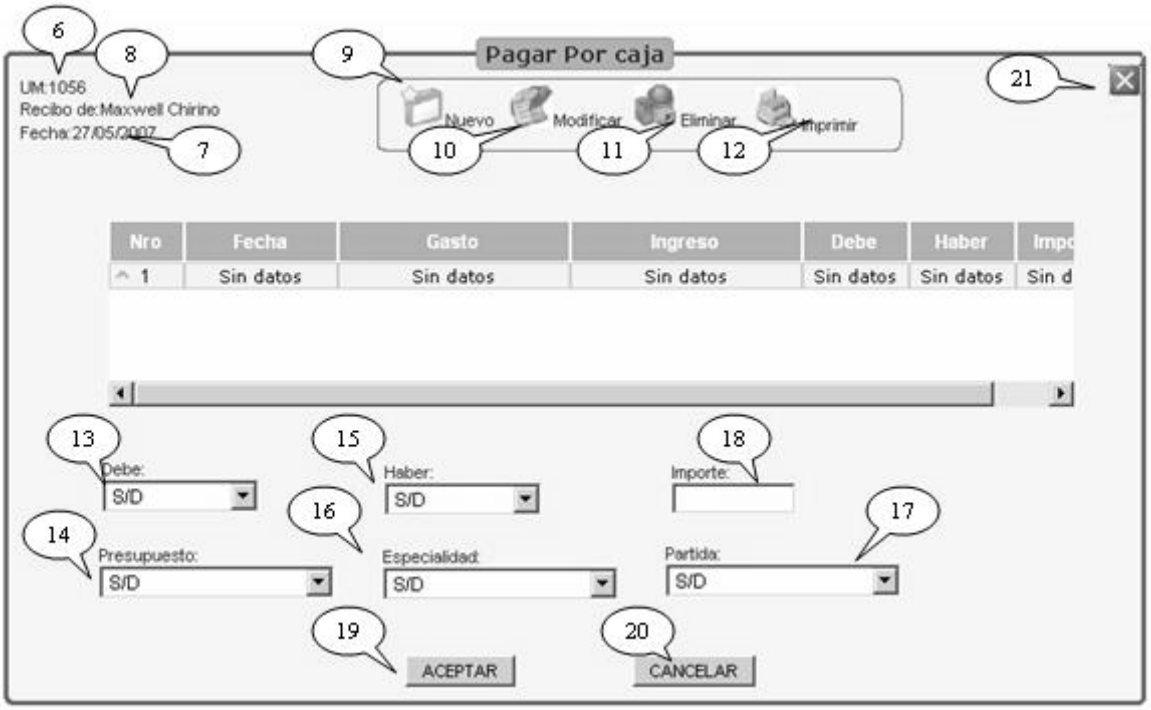

(1)Fecha en que se realiza el pago; Lo selecciona el usuario del control fecha, atributo fecha de la entidad dat_pagosefectivo.

(2)Unidad Militar a la cual que pertenece el usuario que recibe el pago; su valor se toma de una lista que se forma partiendo de la entidad nom_organo, atributo denom.

(3) Nombre de la persona que recibe el pago; su valor se toma de una lista que se forma partiendo de la entidad dat_persona; atributo nombre.

**Curso normal de eventos para el caso de uso**

| **Acción del actor** | **Respuesta del sistema** |
|---|---|
| 1. El Cajero selecciona del menú principal la opción Pagos Menores. | 2. El sistema muestra la interfaz I con las opciones aceptar (4) y cancelar (5) activas. |
| 3. El Cajero introduce los datos solicitados y presiona el botón aceptar (4). | El sistema muestra el mensaje:  |
| 4. El Cajero acepta el mensaje. | 5. El sistema muestra la interfaz II con las opciones aceptar y cancelar inactivas. |




(6) UM: Número de la unidad militar. El sistema lo muestra de forma automática.

(7) Recibo de: Nombre de la persona que recibe el pago. El sistema lo muestra de forma automática.

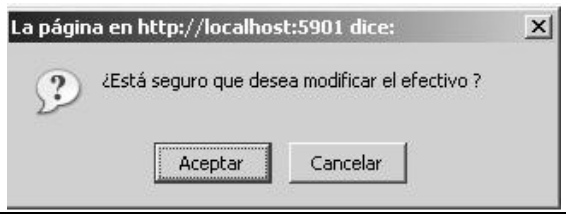
(8) Fecha: Fecha en que se recibe el pago. El sistema la muestra de forma automática.

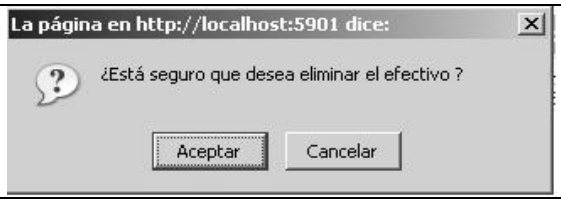
(13) Debe: código de la cuenta por el cual se registra un concepto del pago. Atributo idcuenta de la entidad nom_cuenta.

(14) Presupuesto: su valor se toma de una lista que se forma partiendo de la entidad nom_presup, atributos idpresup y denom.

(15) Haber: código de la cuenta por el cual se registra un concepto del pago. Atributo idcuenta de la entidad nom_cuenta.

(16) Especialidad: su valor se toma de una lista que se forma partiendo de la entidad nom_entidad, atributos identidad y denom.

<table>
<tr><td colspan="2">(17) Partida: su valor se toma de una lista que se forma partiendo de la entidad nom_gasto, atributos id_gasto, denom.<br><br>(18) Importe: Importe de un concepto del pago por caja. Atributo importe de la entidad dat_detalle.<br><br>(20) Botón cerrar que le permite al Cajero salir del caso de uso cuando lo desee.</td></tr>
<tr><td>6. El Cajero decide:<br>• Nuevo (9) (Ver sección nuevo)<br>• Modificar(10) (Ver sección modificar)<br>• Eliminar(11) (Ver sección eliminar)<br>• Imprimir(12) (Ver sección imprimir)</td><td></td></tr>
<tr><td colspan="2">Sección Nuevo</td></tr>
<tr><td>1. El Cajero selecciona la opción Nuevo (9).</td><td>2. El sistema visualiza las opciones aceptar y cancelar.</td></tr>
<tr><td>3. El Cajero inserta los datos.</td><td></td></tr>
<tr><td>4. El Cajero presiona Aceptar.</td><td>5. El sistema valida los datos.</td></tr>
<tr><td></td><td>6. El sistema muestra la información insertada en el grid.</td></tr>
<tr><td colspan="2">Sección Modificar</td></tr>
<tr><td></td><td>1. El sistema le permite al Cajero desplazarse por el documento seleccionar el detalle que le interesa modificar y cambiar los datos que considere necesario.</td></tr>
<tr><td>2. El Cajero selecciona la opción Modificar.</td><td>3. El sistema valida los datos.</td></tr>
<tr><td></td><td>4. El sistema pide confirmación de la acción:<br> 
</td></tr>
<tr><td>5. El Cajero presiona Aceptar.</td><td>6. El sistema actualiza los cambios y muestra los datos modificados en el grid.</td></tr>
<tr><td colspan="2">Sección Eliminar</td></tr>
<tr><td>1. El Cajero selecciona el detalle y presiona la opción Eliminar (11).</td><td>2. El sistema solicita confirmación al actor.</td></tr>
</table>

| | |
|---|---|
| |  |
| 3. El actor decide Aceptar. | 4. El sistema elimina el detalle seleccionado. |
| **Sección Imprimir** | |
| 1. El Cajero selecciona la opción Imprimir(12) | 3. El sistema muestra una interfaz con el modelo correspondiente y los datos insertados.  |
| 2. El Cajero presiona Imprimir. | 4. El sistema imprime el modelo. |

MINFAR UM:1056 — RECIBO DE EFECTIVO — Fecha 18/05/07

RECIBIDO DE:22

CANTIDAD DE LETRAS:

| CONCEPTO | IMPORTE | CONTABILIDAD | |
|---|---|---|---|
| | | CTA | ANÁLISIS |
| Financiamiento Comprometido Pendiente de Pago- Gastos de Capital- Año Anterior | 558.00 | 141 | 2 |
| Pagos Emitidos no Registrados por Banco- Haberes y Salarios- Año Actual | 784.00 | 145 | 2 |
| Total | 1342 | | |

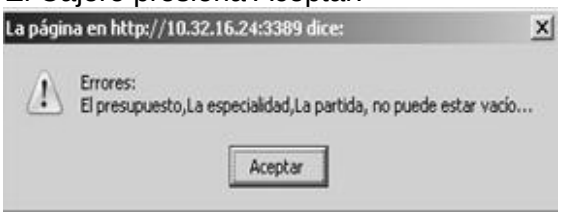
RECIBIDO POR: | ANOTADO EN SUBMAYOR POR: | No:

ENTREGADO POR:

**Cursos alternos**

**Acción 3:** Si el Cajero decide cancelar se termina el caso de uso.
**Acción 5:** Si el Cajero cancela el mensaje el sistema no ejecuta ninguna acción.
**Sección Nuevo:**
**Acción 4:** Si el Cajero presiona cancelar se termina el caso de uso.
**Acción 5:** Si el Cajero no introduce el importe el sistema muestra el siguiente mensaje. El Cajero presiona Aceptar.


**Acción 5:** Si el Cajero no inserta algún dato el sistema muestra un mensaje notificando el dato que debe insertar. El Cajero presiona Aceptar.

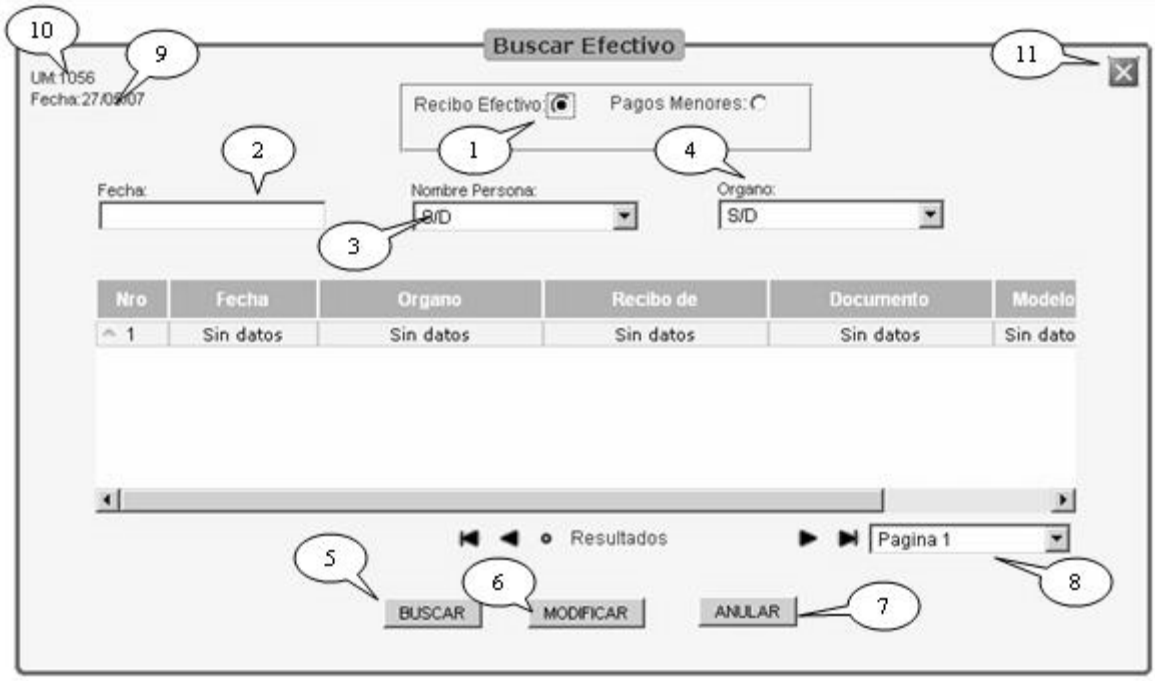



**Sección Modificar:**
**Acción 3:** Si el Cajero no seleccionó el detalle a modificar el sistema muestra el siguiente mensaje. El Cajero presiona Aceptar.


**Acción 5:** Si el Cajero decide cancelar, el sistema no ejecuta ninguna acción.

**Sección Eliminar:**
**Línea 3:** Si el Cajero decide cancelar el sistema no ejecuta ninguna acción.

| **Caso de uso:** | Modificar E/S Efectivo |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Modificar datos de un modelo Recibo de Efectivo (SCF-01) o un Vale para Pagos Menores (SCF-02). |
| **Resumen:** | El Cajero inicia este caso de uso cuando decide modificar los datos de un Recibo de Efectivo (SCF-01) o de un Vale para Pagos Menores (SCF-02) para ello tiene la posibilidad de hacer una búsqueda del modelo a modificar por varios criterios. Termina el caso de uso cuando el cajero concluye las acciones que haya seleccionado. |
| **Precondiciones:** | Debe existir el modelo previo a modificar. |
| **Poscondiciones:** | Se modifica el modelo Recibo de Efectivo (SCF-01) o el modelo Vale para pagos Menores (SCF-02) |
| **Referencias:** | R3, R3.1, R3.2 |
| **Interfaz I** | |
| | |




(1)Radiobutton para indicar si se va a modificar un Recibo de Efectivo (SCF-01) o un Vale para Pagos Menores (SCF-02).

(2)Fecha en que se recibió el pago o entregó el efectivo; Lo selecciona el usuario del control fecha, atributo fecha de la entidad dat_pagosefectivo.

(3) Nombre de la persona que recibió el pago por caja o depositó el efectivo; su valor se toma de una lista que se forma partiendo de la entidad dat_persona relacionada con la entidad dat_pagosefectivo; atributo nombre.

(4)Unidad Militar a la que pertenece el usuario que recibió el pago por caja o depositó el efectivo; su valor se toma de una lista que se forma partiendo de la entidad nom_organo relacionada con la entidad dat_pagosefectivo, atributo denom.

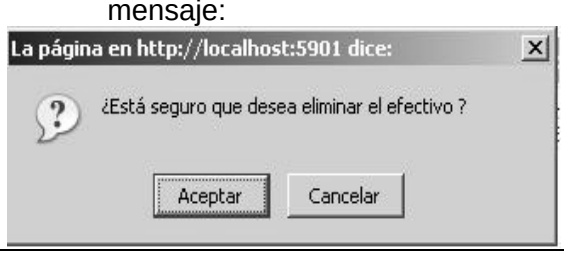
(8) Control paginado que le permite al cajero navegar por el grid.

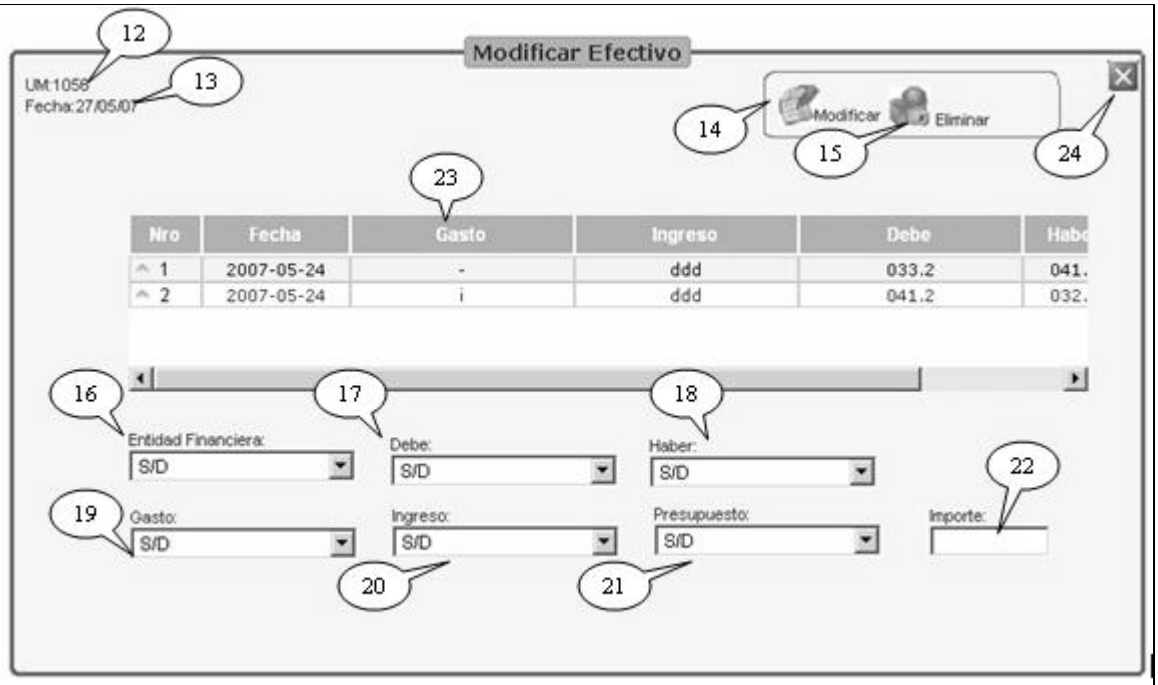
(9) Fecha en la que se realiza la acción, el sistema la muestra de forma automática.

(10) Unidad Militar que realiza la operación, el sistema la muestra de forma automática.

(11) Botón que le permite al Cajero salir del caso de uso cuando lo desee.

| **Curso normal de eventos para el caso de uso** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El Cajero selecciona del menú principal la opción Modificar Efectivo. | 2. El sistema muestra la interfaz I con Recibo de Efectivo marcado por defecto. |
| 3. El Cajero deja la opción Recibo Efectivo marcada por defecto si desea modificar los datos de un recibo de efectivo, sino selecciona Pagos Menores. Selecciona los criterios por los cuales desea buscar. | |
| 4. El Cajero presiona Buscar(5) | 5. El sistema muestra en el grid (8) los resultados de la búsqueda. |
| 6. El Cajero selecciona en el grid (8) el modelo al que desea hacer los cambios. | |
| 7. El Cajero decide:<br>• Modificar(6)(ver sección Modificar)<br>• Eliminar(7) (ver sección Eliminar) | |
| **Sección Eliminar** | |
| 1. El Cajero selecciona la opción Eliminar (7). | 2. El sistema muestra el siguiente mensaje:<br>  |
| 3. El Cajero presiona Aceptar. | 4. Se elimina el modelo. |
| **Sección Modificar** | |
| 1. El Cajero selecciona la opción modificar(6) | 2. El sistema muestra la interfaz II con un grid que contiene los datos de los detalles correspondientes al modelo que el usuario seleccionó, además de los controles necesarios para modificar los datos de dichos detalles. |
| | |




(12) Unidad Militar que realiza la operación, el sistema la muestra de forma automática

(13) Fecha en la que se realiza la acción, el sistema la muestra de forma automática.

(16) Entidad: su valor se toma de una lista que se forma partiendo de la entidad nom_entidad, atributos identidad y denom.

(17) Debe: código de la cuenta por el cual se registra un concepto del efectivo. Atributo idcuenta de la entidad nom_cuenta.

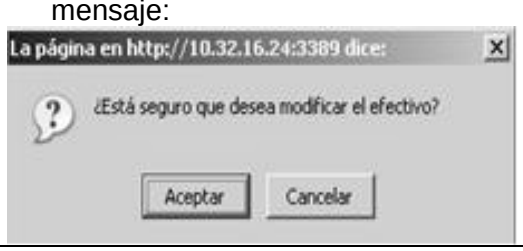
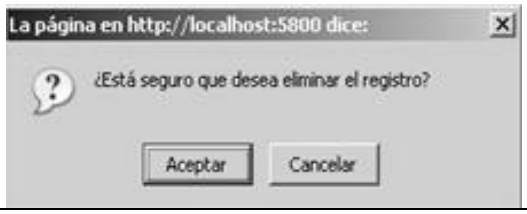
(18) Haber: código de la cuenta por el cual se registra un concepto del efectivo. Atributo idcuenta de la entidad nom_cuenta.

(19) Gasto: su valor se toma de una lista que se forma partiendo de la entidad nom_gasto, atributos id_gasto, denom.

(20) Ingreso: su valor se toma de una lista que se forma partiendo de la entidad nom_ingreso, atributos idingreso, denom.

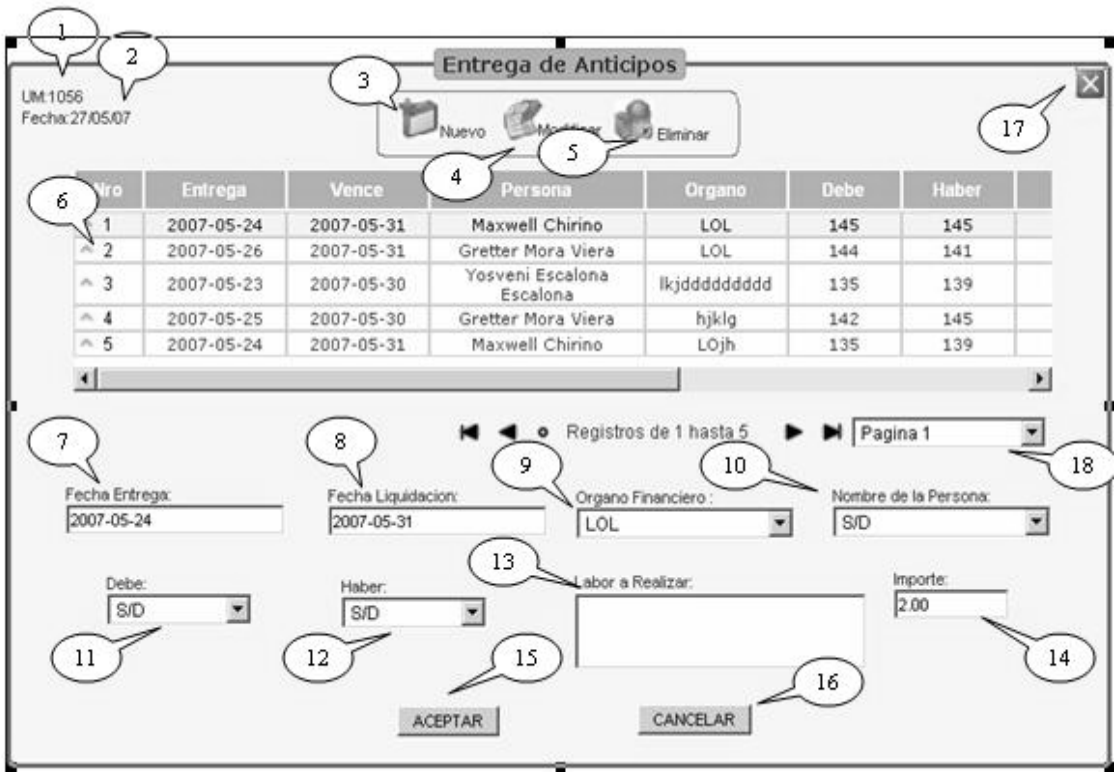
(21) Presupuesto: su valor se toma de una lista que se forma partiendo de la entidad nom_presup, atributos idpresup y denom.

(22) Importe: Importe de un concepto del pago por caja. Atributo importe de la entidad

<table>
<tr><td colspan="2">dat_detalle.<br><br>(23) Componente Grid para mostrar los datos, el cual se llena de acuerdo al efectivo que el cajero selecciono para modificar o eliminar.<br><br>(24) Botón que le permite al Cajero salir del caso de uso cuando este lo desee.</td></tr>
<tr><td>3. El Cajero selecciona en el grid (20) el detalle a afectar.</td><td>4. El sistema muestra los datos del detalle seleccionado en los controles (16), (17), (18), (19), (20), (21) y (22).</td></tr>
<tr><td>5. El Cajero decide:<br>• Modificar (10) (Ver sección Modificar II)<br>• Eliminar (11) (Ver sección Eliminar II)</td><td></td></tr>
<tr><td colspan="2">Sección Modificar II</td></tr>
<tr><td>1. El Cajero modifica los datos que desee.</td><td></td></tr>
<tr><td>2. El Cajero selecciona la opción Modificar(14)</td><td>3. El sistema valida los datos.</td></tr>
<tr><td></td><td>4. El sistema muestra el siguiente mensaje:<br> 
</td></tr>
<tr><td>5. El Cajero presiona Aceptar.</td><td>6. Se modifica el detalle seleccionado mostrándose sus datos en el grid (23).</td></tr>
<tr><td colspan="2">Sección Eliminar II</td></tr>
<tr><td>1. El Cajero selecciona Eliminar(15)</td><td>2. El sistema pide confirmación de la acción.<br> 
</td></tr>
</table>

| | |
|---|---|
| 3. El Cajero presiona Aceptar. | 4. El sistema elimina el modelo seleccionado. |
| **Cursos alternos** | |
| **Acción 4:** Si el Cajero presiona cancelar se termina el caso de uso.<br>**Acción 7:** Si el Cajero presiona cancelar se termina el caso de uso.<br>**Sección Eliminar:**<br>**Acción 3:** Si el Cajero presiona cancelar el sistema no ejecuta ninguna acción.<br>**Sección Modificar II:**<br>**Acción 3**: Si el Cajero no inserta el importe el sistema muestra el siguiente mensaje. El Cajero presiona Aceptar.<br>  <br><br>**Acción 5:** Si el Cajero presiona Cancelar el sistema no ejecuta ninguna acción.<br>**Sección Eliminar II:**<br>**Acción 3:** Si el Cajero presiona Cancelar el sistema no ejecuta ninguna acción. | |

| **Caso de uso:** | Entregar Anticipo |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Insertar los datos de Anticipos o modificar datos de otros pendientes a liquidar. |
| **Resumen:** | El Cajero inicia este caso de uso decide Insertar los datos de un anticipo (Modelo Anticipo y Liquidación de Gastos SCF-03), o modificar los datos de otros anticipos pendientes a liquidar. Termina el caso de uso cuando el Cajero culmina con la opción seleccionada. |
| **Precondiciones:** | La persona que recibe el anticipo no puede tener uno pendiente a liquidar. |
| **Poscondiciones:** | Se insertan o modifican los datos de un anticipo. |
| **Referencias:** | R4, R4.1, R4.2, R4.3 |
| **Interfaz I** | |
| | |




(1) Unidad Militar a la que pertenece el usuario que va a recibir el anticipo; atributo denom de la entidad nom_organo.

(2) Fecha en que se realiza la acción, el sistema la muestra de forma automática.

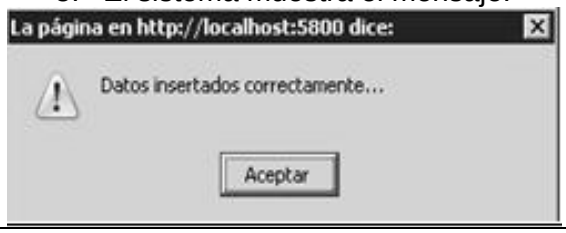
(7) Fecha Entrega: Fecha en que se entrega el anticipo, atributo fanticipo de la tabla dat_anticipo.

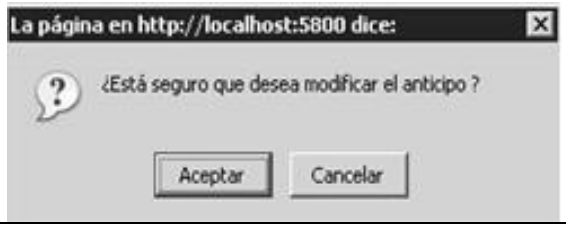
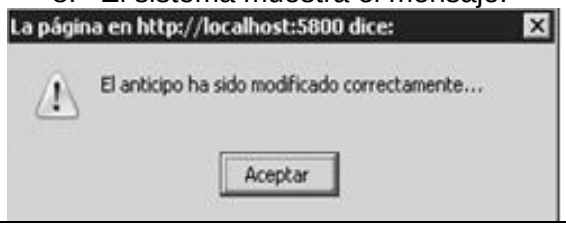
(8) Fecha Liquidación: Fecha estimada de liquidación del anticipo, atributo fliquidacion de la tabla dat_anticipo.

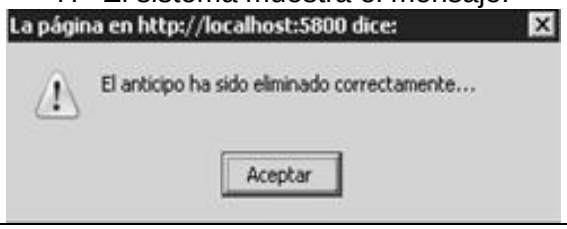
(9) Órgano Financiero: Unidad Militar a la cual que pertenece el usuario que recibe el anticipo; su valor se toma de una lista que se forma partiendo de la entidad dat_anticipo relacionada con la entidad nom_organo, atributo denom de esta ultima.

10) Nombre de la persona que recibe el anticipo; su valor se toma de una lista que se forma partiendo de la entidad dat_anticipo relacionada con la entidad dat_persona; atributo nombre de esta ultima.

(11) Debe: código de la cuenta por el cual se registra el anticipo. Atributo idctadebe de la

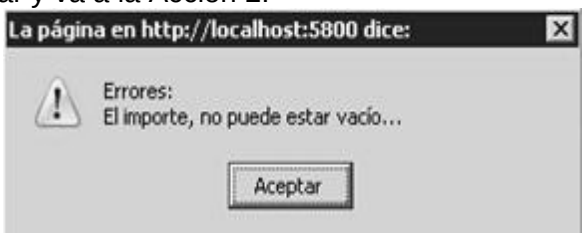
<table>
<tr><td colspan="2">entidad dat_anticipo.<br><br>(12) Haber: código de la cuenta por el cual se registra el anticipo. Atributo idctahaber de la entidad dat_anticipo.<br><br>(13) Labor a realizar: Descripción del motivo por el cual el usuario recibe el anticipo. Atributo descripact de la entidad dat_anticipo.<br><br>(14) Importe: Importe que se entrega en el anticipo. Atributo importeanti de la entidad dat_anticipo.<br><br>(17) Botón que le permite al Cajero salir del caso de uso cuando lo desee.<br><br>(18) Paginado que le permite al Cajero navegar por los resultados en el Grid.</td></tr>
<tr><td colspan="2">Curso normal de eventos para el caso de uso</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El Cajero selecciona del menú principal la opción Entregar Anticipos.</td><td>2. El sistema muestra la interfaz I con los anticipos que se encuentran pendientes de liquidar en el grid (6), las opciones aceptar (15) y cancelar (16) se encuentran activas.</td></tr>
<tr><td>3. El Cajero dedice:<br>• Nuevo(3)(Ver sección Nuevo)<br>• Modificar(4) (Ver sección Modificar)<br>• Eliminar(5) (Ver sección Eliminar)</td><td></td></tr>
<tr><td colspan="2">Sección Nuevo</td></tr>
<tr><td>1. El Cajero selecciona la opción Nuevo(3)</td><td></td></tr>
<tr><td>2. El Cajero introduce los datos del nuevo anticipo.</td><td></td></tr>
<tr><td>3. El Cajero presiona Aceptar (15).</td><td>4. El sistema valida los datos.</td></tr>
<tr><td></td><td>5. El sistema muestra el mensaje:<br> 
</td></tr>
<tr><td>6. El Cajero presiona aceptar.</td><td>7. El sistema inserta los datos del anticipo y los muestra en el grid</td></tr>
</table>

| | |
|---|---|
| | (6). |
| **Sección Modificar** | |
| 1. El Cajero selecciona en el grid (6) el anticipo a modificar. | 2. El sistema muestra en los controles (7), (8), (9), (10), (11), (12), (13) y (14) los datos del anticipo seleccionado. |
| 3. El Cajero realiza los cambios que desee en los datos de anticipo. | |
| 4. El Cajero selecciona la opción Modificar (4). | 5. El sistema valida los datos. |
| | 6. El sistema pide confirmación de la acción:<br>  |
| 7. El Cajero presiona Aceptar. | 8. El sistema muestra el mensaje:<br>  |
| 9. El Cajero presiona Aceptar. | 10. El sistema actualiza los datos del anticipo modificado y los muestra en el grid (6). |
| **Sección Eliminar** | |
| 1. El Cajero selecciona en el grid (6) el anticipo a eliminar. | 2. El sistema muestra los datos del anticipo seleccionado en el grid (6). |
| 3. El Cajero selecciona la opción Eliminar (11). | 4. El sistema solicita confirmación de la acción.<br>  |
| 5. El Cajero presiona Aceptar. | 6. El sistema verifica si el anticipo es el último insertado. |

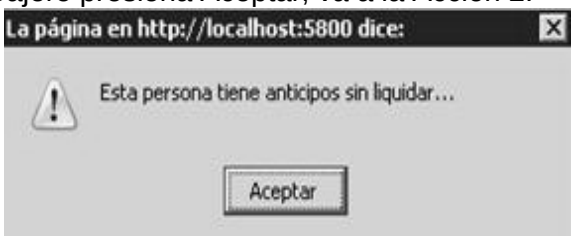
| | |
|---|---|
| | 7. El sistema muestra el mensaje:<br>  |
| 8. El Cajero presiona Aceptar. | 9. El sistema elimina o anula el anticipo. |

**Cursos alternos**

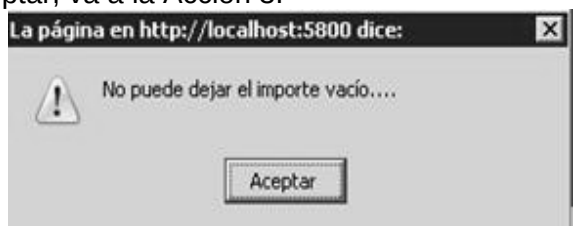
**Acción 3**: Si el Cajero presiona Cancelar (16) ó (17) se termina el caso de uso.
**Sección Nuevo:**
**Acción 3**: Si el Cajero presiona Cancelar (16) ó (17) se termina el caso de uso.
**Acción 4:** Si el Cajero no inserta el importe el sistema muestra el siguiente mensaje, el Cajero presiona Aceptar y va a la Acción 2.




**Acción 4:** Si el Cajero tiene un anticipo pendiente por liquidar el sistema muestra el siguiente mensaje, el Cajero presiona Aceptar, va a la Acción 2.




**Sección Modificar:**
**Acción 5:** Si el Cajero deja el importe en blanco el sistema muestra el siguiente mensaje, el Cajero presiona Aceptar, va a la Acción 3.




**Acción 7:** Si el Cajero presiona cancelar el sistema no ejecuta ninguna acción.
**Sección Eliminar:**
**Acción 5:** Si el Cajero presiona cancelar el sistema no ejecuta ninguna acción.

| **Acción 6:** Si el anticipo no es el último insertado el sistema lo anula. Va a la Acción 7. |
|---|

| **Caso de uso:** | Liquidar Anticipo |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Liquidar los anticipos pendientes. |
| **Resumen:** | El Cajero inicia este caso de uso cuando decide Liquidar un anticipo pendiente. Termina el caso de uso cuando culmina la acción. |
| **Precondiciones:** | Debe existir el anticipo pendiente a liquidar. |
| **Poscondiciones:** | Se liquida un anticipo. |
| **Referencias:** | R5, R5.1, R5.2 |
| **Interfaz I** | |


(1) Unidad Militar que hace la Liquidación, el sistema la muestra de forma automática.

(2) Fecha en que se realiza la Liquidación, el sistema la muestra de forma automática.

(3) Grid en el que se muestran de forma automática los anticipos pendientes a liquidar.

| | |
|---|---|
| (6) Paginado que permite navegar por los resultados del grid (3).<br><br>(7) Botón que le permite al Cajero salir del caso de uso cuando lo desee. | |
| **Curso normal de eventos para el caso de uso** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El Cajero selecciona del menú principal la opción Liquidar Anticipos. | 2. El sistema muestra la interfaz con los anticipos que se encuentran pendientes a liquidar en el grid (3). |
| 3. El Cajero selecciona el anticipo a liquidar en el grid. | |
| 4. El Cajero presiona Liquidar. | 5. El sistema liquida el anticipo. |
| **Cursos alternos** | |
| **Acción 4:** Si el Cajero presiona Cancelar (5) ó (7) se termina el caso de uso. | |

| | |
|---|---|
| **Caso de uso:** | Cuadrar Caja |
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Realizar el cuadre diario de caja. |
| **Resumen:** | El Cajero inicia este caso de uso cuando decide hacer el cuadre diario de caja. Termina el caso de uso cuando culmina la acción. |
| **Precondiciones:** | |
| **Poscondiciones:** | Se realiza el cuadre de caja. |
| **Referencias:** | R8, R8.1 |
| **Interfaz I** | |


(1) Importe que corresponde a las nóminas especiales.

(2) Importe correspondiente al concepto de Haberes y Salarios.

(5) Botón que le permite al Cajero salir del caso de uso cuando lo desee.

**Curso normal de eventos para el caso de uso**

| **Acción del actor** | **Respuesta del sistema** |
|---|---|
| 1. El Cajero selecciona del menú principal la opción Cuadrar Caja. | 2. El sistema muestra la interfaz con las opciones Aceptar y Cancelar activas. |
| 3. El Cajero inserta el importe por nóminas especiales (1) y el importe por haberes y salarios(2) | |
| 4. El Cajero presiona Aceptar. | 5. El sistema muestra la siguiente interfaz con los datos del cuadre de caja.<br>CUADRE DE CAJA 08 DE 06 DEL 2007<br>SALDOS DE CAJA ANTERIOR: 74.00<br>INGRESOS: 0.00<br>EGRESOS: 15.00<br>SALDO: 59.00<br>VALES DE CAJA: 15.00<br>NOMINAS ESPECIALES: 4<br>HABERES Y SALARIOS: 5<br>SALDO DE EFECTIVO: 35.00<br>IMPRIMIR |

| | |
|---|---|
| 6. El Cajero presiona imprimir. | 7. El sistema imprime el cuadre de caja. |
| | 8. Se termina el caso de uso. |
| **Cursos alternos** | |
| **Acción 4:** Si el Cajero presiona Cancelar (4) ó (5) se termina el caso de uso.<br>**Acción 6:** Si el Cajero no presiona imprimir el sistema no ejecuta ninguna acción. | |

| | |
|---|---|
| **Caso de uso:** | Controlar Anticipos |
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Obtener reporte de los Anticipos entregados y liquidados. |
| **Resumen:** | El Cajero inicia este caso de uso cuando decide obtener el reporte de los Anticipos entregados y Liquidados hasta el momento, tiene la opción de imprimirlo. Termina el caso de uso cuando culmina la acción. |
| **Precondiciones:** | |
| **Poscondiciones:** | Se obtiene el Registro de Control de Anticipos (SCF-05). |
| **Referencias:** | R7, R7.1, R7.2 |
| **Interfaz I** | |

MINFAR

UM:1056

REGISTRO Y CONTROL DE ANTICIPOS
SCF-05

No:

| FECHA | NUMEROS | | CO | A NOMBRE DE | DEBE | HABER | SALDO | ANALISIS DE LA LIQUIDACION | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ENTREGA | LIQUIDA | | | | | | VENCE | LIQUIDACION | No LIQU |
| 2007-05-22 | 6 | 0 | ---- | Gretter Mora Viera | 25.00 | 0.00 | 25.00 | 2007-05-25 | - | 0 |
| 2007-05-22 | 7 | 0 | ---- | Maxwell Chirino | 25.00 | 0.00 | 50.00 | 2007-05-25 | - | 0 |
| 2007-05-23 | 8 | 1 | ---- | Gretter Mora Viera | 85.00 | 85.00 | 50.00 | 2007-05-16 | 2007-05-23 | 1 |

| **Curso normal de eventos para el caso de uso** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 9. El Cajero selecciona del menú principal la opción Registro y Control de Anticipos. | 10. El sistema muestra la interfaz con el Registro y Control de Anticipos. |

| 11. El Cajero presiona Imprimir. | 12. El sistema imprime el Registro. |
|---|---|
| **Cursos alternos** | |
| **Acción 12:** Si el Cajero no presiona Imprimir el sistema no ejecuta ninguna acción. | |

| **Caso de uso:** | Registrar Pagos Menores |
|---|---|
| **Actores:** | Cajero (inicia) |
| **Propósito:** | Obtener reporte del movimiento de efectivo en caja. |
| **Resumen:** | El caso de uso lo inicia el Cajero cuando decide obtener un reporte de las operaciones realizadas de entrega de efectivo por concepto de anticipo para gastos y pagos menores por caja y de recibo de efectivo, además de ofrecer la opción de imprimir dicho registro. |
| **Precondiciones:** | |
| **Poscondiciones:** | Se obtiene el Registro Movimiento de Efectivo para Pagos Eventuales (SCF-04). |
| **Referencias:** | R6, R6.1, R6.2 |

**Interfaz I**

| MINFAR<br>UM:1056 | MOVIMIENTO DE EFECTIVO PARA PAGOS EVENTUALES<br>SCF - 03 | | | | | AUTORIZADO POR:<br>No. | FECHA<br>23/05/2007 |
|---|---|---|---|---|---|---|---|
| ANO:2007<br>23/05/07 | REFERENCIA<br>CLAVE | No | DETALLE | CO | DEBE | HABER | SALDO |
| 2007-05-22 | SCF-01 | | RECIBO DE EFECTIVOS | | 52.00 | | 52.0 |
| 2007-05-22 | SCF-01 | | RECIBO DE EFECTIVOS | | 21.00 | | 73.0 |
| 2007-05-22 | SCF-02 | | VALES PARA PAGOS MENORES | | 60.00 | | 133.0 |
| 2007-05-22 | SCF-02 | | VALES PARA PAGOS MENORES | | 32.00 | | 165.0 |
| 2007-05-22 | SCF-03 | | ANTICIPOS Y LIQUIDACION | | | 25.00 | 140.0 |
| 2007-05-22 | SCF-03 | | ANTICIPOS Y LIQUIDACION | | 25.00 | | 165.0 |
| 2007-05-23 | SCF-03 | | ANTICIPOS Y LIQUIDACION | | | 85.00 | 80.0 |
| | | | **Totales** | | 190.00 | 110.00 | 80.00 |

| **Curso normal de eventos para el caso de uso** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El actor selecciona del menú principal la opción Movimiento de Efectivo para Pagos Eventuales. | 2. El sistema muestra la interfaz I con el Movimiento de Efectivo para Pagos Eventuales. |

| | |
|---|---|
| 3. El Cajero presiona Imprimir. | 4. El sistema imprime el Registro. |
| **Cursos alternos** | |
| **Acción 4:** Si el Cajero no presiona Imprimir el sistema no ejecuta ninguna acción. | |

**Conclusiones:**

En este capítulo se han tomado decisiones importantes, pues se ha comprendido a fondo la dinámica del negocio, obteniendo de forma correcta los requisitos necesarios que el software debe cumplir, tanto funcionales como no funcionales. Se ha obtenido además el modelo del sistema, especificando detalladamente cada caso de uso, de forma tal que se puede continuar con el próximo flujo de trabajo de ingeniería, el análisis y diseño.

## Capitulo 3. Análisis y Diseño del sistema.

### Introducción.

En el presente capítulo, se pretende comenzar con el siguiente flujo de ingeniería después de la captura de requisitos, en el cual se obtuvo una vista externa del sistema puntualizando de forma concreta y detallada los procesos que soporta el mismo. El siguiente flujo de trabajo pertenece fundamentalmente a la fase de elaboración y constituye el análisis y diseño del sistema; se centra principalmente en obtener una vista interna del mismo, profundizando en cómo realizar cada uno de los procesos identificados como casos de uso, especificándolos de una forma más detallada y definiendo las clases necesarias para realizar las funcionalidades previstas. Serán analizados además una serie de patrones que influirán en la construcción de la arquitectura y en los mecanismos de diseño de la misma.

### Análisis.

El Modelo de análisis se realiza para obtener una visión del sistema sobre los requisitos funcionales expresados ya en un lenguaje técnico, constituye una primera aproximación al modelo de diseño. El mismo ofrece ventajas tales como: suavizar la transición al diseño, apoyar el cambio a otra plataforma de programación, servir para obtener una visión general de la propuesta del sistema así como para planificar y dividir el diseño e implementación en pequeños módulos, apoya la aplicación de reingeniería a aplicaciones existentes (al ser en un lenguaje menos técnico ayuda a entender mejor la propuesta de solución). Está orientado a analizar como el sistema va a cumplir sus funcionalidades.

El análisis debe, pues, capturar los requisitos de usuario sin adoptar prematuramente decisiones de implementación, es decir, omitiendo detalles dependientes de la tecnología, y utilizando conceptos extraídos únicamente del dominio del problema.

A continuación se representan los modelos del análisis por cada uno de los casos de uso definidos.

**Modelo de clases del análisis.**

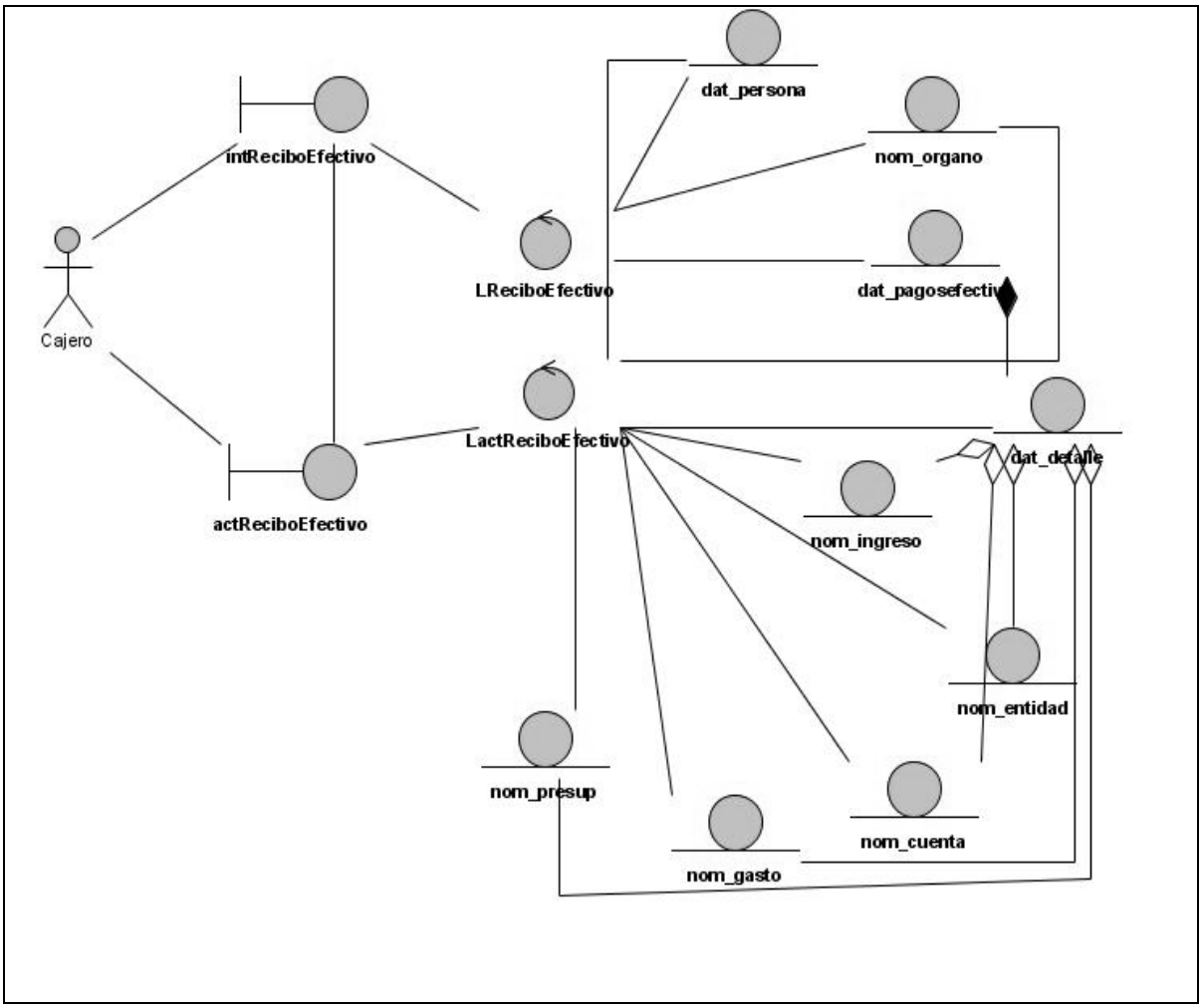



**Fig.7 Diagrama de clases del análisis. Caso de Uso Recibir Efectivo.**

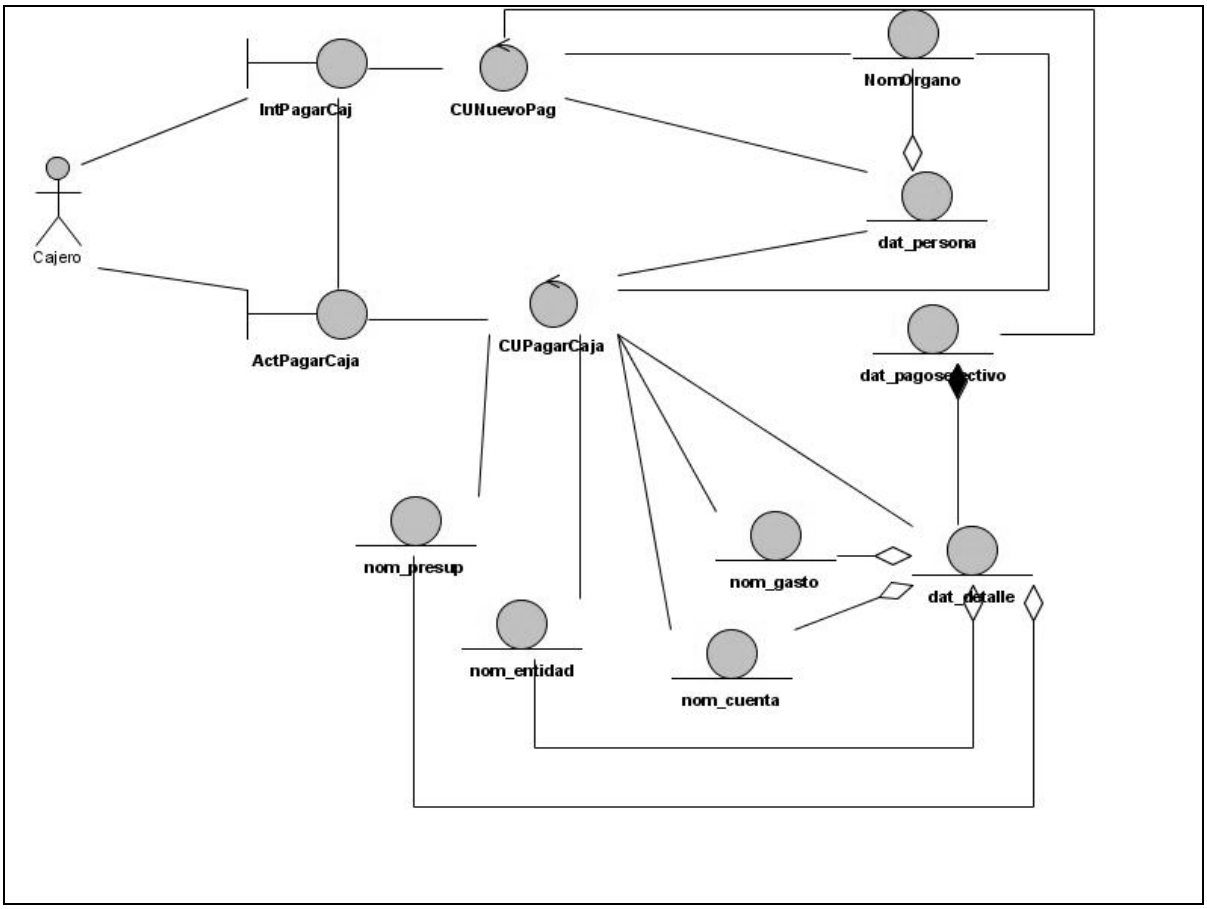



**Fig.8 Diagrama de clases del análisis. Caso de Uso Pagar por caja.**

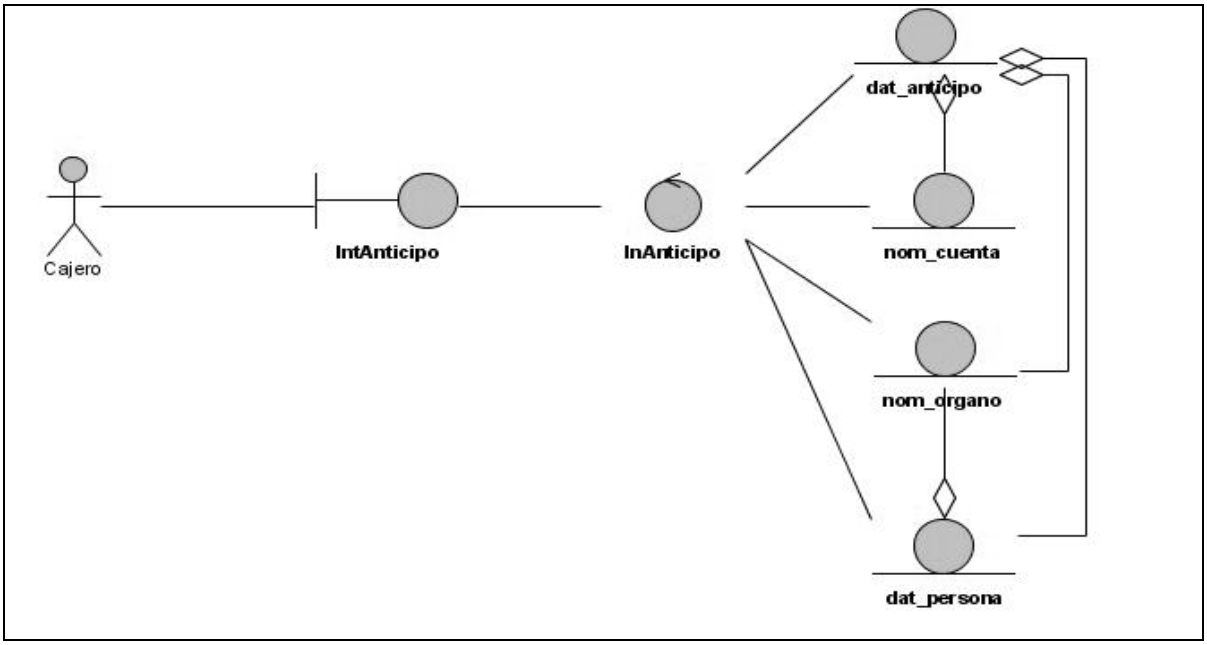



**Fig.9 Diagrama de clases del análisis. Caso de Uso Entregar Anticipos.**

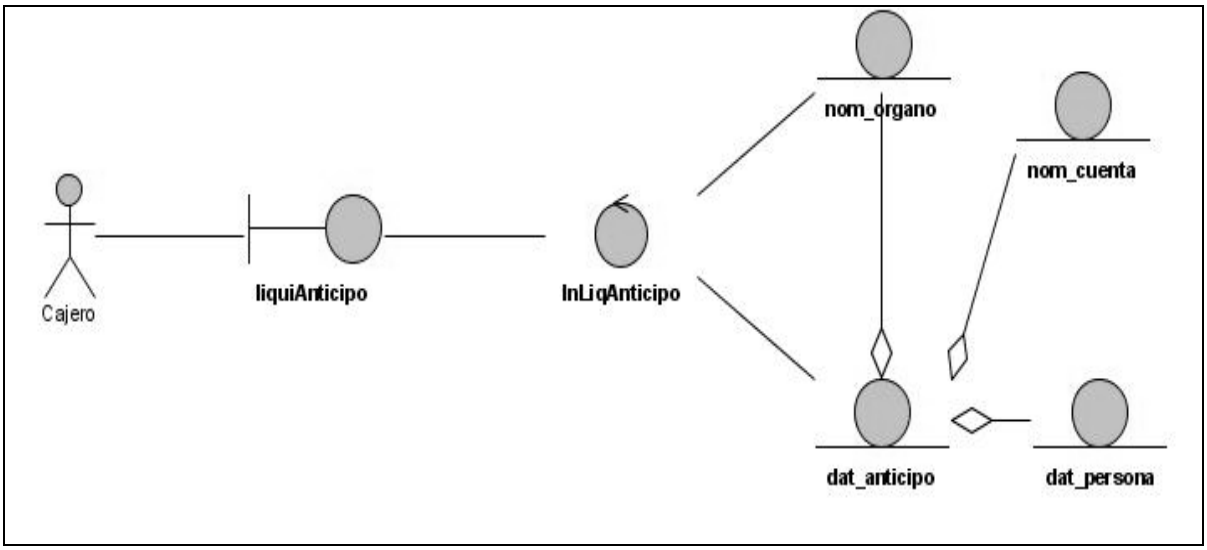



**Fig. 10 Diagrama de clases del análisis. Caso de Uso Liquidar Anticipo.**

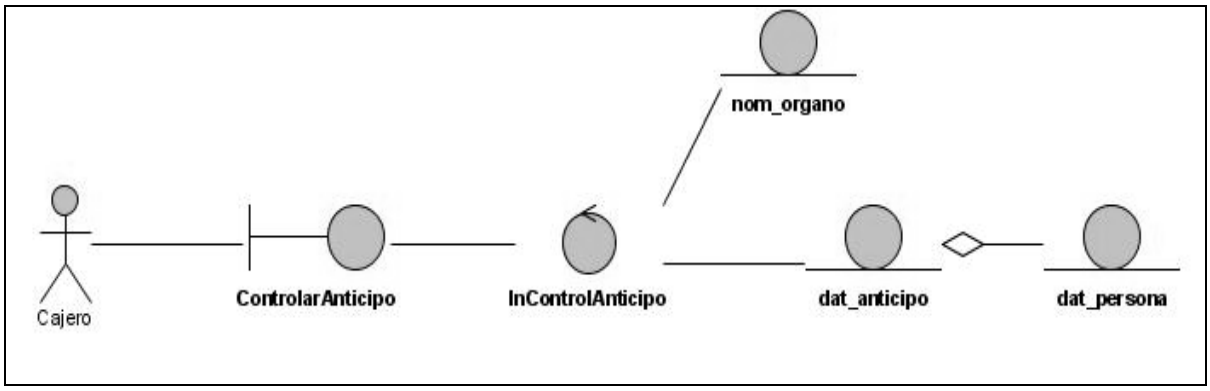



**Fig. 11 Diagrama de clases del análisis. Caso de Uso Controlar Anticipos.**

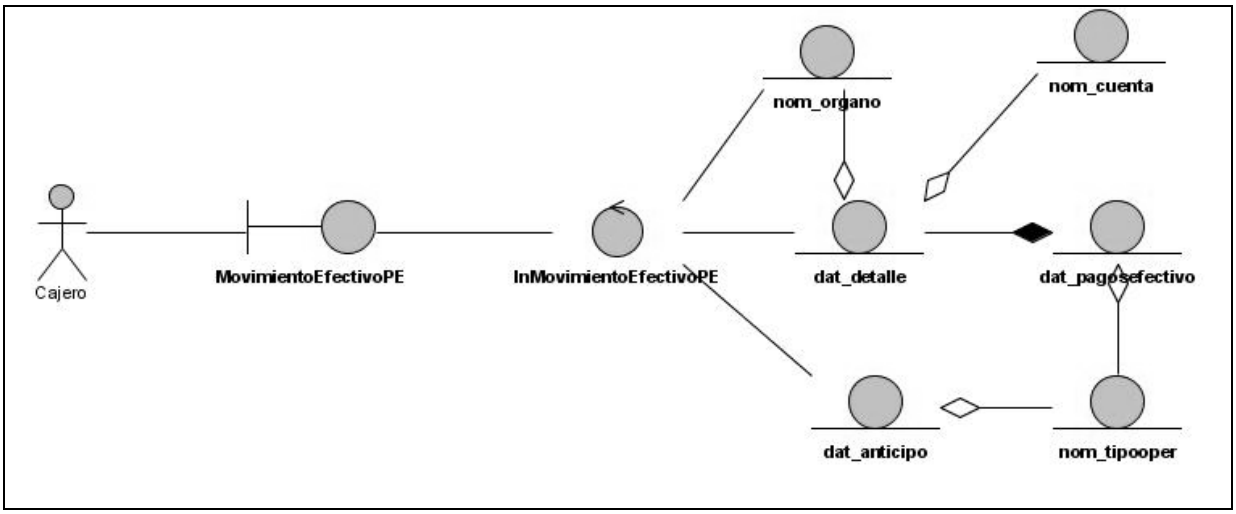



**Fig. 12 Diagrama de clases del análisis. Caso de Uso Registrar Pagos Menores.**

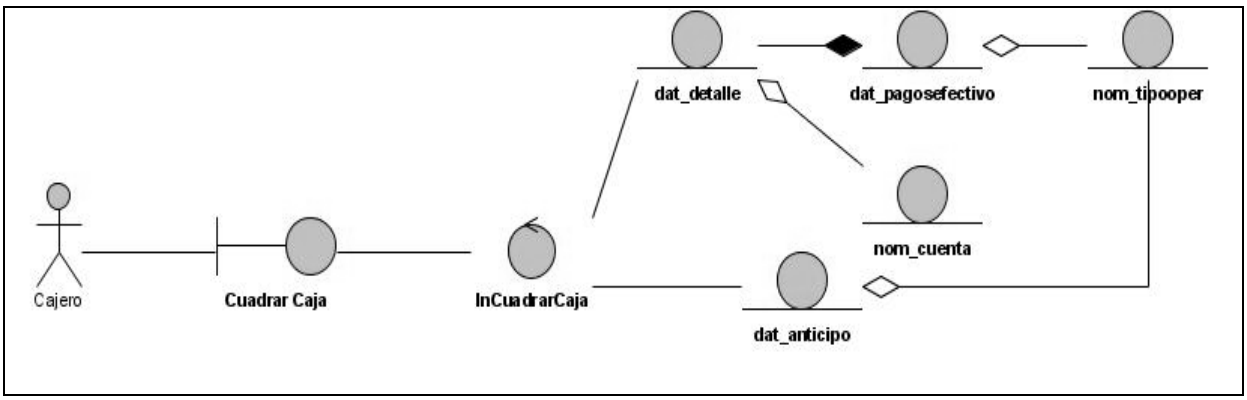



**Fig. 13 Diagrama de clases del análisis. Caso de Uso Cuadrar Caja.**

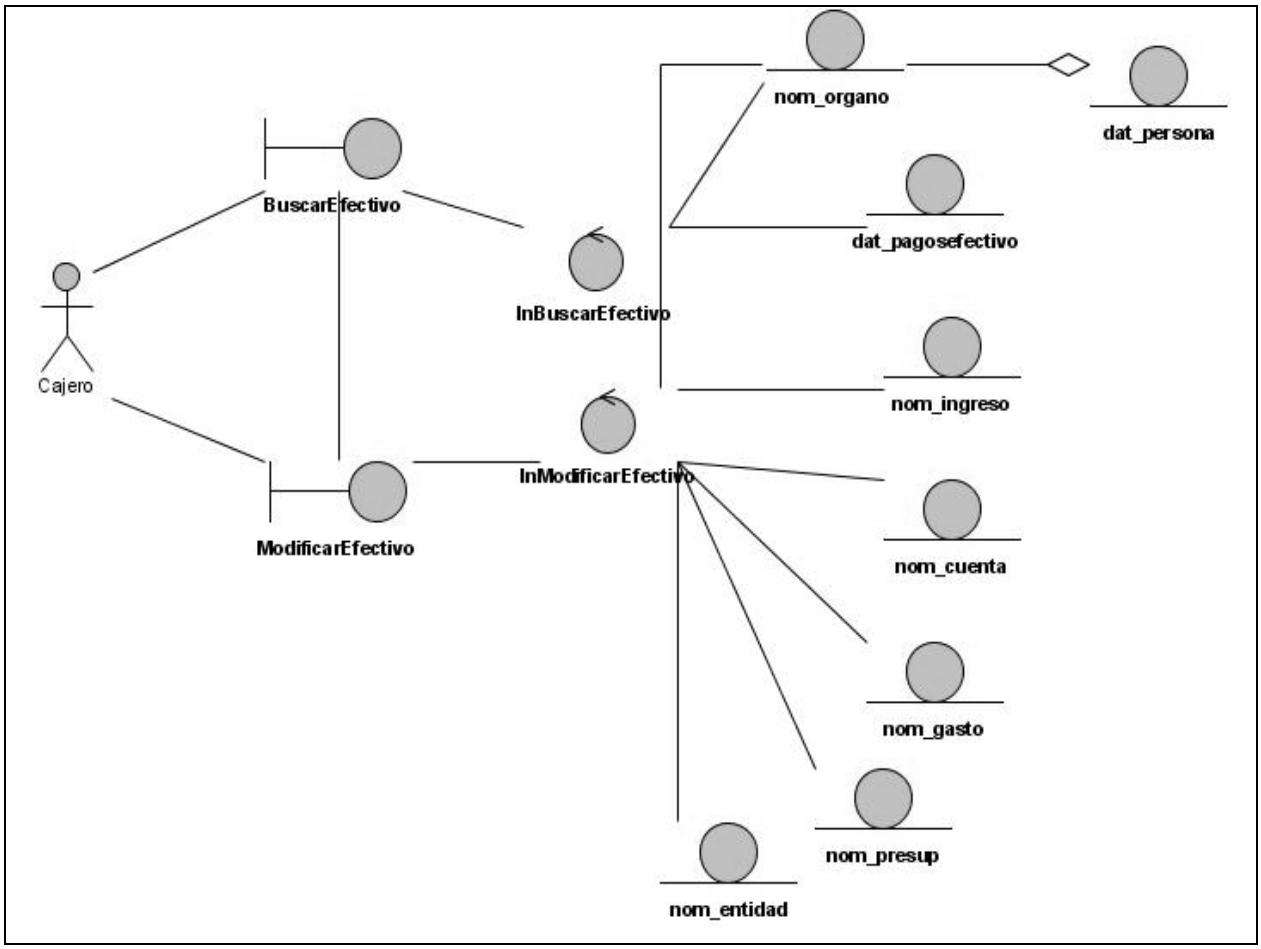



**Fig. 14 Diagrama de clases del analisis. Caso de Uso Modificar E/S Efectivo.**

**Diseño.**

Al contrario del análisis, el diseño debe definir una solución software que satisfaga de modo efectivo y eficiente los requisitos especificados en el análisis, y al hacer esto el modelo incorporará nuevos artefactos (nuevas clases, nuevos atributos y operaciones para las clases, etc.) y tendrá en cuenta la plataforma tecnológica concreta sobre la que debe construirse el sistema informático. De hecho, el diseño debe proporcionar una solución creativa para el problema especificado en el análisis.

A continuación se representan las realizaciones de los principales escenarios de los casos de uso críticos, como pieza indispensable del modelo de diseño.

**Diagramas de interacción**

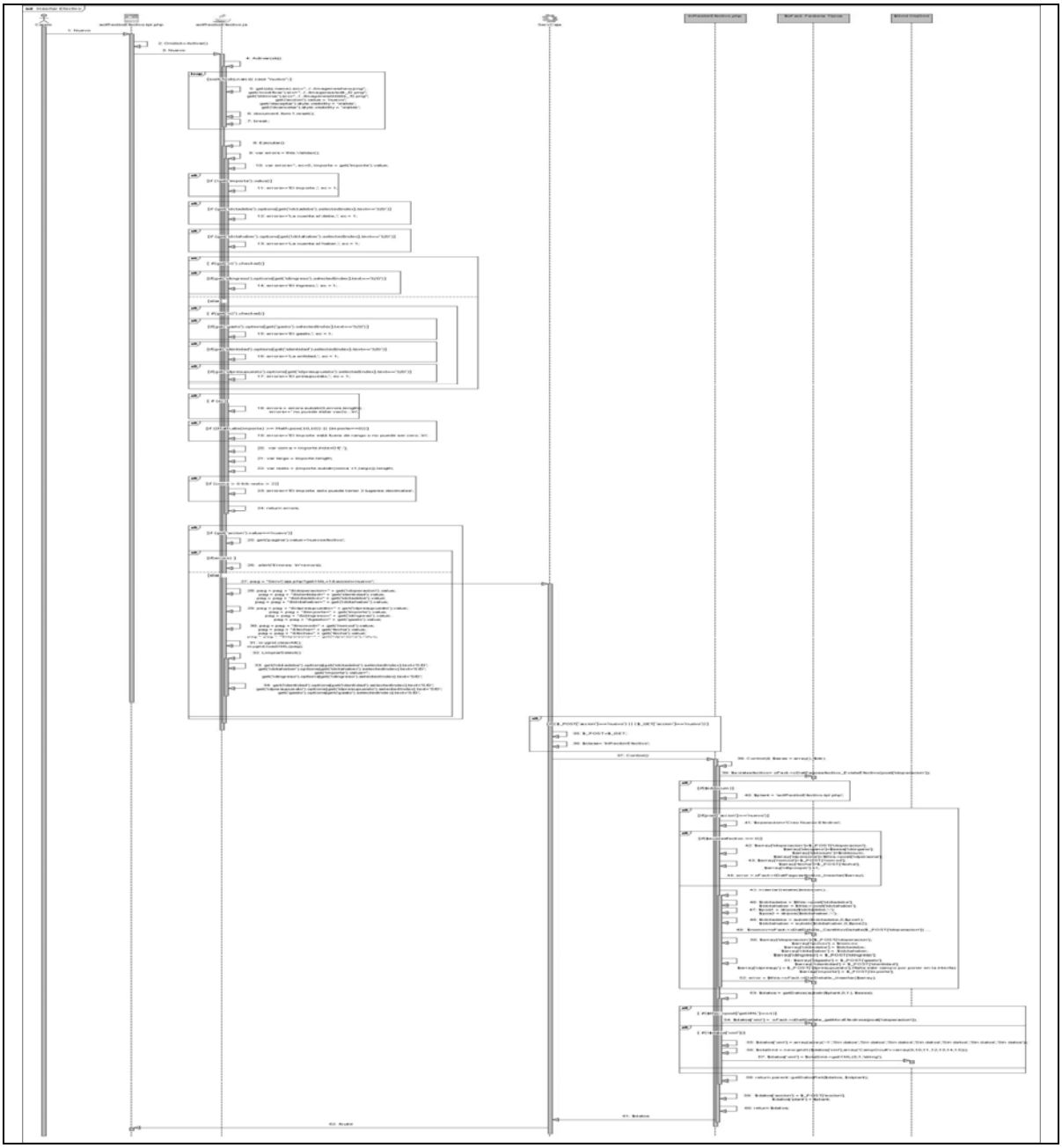


**Fig. 15 Diagrama de secuencia. Caso de Uso Recibir Efectivo. Escenario Insertar Efectivo.**

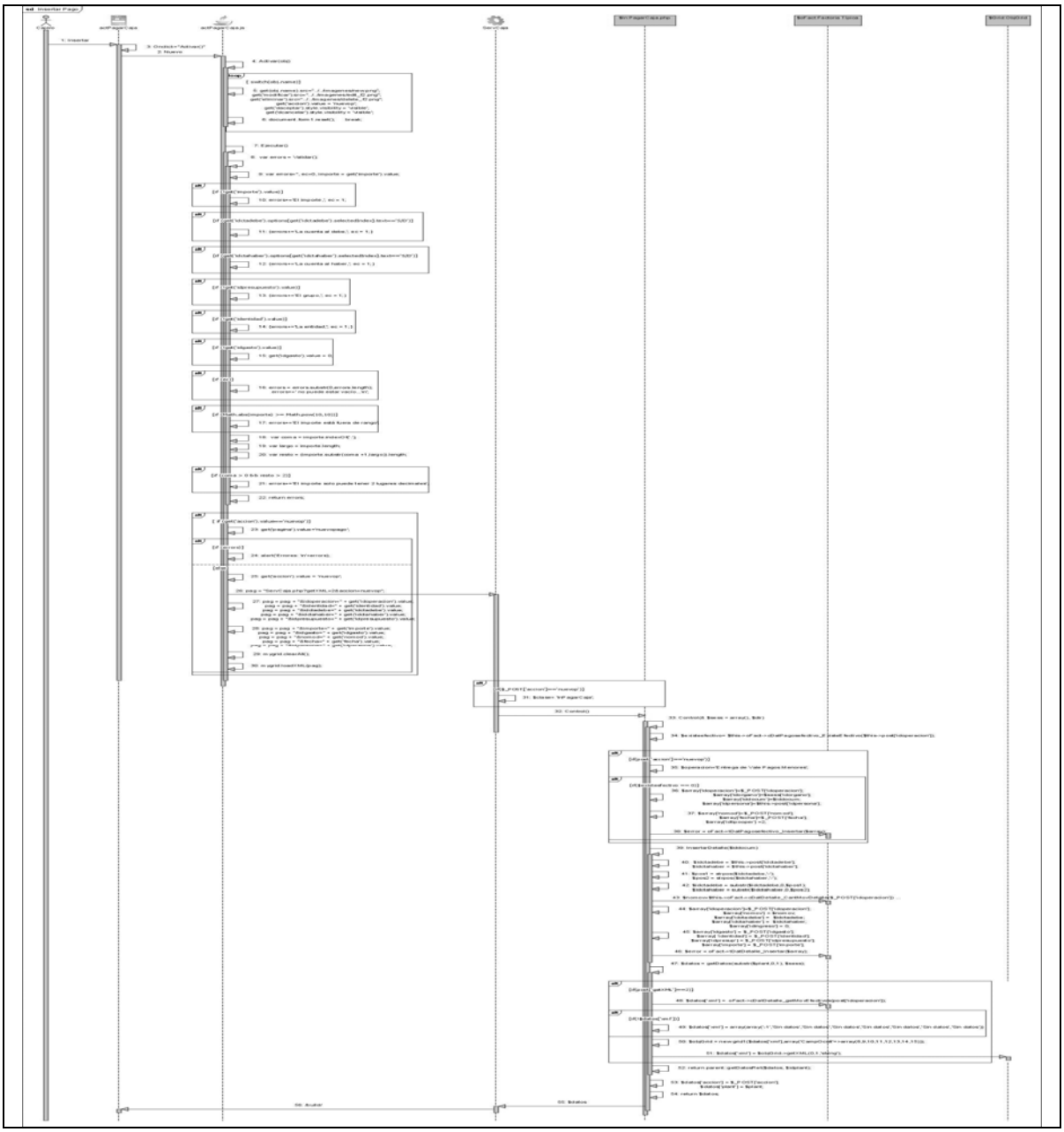


**Fig. 16 Diagrama de secuencia. Caso de Uso Pagar por Caja. Escenario Insertar Pago.**

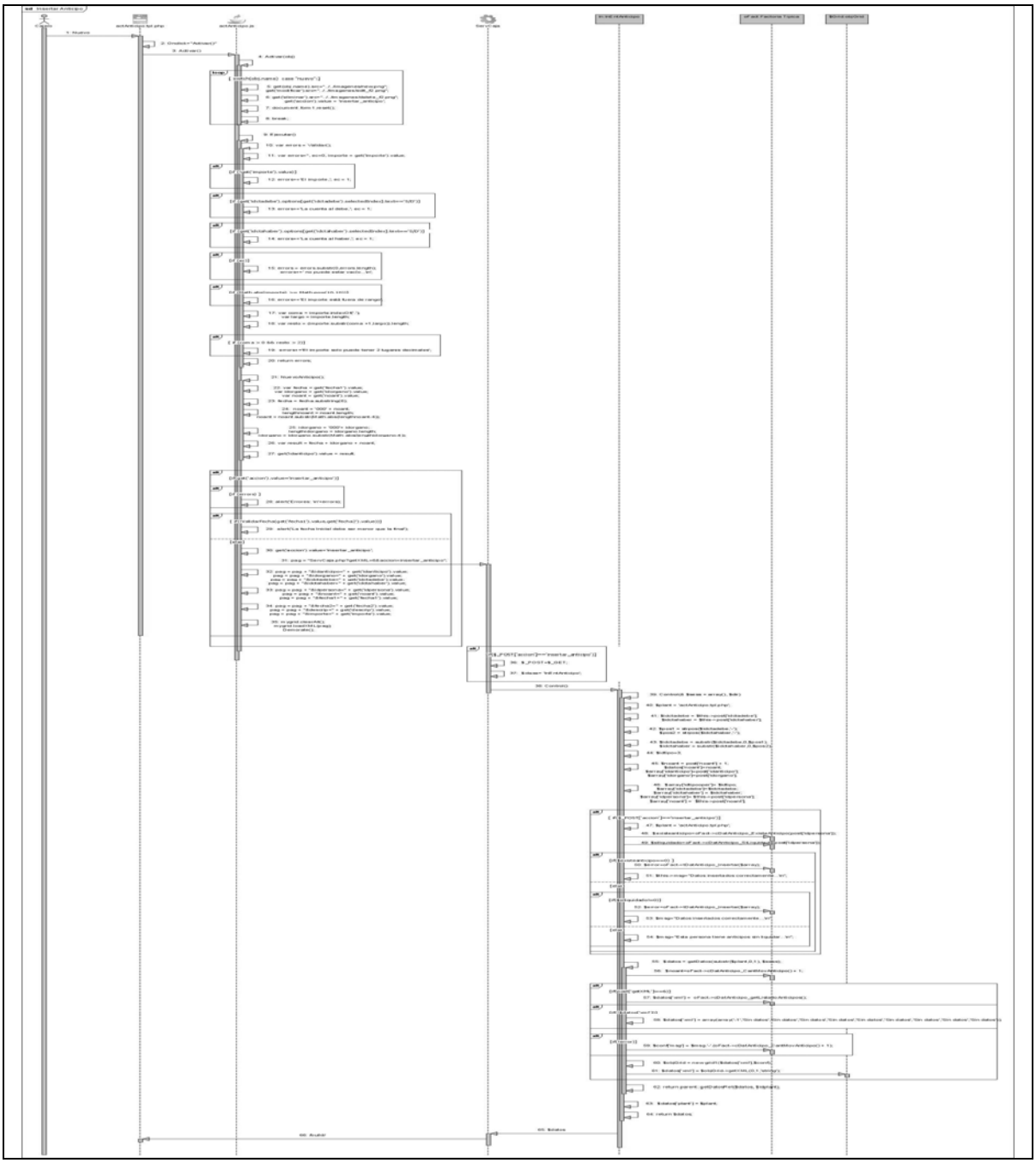


**Fig. 17 Diagrama de secuencia. Caso de Uso Entregar Anticipo. Escenario Insertar Anticipo.**

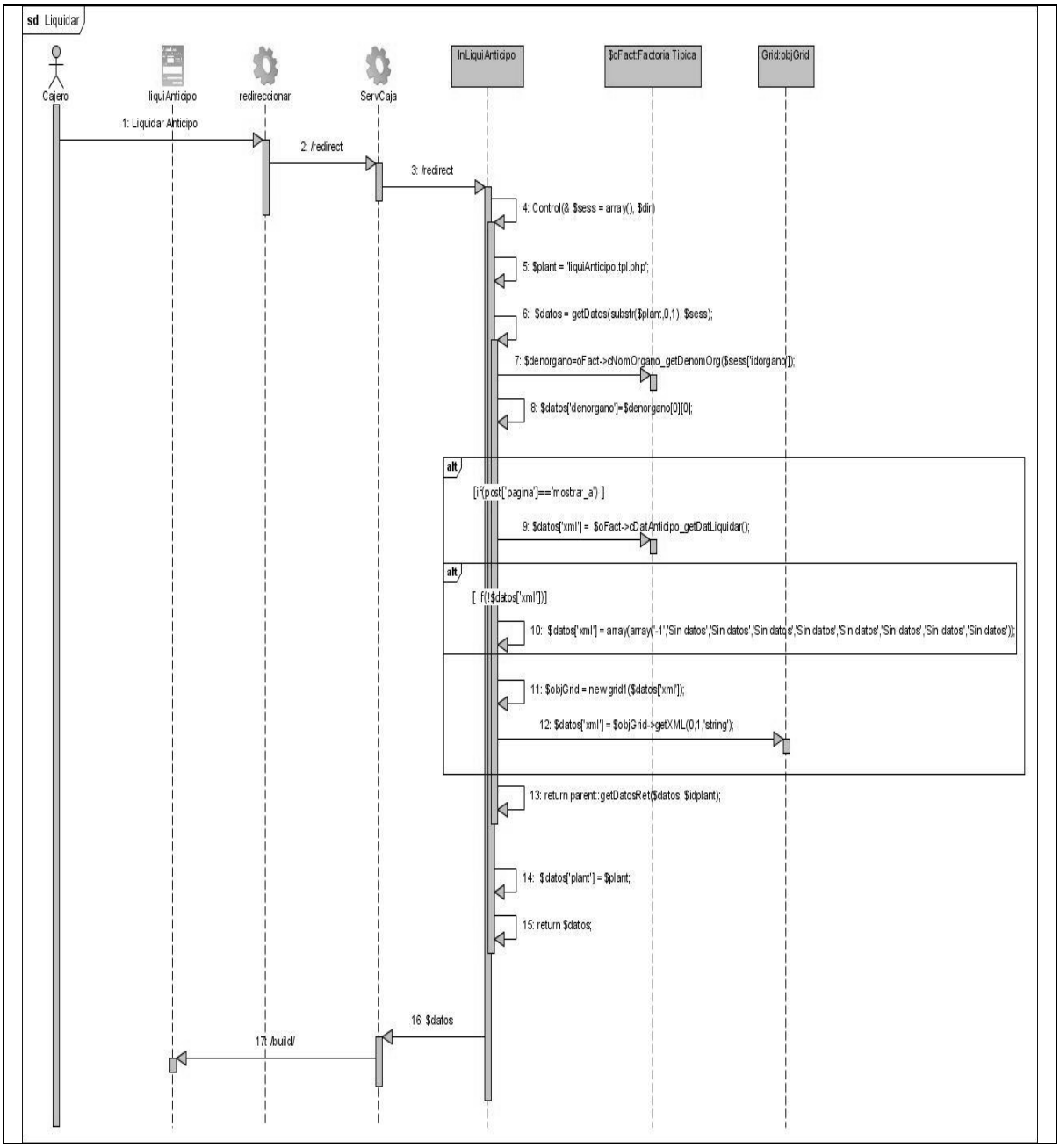



**Fig. 18 Diagrama de secuencia. Caso de Uso Liquidar Anticipos. Escenario Liquidar Anticipo.**

Como parte importante de la realización del modelo de diseño, se representan a continuación los diagramas de clases del diseño de los principales escenarios de los casos de uso críticos del sistema.

**Diagramas de clases**

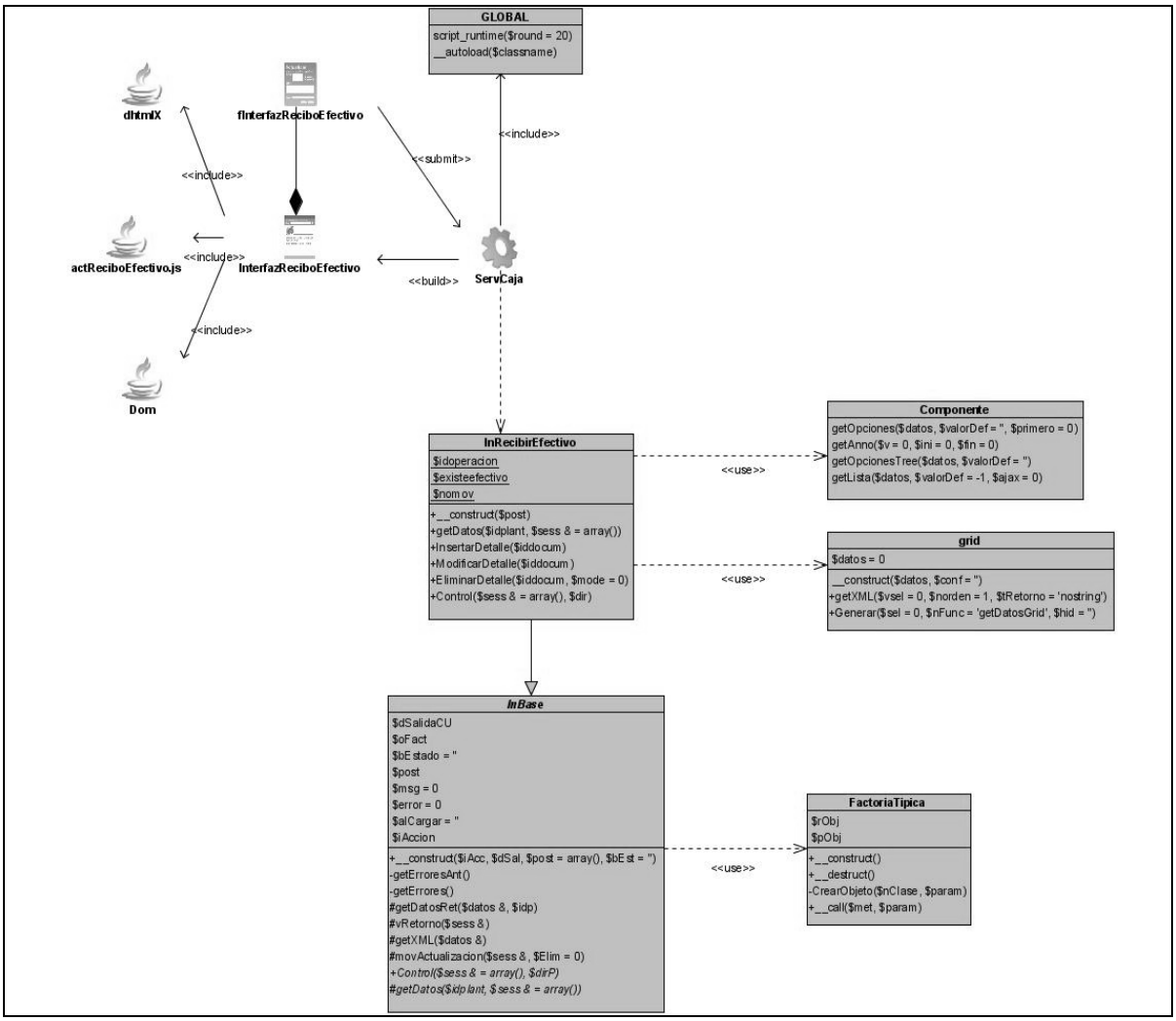



**Fig. 19 Diagrama de clases del diseño. Caso de Uso Recibir Efectivo.**

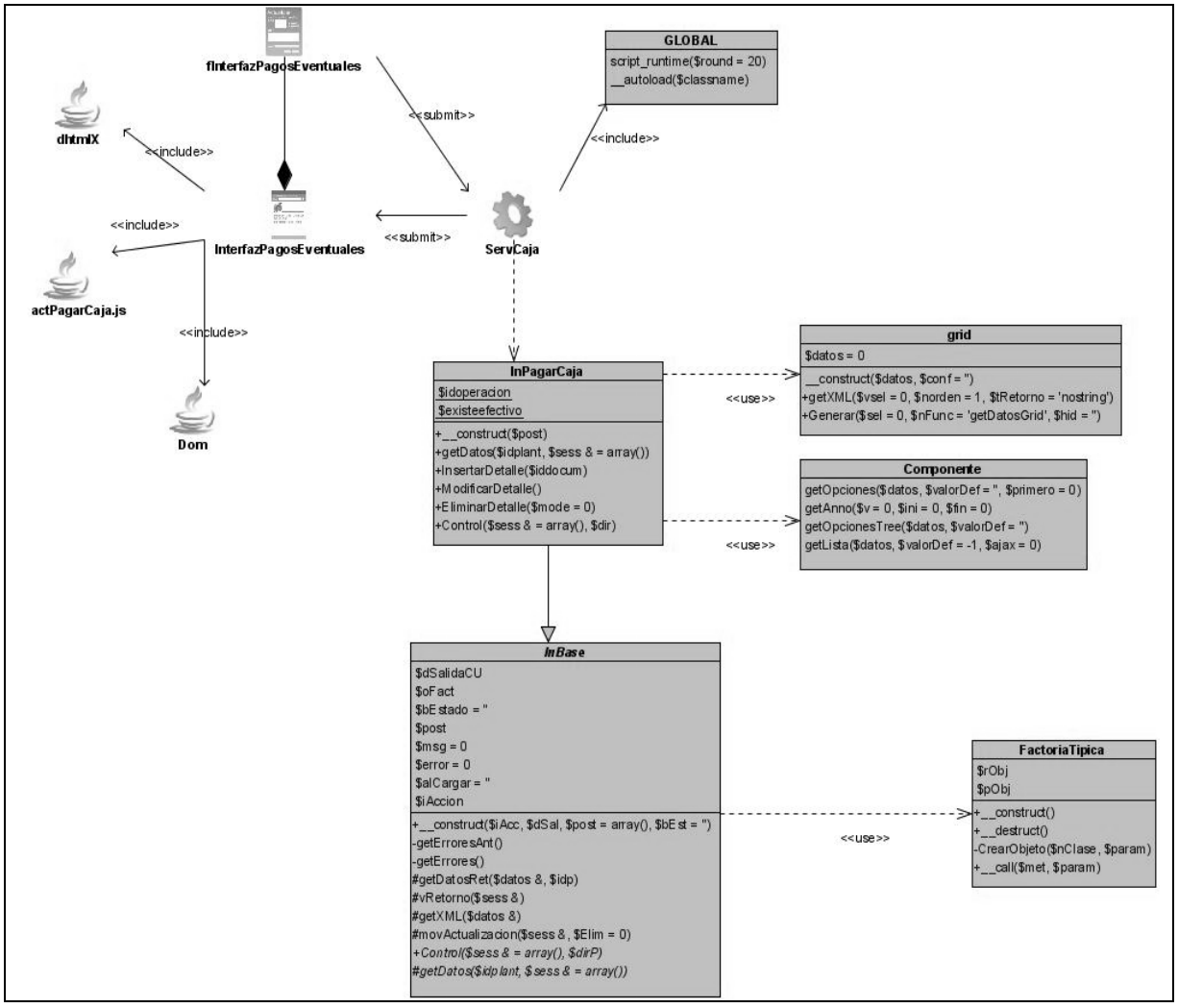



**Fig. 20 Diagrama de clases del diseño. Caso de Uso Pagar por Caja.**

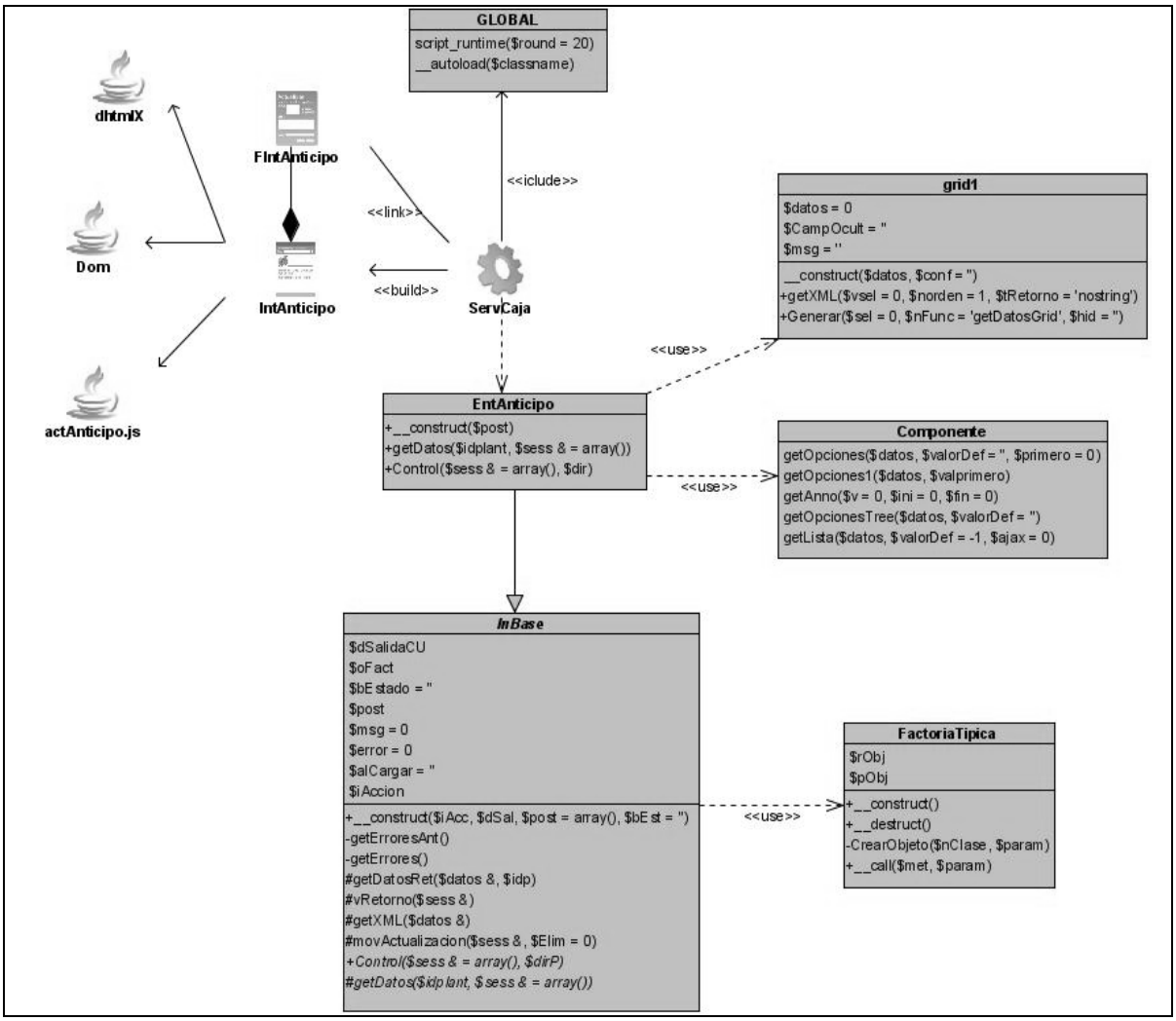



**Fig. 21 Diagrama de clases del diseño. Caso de Uso Entregar Anticipo.**

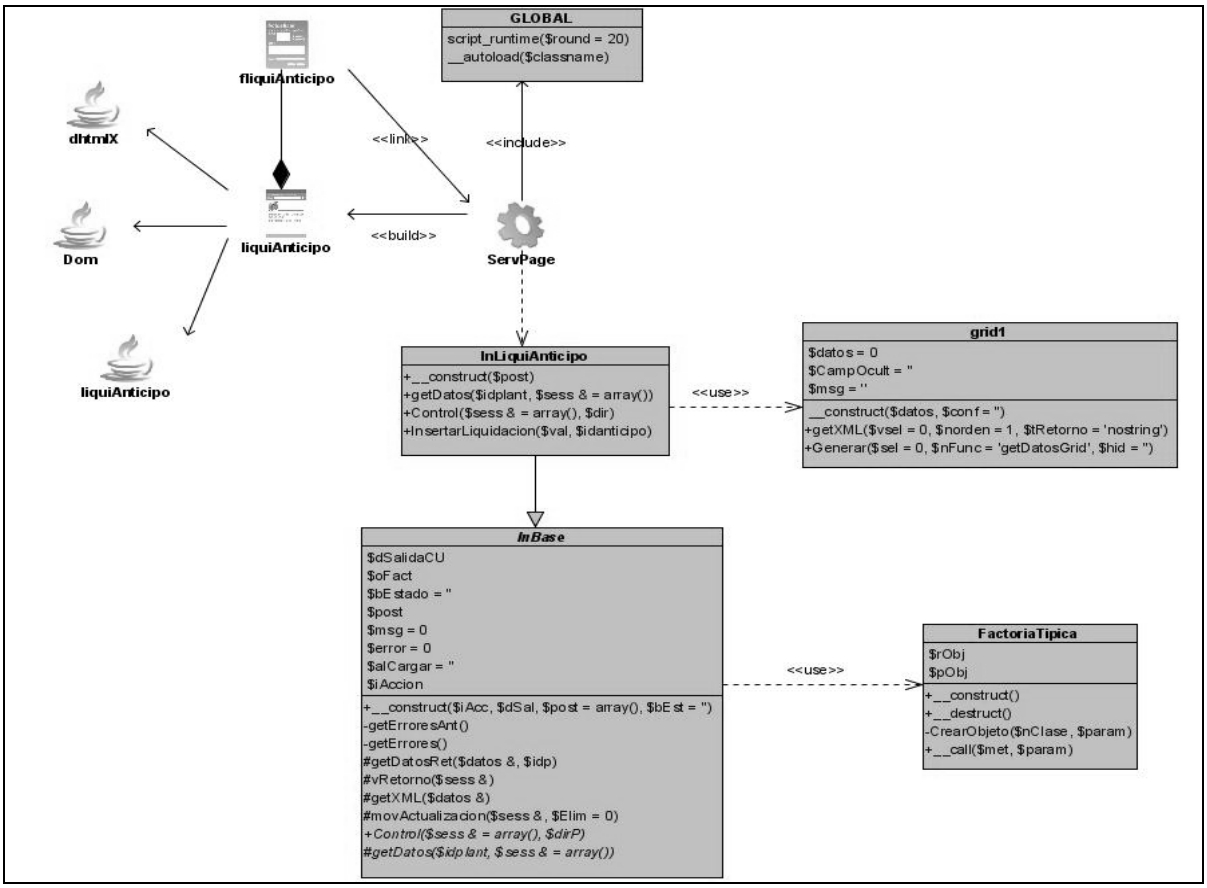



**Fig. 22 Diagrama de clases del diseño. Caso de uso Liquidar Anticipos.**

En la siguiente figura, se representa un fragmento genérico a todos los diagramas de clases anteriormente presentados que corresponde a la seguridad, la imagen parte de la idea de que la clase GLOBAL posee con controlacceso una relación de tipo include, a su vez controlacceso utiliza los métodos ofrecidos por la clase interfaz del paquete de seguridad.

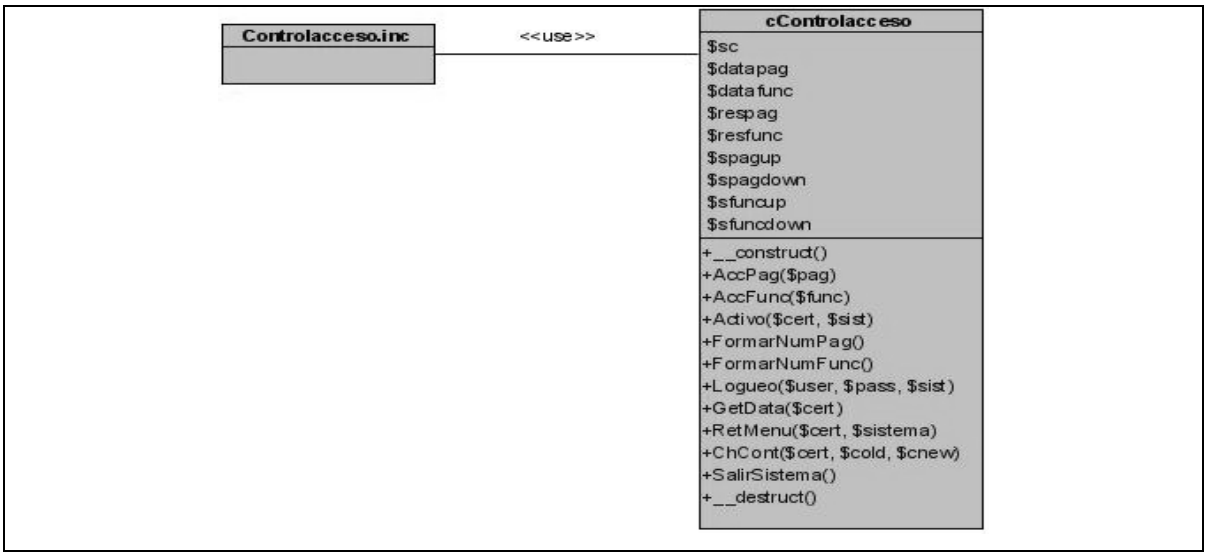



**Fig. 23 Diagrama de clases genérico.**

A continuación se detallan las clases necesarias para la implementación del sistema.

**Descripción de las clases**

| Nombre: class lnNuevoEfectivo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| datos | array |
| nomod | array |
| persona | array |
| organo | array |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Función encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: lnRecibirEfectivo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| iddocu | integer |
| denorgano | array |
| idconcepto | array |
| idingreso | array |
| identidad | array |
| idctadebe | array |
| idctahaber | array |
| idgasto | array |
| idpresup | array |
| datos | array |
| objGrid | Grid1 |
| fechaactual | date |
| horaactual | date |
| plant | String |
| existeefectivo | integer |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function InsertarDetalle($iddocum) | Función encargada de insertar los datos de los detalles de un recibo de efectivo. |
| function ModificarDetalle($iddocum) | Función encargada de modificar los datos de los detalles de un recibo de efectivo. |
| function EliminarDetalle($iddocum,$mode = 0) | Función encargada de eliminar los datos de los detalles de un recibo de efectivo. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnImprimirRecibo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| denorgano | array |
| idoperacion | array |
| resnomod | array |
| persona | array |
| resulrecibo | array |
| nomov | array |
| conceptod | array |
| conceptoh | array |
| recibo | array |
| total | numeric |
| plant | String |
| datos | array |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Función encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function ElaborarEfectivo($resulrecibo) | Función encargada de procesar la lógica del reporte a imprimir. |
| function Reporte($arreglo) | Función encargada de construir la inferfaz del reporte y de calcular el saldo total a mostrar en el mismo. |
| function MostrarRecibo() | Función encargada de mostrar el reporte a imprimir. |
| function Control(& $sess = array(), $dirP) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnNuevoPago | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| datos | array |
| nomod | integer |
| persona | array |
| organo | array |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Función encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: lnPagarCaja | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| iddocu | integer |
| denorgano | array |
| nombp | array |
| identidad | array |
| idcuenta | array |
| idgasto | array |
| idpresup | array |
| idctadebe | array |
| idctahaber | array |
| datos | array |
| objGrid | Grid1 |
| iduser | integer |
| fechaactual | date |
| horaactual | date |
| plant | String |
| existeefectivo | integer |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = | Esta función es la encargada de recuperar |

| | |
|---|---|
| array()) | la información necesaria en los componentes de la plantilla correspondiente. |
| function InsertarDetalle($iddocum) | Función encargada de insertar los datos de los detalles de un pago por caja. |
| function ModificarDetalle() | Función encargada de modificar los datos de los detalles de un pago por caja. |
| function EliminarDetalle($mode = 0) | Función encargada de eliminar los datos de los detalles de un pago por caja. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnImprimirPmenor | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| denorgano | array |
| idoperacion | array |
| resnomod | array |
| idoperacion | array |
| resulrecibo | array |
| nomov | array |
| conceptod | array |
| conceptoh | array |
| recibo | array |
| total | numeric |
| plant | String |
| datos | array |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Función encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function ElaborarPago($resulrecibo) | Función encargada de procesar la lógica del reporte a imprimir. |
| function Reporte($arreglo) | Función encargada de construir la inferfaz del reporte y de calcular el saldo total a mostrar en el mismo. |
| function MostrarPago() | Función encargada de mostrar el reporte a imprimir. |
| function Control(& $sess = array(), $dirP) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnBuscarEfectivo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| datos | array |
| idmod | array |
| persona | array |
| nombpers | array |
| objGrid | Grid1 |
| result | array |
| plant | String |
| max | integer |
| actual | array |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function Control(& $sess = array() | Función encargada de llamar a la plantilla correspondiente |
| function BuscarEfectivo($iddocum,$organo,$fecha,$idpersona) | Función para la búsqueda de un modelo (ya sea de un recibo de efectivo o de un pago por caja. |
| function EliminarEfectivo($idoperacion,$mode = 0) | Función para eliminar un modelo (ya sea un recibo de efectivo o un pago por caja). |
| function ModificarEstado($val,$idoperacion) | Función para modificar los datos de un modelo(ya sea de un recibo de efectivo o de un pago por caja) |

| Nombre: class lnModificarEfectivo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| datos | array |
| idingreso | array |
| idctadebe | array |
| idctahaber | array |
| idgasto | array |

| | |
|---|---|
| idpresup | array |
| identidad | array |
| objGrid | Grid1 |
| plant | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function Modificar() | Función para modificar los detalles de un modelo (ya sea un recibo de efectivo o un vale para pagos menores.) |
| function Eliminar() | Función eliminar los detalles de un modelo (ya sea de un recibo de efectivo o de un pago por caja. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnEntAnticipo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| oComp | Componente |
| datos | array |
| noant | integer |
| idctadebe | array |
| idctahaber | array |
| persona | array |
| objGrid | Grid1 |
| plant | String |
| existeanticipo | array |
| siliquidado | integer |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnLiquiAnticipo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| datos | array |
| objGrid | Grid1 |
| plant | String |
| result | array |
| array | array |
| fecha | date |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| | |
| function Control(& $sess = array(), $dir) | Función encargada de llamar a la plantilla correspondiente. |
| function InsertarLiquidacion($val,$idanticipo) | Función para liquidar un anticipo. |

| Nombre: class lnControlAnticipo | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| resultado | array |
| organo | array |
| denorgano | array |
| datos | array |
| saldo | float |
| plant | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| ElaboraControl($resul) | Función encargada de procesar la lógica del reporte a imprimir. |
| function Reporte($matriz) | Función encargada de construir la inferfaz del reporte y de calcular el saldo total a mostrar en el mismo. |

| | |
|---|---|
| function MostrarRControlA() | Función encargada de mostrar el reporte a imprimir. |
| function SaldoTotal($saldo,$haber,$debe) | Función encargada de calcular el saldo total de las cuentas. |
| function Concatena($valor) | Función encargada de eliminar el signo negativo a los saldos. |
| function Control(& $sess = array(), $dirP) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class lnMovEfectivoPE | |
|---|---|
| Tipo de clase: controladora | |
| Atributo | Tipo |
| resul | array |
| organo | array |
| denorgano | array |
| datos | array |
| saldo | float |
| plant | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct($post) | Función para inicializar los objetos de la clase. |
| function getDatos($idplant,& $sess = array()) | Esta función es la encargada de recuperar la información necesaria en los componentes de la plantilla correspondiente. |
| ElaboraMovEfectivoPE | Función encargada de procesar la lógica del reporte a imprimir. |
| function Reporte($matriz) | Función encargada de construir la inferfaz del reporte y de calcular el saldo total a mostrar en el mismo. |
| function MostrarMovEfectivoPE | Función encargada de mostrar el reporte a imprimir. |
| function SaldoTotal($saldo,$haber,$debe) | Función encargada de calcular el saldo total de las cuentas. |
| function Concatena($valor) | Función encargada de eliminar el signo negativo a los saldos. |
| function Control(& $sess = array(), $dirP) | Función encargada de llamar a la plantilla correspondiente. |

| Nombre: class cDatPagosefectivo | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| efectivo | tDatPagosefectivo |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idoperacion = '' ,$idorgano = '' ,$iddocum = '' ,$idpersona = '' ,$nomod = '' ,$fecha = '' ,$idtipooper = '' ) | Función que devuelve todos los datos de la tabla dat_pagosefectivo. |
| function ExisteEfectivo($idoperacion) | Función para verificar si existen datos de un modelo en la tabla dat_pagosefectivo. |
| function DevolverEfectivo($nombre) | Función para devolver los datos de un modelo de la tabla dat_pagosefectivo. |
| function getNomod() | Función para buscar los números de modelos que existen en la tabla dat_pagosefectivo. |
| function getListadoEfectivo($iddocum,$organo,$fecha,$idpersona) | Función para buscar los datos de un modelo en la tabla dat_pagosefectivo. |
| function MaximoModEfectivo() | Función para buscar el máximo del número de modelos de la tabla dat_pagosefectivo. |
| function ModActual($idoperacion) | Función para buscar un número de modelo específico. |

| Nombre: class cDatPersona | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| persona | tDatPersona |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idpersona = '' ,$codigo = '' ,$nombre = '',$idorgano = '', | Función que devuelve todos los datos de la tabla dat_persona. |

| | |
|---|---|
| $actual = '1' ) | |
| function ListaPersona() | Función que devuelve los datos de las personas que pertenecen a un determinado órgano. |
| function getDenomPersona($idpersona) | Función que devuelve el nombre de una persona de acuerdo al identificador (idpersona). |
| function getNomPersona() | Función que devuelve los nombres de las personas que pertenecen a un determinado órgano. |

| Nombre: class cNomOrgano | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| organo | tNomOrgano |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idorgano = '',$denom = '',$ctagasto = '',$ctainvers = '',$ctaingreso = '',$ctafondo = '',$otras = '',$padre = '',$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_organo. |
| function getDatosPadreOrg($val) | Función que devuelve los datos de la tabla nom_organo de acuerdo a la sesión de trabajo. |
| function getDenomOrg($org) | Función que devuelve la denominación (denom) de un órgano en específico. |

| Nombre: class cNomIngreso | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| ingreso | tNomIngreso |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idingreso = '' ,$codigo = '' ,$denom = '' ,$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_ingreso. |
| function getNomIngreso() | Función que devuelve el identificador (idingreso) y la denominación (denom) de |

| | |
|---|---|
| | los ingresos que se usan. |

| Nombre: class cNomEntidad | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| entidad | tNomEntidad |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$identidad = '' ,$codigo = '' ,$denom = '' ,$abrev = '' ,$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_entidad. |
| function getDenomEntidad() | Función que devuelve el identificador (identidad) y la denominación (denom) de las entidades que se usan. |

| Nombre: class cNomCuenta | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| cuenta | tNomCuenta |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idcuenta = '' ,$codigo = '' ,$denom = '' ,$balance = '' ,$naturaleza = '' ,$contabiliza = '' ,$analisis = '' ,$idgrupo = '' ,$padre = '' ,$idfijo = '' ,$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_cuenta. |
| function getDatosCuenta() | Función que devuelve el código de cada cuenta. |
| public function SelectCuenta() | Función que devuelve el identificador (idcuenta) y la denominación (denom) de las cuentas que se usan. |
| function getDenomTipo($codigo) | Función que devuelve la denominación (denom) de una cuenta en dependencia del código (codigo). |
| function getCodigoCta() | Función que devuelve el identificador (idcuenta) y el código (codigo) de las cuentas que usan. |

| Nombre: class cNomGasto | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| gasto | tNomGasto |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idgasto = '' ,$codigo = '' ,$denom = '' ,$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_gasto. |
| function getDenomGasto() | Función que devuelve el identificador (idgasto) y la denominación (denom) de los gastos que están en uso. |

| Nombre: class cNomPresup | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| presup | tNomPresup |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idpresup = '' ,$codigo = '' ,$denom = '' ,$actual = '1' ) | Función que devuelve todos los datos de la tabla nom_presup. |
| function getDenomPresu() | Función que devuelve el identificador (idpresup) y la denominación (denom) de los presupuestos presupuesto que están en uso. |

| Nombre: class cDatAntidipo | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| anticipo | tDatAnticipo |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$idanticipo = '' ,$idorgano = '' ,$idtipooper = '' ,$idctadebe = '' ,$idctahaber = '' ,$idpersona = '' ,$noant = '' ,$fanticipo = '' ,$descripact = '' ,$importe = '' ,$fliquidacion = '' ,$noliqui = '' ) | Función que devuelve todos los datos de la tabla dat_anticipo. |
| function CantMovAnticipo() | Función que devuelve el máximo del número de anticipos. |
| function ExisteAnticipo($idpersona) | Función que devuelve el identificador de un anticipo (idanticipo) en dependencia del identificador de la persona (idpersona). |
| function SiLiquidado($idpersona) | Función que devuelve el máximo del número de liquidaciones en dependencia del identificador de la persona (idepersona). |
| function getDatLiquidar() | Función que devuelve los datos de los anticipos que no estén liquidados. |
| function getNumLiquidacion() | Función que devuelve el máximo del número de las liquidaciones. |
| function getListadoAnticipos() | Función que devuelve los datos de todos los anticipos que han sido liquidados. |

| Nombre: class cDatDetalle | |
|---|---|
| Tipo de clase: entidad | |
| Atributo | Tipo |
| ddetalle | tDatDetalle |
| filtro | String |
| Para cada responsabilidad: | |
| Nombre: | Descripción |
| function __construct() | Función para inicializar los objetos de la clase. |
| function getDatos($array = 0,$iddetalle = '' ,$idefectivo = '' ,$idpagmenor = '' ,$importe = '' ,$idingreso = '' ,$idgasto = '' ,$identidad = '' ,$idgrupo = '' ,$idcuenta = '' ) | Función que devuelve todos los datos de la tabla dat_detalle. |
| function CantMovDetalle($idoperacion) | Función que devuelve máximo del número de movimientos (nomov) de un modelo en dependencia del identificador de la operación (idoperacion). |
| function getMovEfectivos($idoperacion) | Función que devuelve todos los datos correspondientes a los detalles de un modelo. |
| function DatosImprimir($idoperacion) | Función que devuelve los datos a Imprimir de un modelo. |

**Diseño de la Base de Datos**

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda, el diseño la misma. El objetivo principal del diseño de bases de datos es generar tablas que modelan los registros en los que se guardará información. Si las tablas no son definidas apropiadamente, puede resultar un problema el proceso de ejecutar consultas a la base de datos para tratar de obtener algún tipo de información. No importa si una base de datos tiene sólo 20 registros, o algunos cuantos miles, es importante asegurarse de que la misma esté correctamente diseñada para que tenga eficiencia y que se pueda seguir utilizando por largo tiempo. Es importante que esta información se almacene sin redundancia para que se pueda tener una recuperación rápida y eficiente de los datos.

A continuación se representa el diseño propuesto de la base de datos que será utilizada para la implementación del sistema propuesto.

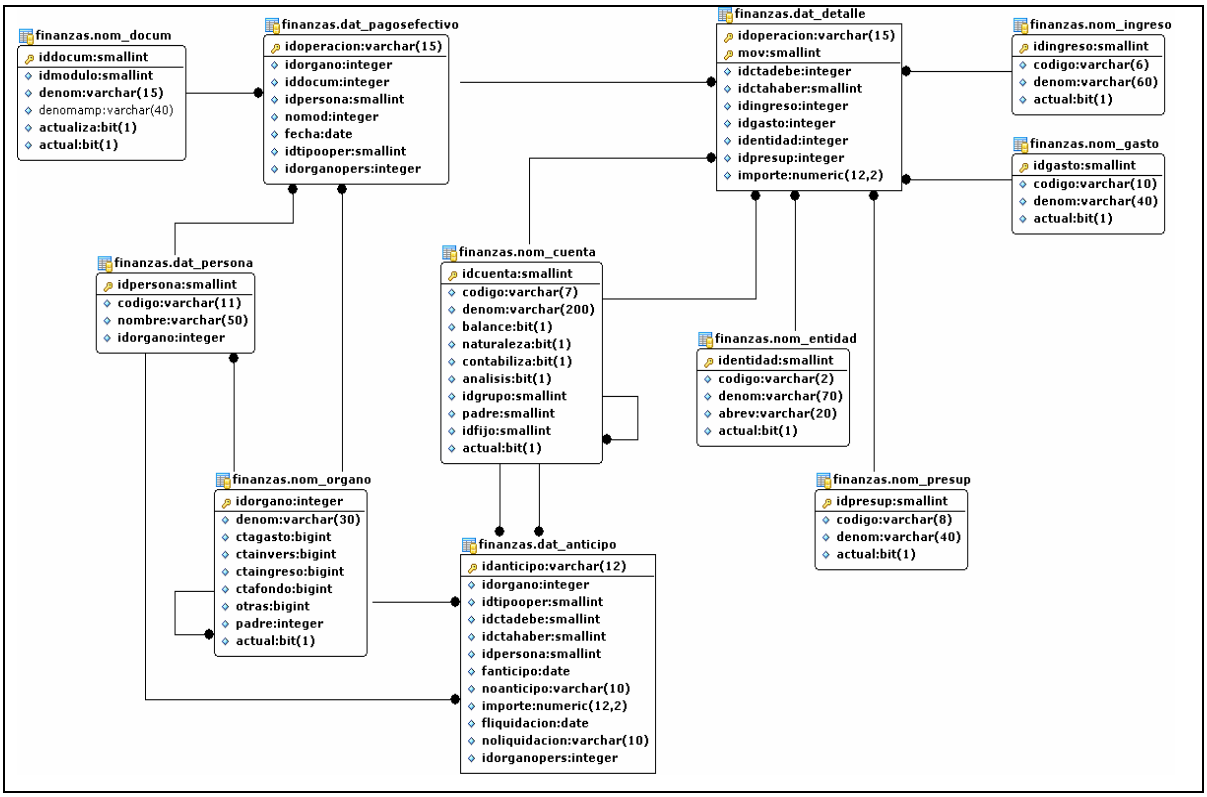



**Fig. 23 Diagrama Entidad Relación de la base de datos.**

**Descripción de las tablas.**

| **Nombre:** nom_docum | | |
|---|---|---|
| **Descripción:** Nomenclador de documentos oficiales del sistema | | |
| **Atributo** | **Tipo** | **Descripción** |
| iddocum | smallint | identificador del documento |
| idmodulo | smallint | viene de nom_modulo |
| denom | varchar(15) | denominación del documento por siglas |
| denomamp | Varchar(40) | denominación ampliada del nombre del documento |
| actualiza | Bit(1) | 1 es para los documentos que se actualizan, 0 para los que solo se recuperan |
| actual | Bit(1) | indicador de uso |

| **Nombre:** dat_pagosefectivo | | |
|---|---|---|
| **Descripción:** Corresponde al modulo de Caja, se guardan todos los pagos y los efectivos emitidos en el año para el órgano financiero. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idoperacion | varchar(15) | identificador del efectivo o vale se forma concatenando los valores: año(4)+idorgano(4)+iddocum(3)+nomod(4) ej.200700030010001 |
| idorgano | integer | viene de nom_organo |
| iddocum | integer | viene de nom_docum |
| idpersona | smallint | viene de dat_persona |
| nomod | integer | numero del modelo ya sea un vale o un efectivo, se inicializa en 1 por año y se incrementa de 1 en 1, esta función se realiza en php |
| fecha | date | fecha en que se realiza la operación por el cajero |
| idtipooper | smallint | |
| estado | bit(1) | campo para controlar que el efectivo este activo o anulado(0-anulado y 1-activo) |

| **Nombre:** dat_persona | | |
|---|---|---|
| **Descripción:** Listado de las personas que tienen retención al banco. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idpersona | smallint | identificador de persona |
| codigo | varchar(11) | numero del carné de identidad |
| nombre | varchar(50) | nombre propio de las personas |
| idorgano | integer | viene de nom_organo |
| idmunic | smallint | |
| idprovincia | smallint | |

| **Nombre:** nom_organo | | |
|---|---|---|
| **Descripción:** Nomenclador de Órganos Financieros. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idorgano | integer | identificador del órgano |
| denom | varchar(30) | denominación del órgano |
| ctagasto | bigint | cuenta bancaria para gastos |
| ctainvers | bigint | cuenta bancaria para inversiones |
| ctaingreso | bigint | cuenta bancaria para ingresos |

| | | |
|---|---|---|
| ctafondo | bigint | cuenta bancaria para fondos de operaciones |
| otras | bigint | otras cuentas bancarias |
| padre | integer | determina si el órgano es un consumidor o administrador |
| actual | bit(1) | Indicador de uso |

| **Nombre:** nom_cuenta | | |
|---|---|---|
| **Descripción:** Nomenclador de cuentas contables | | |
| **Atributo** | **Tipo** | **Descripción** |
| idcuenta | smallint | identificador de cuenta(autonumérico) |
| codigo | varchar(7) | código de la cuenta |
| denom | varchar(200) | denominación de la cuenta |
| balance | bit(1) | para saber si es una cuenta de patrimonio (1) o de orden(0) |
| naturaleza | bit(1) | si es una cuenta deudora(0) o acreedora(1) |
| contabiliza | bit(1) | si por esa cuenta se contabiliza(1), no se contabiliza por ella(0) |
| analisis | bit(1) | Si esa cuenta lleva análisis (1)(grupo, entidad, gasto, ingreso, costo, retenciones ,etc., no lleva análisis(0) |
| idgrupo | smallint | viene de nom_grupo |
| padre | smallint | código del padre de la cuenta |
| idfijo | smallint | viene de nom_fijo |
| actual | bit(1) | indicador de uso |

| **Nombre:** dat_anticipo | | |
|---|---|---|
| **Descripción:** Corresponde al modulo de Caja, se guardan los datos correspondientes a los anticipos y sus liquidaciones. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idanticipo | varchar(12) | Identificador del anticipo, se forma concatenando los valores: año+idorgano+noant ej. 200700030001 |
| idorgano | integer | viene de nom_organo |
| idtipooper | smallint | viene de nom_tipooper |
| idctadebe | smallint | viene de nom_cuenta |
| idctahaber | smallint | viene de nom_cuenta |
| idpersona | smallint | viene de dat_persona |
| noant | integer | Se incremente en de 1 en 1 a través de todo el año, función que se implementa en php |
| fanticipo | date | fecha en que se entrega el anticipo a la persona |
| descripact | varchar(50) | Descripción de la actividad que se realiza. |
| importeanti | numeric(12,2) | Importe del anticipo. |

| | | |
|---|---|---|
| fliquidacion | date | Fecha estimada de liquidación del anticipo. |
| frealliqui | date | Fecha real en la que se liquida el anticipo. |
| noliqui | integer | Numero de la liquidación se incrementa de 1 en 1 durante todo el año |
| importeliqui | numeric(12,2) | |
| estado | bit(1) | campos que controla el estado de un anticipo 0-anulado y 1-activo |

| **Nombre:** dat_detalle | | |
|---|---|---|
| **Descripción:** Corresponde al modulo de Caja, se guardan todas las operaciones correspondientes a los efectivos o los vales. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idoperacion | varchar(15) | Viene de dat_pagosefectivo |
| nomov | smallint | contador para saber la cantidad de movimientos asociados a una operación ya sea un efectivo o un vale |
| idctadebe | integer | viene de nom_cuenta |
| idctahaber | smallint | viene de nom_cuenta |
| idingreso | integer | viene de nom_ingreso |
| idgasto | integer | viene de nom_gasto |
| identidad | integer | viene de nom_entidad |
| idpresup | integer | viene de nom_presup |
| importe | numeric(12,2) | importe asignado a la persona |

| **Nombre:** nom_entidad | | |
|---|---|---|
| **Descripción:** Nomenclador de Entidad o Especialidad. | | |
| **Atributo** | **Tipo** | **Descripción** |
| identidad | smallint | identificador de entidad |
| codigo | varchar(2) | código de la entidad o especialidad |
| denom | varchar(70) | denominación de la entidad o especialidad |
| abrev | varchar(20) | abreviatura de la entidad |
| actual | bit(1) | indicador de uso |

| **Nombre:** nom_presup | | |
|---|---|---|
| **Descripción:** Nomenclador de Grupos Presupuestarios. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idpresup | smallint | identificador del grupo presupuestario |
| codigo | varchar(8) | código del presupuesto solo números y punto(111111.2) |
| denom | varchar(40) | denominación del grupo presupuestario |
| actual | bit(1) | indicador de uso |

| **Nombre:** nom_ingreso | | |
|---|---|---|
| **Descripción:** Nomenclador de ingresos al presupuesto del estado. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idingreso | smallint | identificador de ingreso |
| codigo | varchar(6) | código del ingreso (solo números) |
| denom | varchar(60) | denominación del ingreso |
| actual | bit(1) | indicador de uso |

| **Nombre:** nom_gasto | | |
|---|---|---|
| **Descripción:** Nomenclador de Gastos en las FAR. | | |
| **Atributo** | **Tipo** | **Descripción** |
| idgasto | smallint | identificador del gasto o partida |
| codigo | varchar(10) | código del gasto |
| denom | varchar(40) | denominación del gasto |
| actual | bit(1) | indicador de uso |

**Mecanismos de diseño**

Durante la fase de inicio del software, se pueden comenzar a realizar actividades específicas del flujo de trabajo de análisis y diseño, flujo de trabajo que, como se ha visto, cobra su mayor importancia en la fase de elaboración, en la cual uno de los principales objetivos, además de obtener en términos de diagramas de clases y diagramas de interacción la representación de la implementación de cada una de las funcionalidades previstas, es el establecimiento de la línea base de la arquitectura, para lo cual se recomienda emplear patrones que permiten construir mecanismos de diseño que contribuyan a diseñar aplicaciones más reusables, menos complejas, es decir, construir mejores soluciones.

**Mecanismo de Diseño de Acceso a Datos**

Como parte de la tarea de diseñar la base de datos contenedora de la información necesaria a manipular, en el estudio de cada entidad y la forma de acceder a las mismas para la obtención de la información solicitada, fueron identificadas un grupo de funcionalidades comunes para todos los casos de uso propuestos, así como un conjunto de objetos que se involucran siempre en este tipo de operación. En aras de lograr una mayor claridad en los diagramas logrando así un mayor rendimiento en el presente flujo de trabajo. Fue diseñado el siguiente mecanismo de acceso a datos en el cual se observan un conjunto de clases que se comunican e interactúan facilitando el acceso y manipulación de los datos.

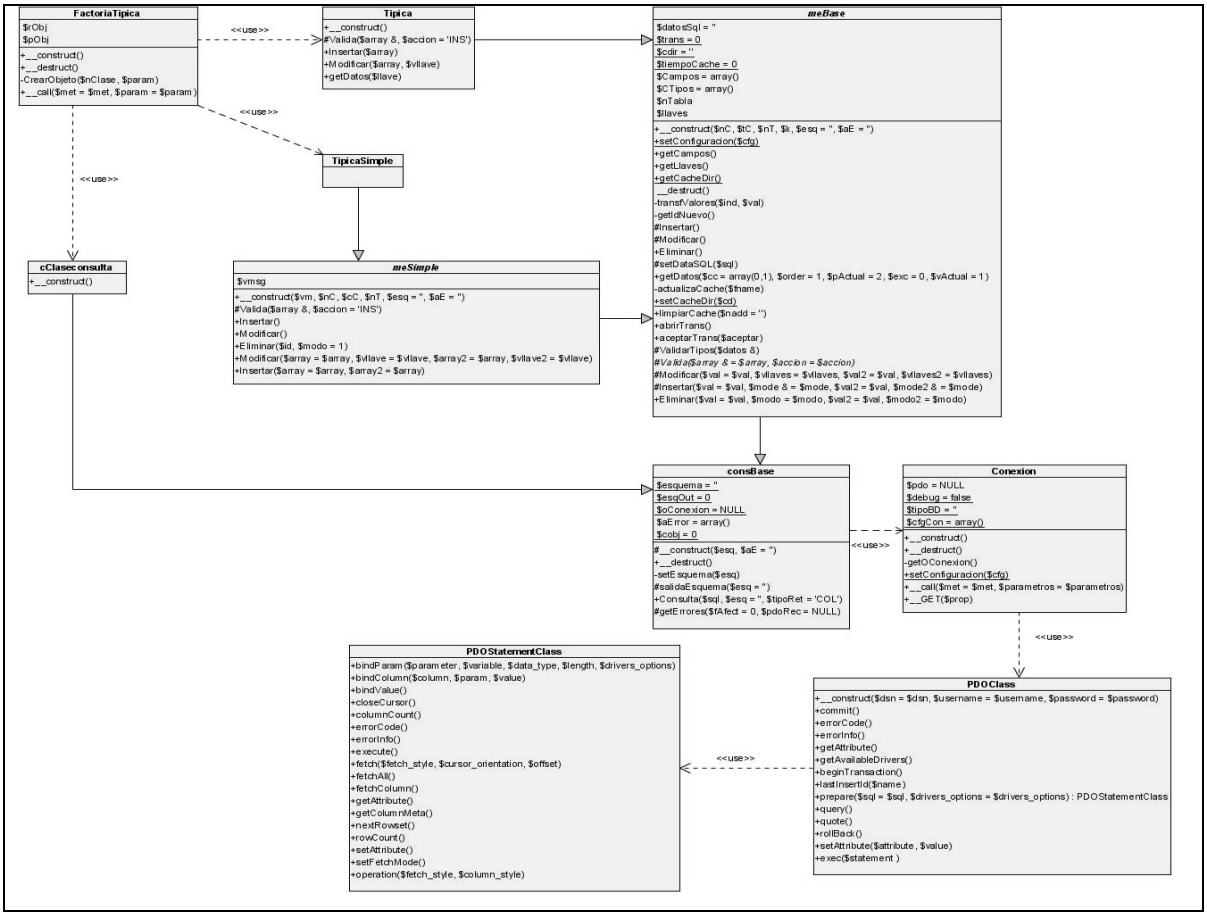



**Fig. 24 Mecanismo de Acceso a Datos.**

A continuación se hace una breve descripción de las clases que conforman el mencionado modelo de acceso a datos.

Clase Factoría Típica:

Esta clase es el resultado de aplicar patrones de diseño, los cual es muy ventajoso pues la funcionalidad de los mismos consiste en describir un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software, y entregar una buena solución ya probada. Esto ayuda a lograr un diseño de forma correcta en menos tiempo, a construir problemas reutilizables y extensibles, y facilita la documentación.

En el caso de Factoría Típica se ha utilizado un patrón de diseño de tipo creador llamado Factoría. Los patrones de creación son las soluciones aceptadas como "buenas" a los problemas de creación de

instancias de objetos. Los programas orientados a objetos crean decenas, cientos o incluso miles de instancias de objetos, es por ello, que esta no es una tarea que se puede realizar a la ligera.

El patrón factoría es uno de los varios patrones creadores existentes. La idea que se esconde detrás de este patrón es la de centralizar el sitio donde se crean los objetos, normalmente donde se crean objetos de una misma "familia", sin dar una definición clara de lo que el software puede entender como familia, como podría ser componentes visuales, componentes de la lógica del negocio, u objetos concurrentes en el tiempo.

La clase factoría devuelve una instancia de un objeto según los datos que se le pasan como parámetros. Para que la creación centralizada de objetos sea lo más "útil y eficaz" posible, es de esperar que todos los objetos creados desciendan de la misma clase o implementen el mismo interface (es decir, hagan una operación similar pero de distintas formas), haciéndose posible así usarlos todos de la misma manera, con los mismos métodos (gracias al polimorfismo), sin importar que clase concreta se esté tratando en cada momento.

La clase Factoría típica está concebida para instanciar todas las clases típicas del sistema para utilizarlas cuando se necesite.

Las clases típicas son clases que surgen también de la aplicación de patrones de arquitectura, en este caso se trata del patrón Table Data Gateway que se encarga de realizar una clase para instanciar cada tabla de la base de datos.

Esta clase Factoría típica implementa un método llamado __call($met, $param) el cual permite de forma transparente la interacción con el resto de las clases del modelo de persistencias. Dicho método implementa de forma explicita tres métodos base:

Crear: Permite la creación de instancias de clases. Se emplea solo cuando se desea ejecutar una única operación de las que implementa la clase instanciada (siempre se destruirá automáticamente la instancia después de la ejecución del método) y además su constructor debe recibir parámetros.

pCrear: Permite la creación de instancias de clases. Se emplea cuando se desea ejecutar varias operaciones de las que implementa la clase instanciada. La destrucción del objeto creado será responsabilidad del programador, aunque al concluir la ejecución del script esta se destruye.

Destruir: Permite destruir una instancia creada.

Dicho método implementa además de forma implícita el resto de las operaciones que están contenidas dentro de las clases del modelo de acceso a datos, las más empleadas son por ejemplo:

Insertar: Para insertar datos en la entidad dada.
Modificar: Para modificar datos en la entidad dada.
Eliminar: Para eliminar datos en la entidad dada.
getDatos: Para consultar datos en las entidades de nomenclaturas.
Clase meBase: Es una clase abstracta, base para el resto de las que implementen funcionalidades para el trabajo con las entidades del sistema a implementar. Dicha clase Implementa las operaciones básicas que pudieran realizarse a una entidad (insert, delete, update y consultas) además de encapsular lo relacionado con la conexión al gestor de base de datos.
Clase meSimple: Clase abstracta que hereda de meBase, base para la implementación de las típicas que responderán a los nomencladores simples del modelo de persistencia dado. Redefine las operaciones básicas con la funcionalidad de Validación dada. Define las operaciones básicas que pudieran realizarse a una entidad (insert, delete, update) para los nomencladores simples.
Clase Conexión: Esta clase, se encarga de establecer la conexión a la base de datos a través de objetos nativos brindados por el entorno de desarrollo PHP.
Clase consBase: Esta clase es la base en toda la jerarquía de Acceso a Datos y es empleada para aportar contenido dinámico a las plantillas. Encapsula el objeto creado para la conexión.

**Mecanismo de seguridad**

De manera general, cuando se utiliza la Web para el modelado de sistemas se debe tener en cuenta que el acceso a los mismos debe ser de forma controlada, es decir que cada usuario tenga acceso sólo a la información que le concierne. Incorporando esta idea al contexto del presente trabajo, se debe puntualizar que, el módulo que se está implementando (como se ha mencionado en capítulos anteriores), es una parte de un sistema de contabilidad financiera, que pertenece a su vez a un software ERP (ERP-FAR), el cual tiene incluido además un módulo para gestionar la seguridad. El mecanismo que implementa dicho módulo de seguridad parte de la idea que a través de una página cliente el usuario accede al sistema (al ERP-FAR), una vez autenticado sus datos son enviados a través de una página servidora a una clase llamada control acceso, clase interfaz que verifica la validez de los datos a través del paquete de seguridad, una vez verificados, si son válidos dicha clase devuelve el rol que le pertenece al usuario autenticado, atributo a partir del cual se obtienen los privilegios que tiene el mismo en el sistema al que ha accedido, controlando de esta forma que cada usuario interactúe con el sistema de forma personalizada, de acuerdo con su responsabilidad.

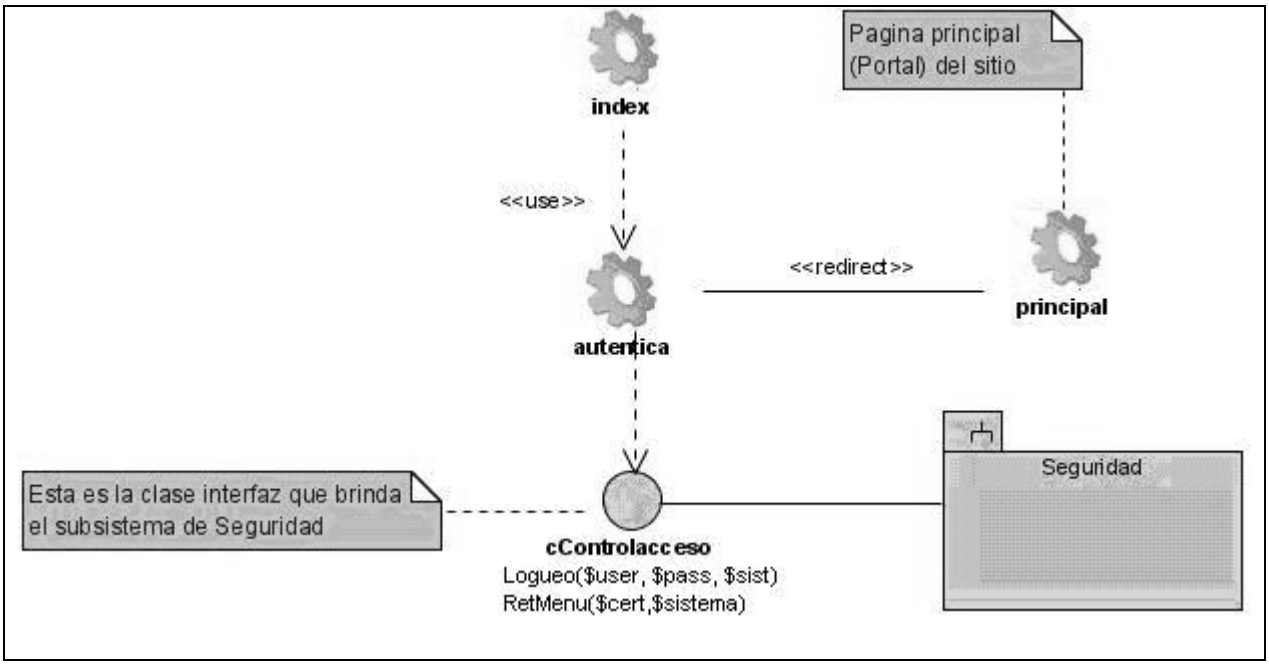



**Fig. 25 Mecanismo de Seguridad.**

**Concepción de la ayuda**

Con el propósito de hacer más fácil para el usuario el trabajo con el software final, se confeccionará una ayuda para que el mismo se familiarice con el uso de la aplicación antes de comenzar a trabajar con ella y la consulte cada vez que lo considere necesario, asegurando con esto que el uso del producto sea lo más eficiente posible.

**Tratamiento de errores**

Se considera que el tratamiento de errores es un elemento fundamental para el correcto funcionamiento del sistema propuesto, pues el usuario, en este caso el cajero, estará en constante intercambio con el software, es decir, ejecutará operaciones de selección, inserción y modificación de datos, lo cual significa, que debe existir un mecanismo que no permita hacer este tipo de acciones de forma incorrecta, para lograr esto, cada vez que el cajero intente ejecutar alguna operación errónea, el sistema no se lo permitirá y le mostrará a su vez un mensaje de error notificándole la naturaleza del mismo, de tal forma que le será posible percatarse de la equivocación y rectificarla ejecutando nuevamente la acción de forma correcta.

**Conclusiones**

Se considera que con la culminación de este capítulo, se está en condiciones de pasar a la fase de construcción del software, principalmente al flujo de trabajo de implementación, pues elementos como el modelo de diseño, diagrama de la base de datos, mecanismos de diseño, entre otros se consideran suficientes para dar un gran paso adelante, un paso hacia la implementación del sistema propuesto.

**Capitulo 4. Implementación y Prueba**.

**Introducción**

En el presente capítulo se pretende comenzar con la fase de construcción, en la cual toman el papel protagónico los flujos de trabajo implementación y prueba, obteniendo el modelo de implementación que incluye componentes (representan el código fuente) y la correspondencia de las clases con los mismos, es decir denotan la implementación actual del sistema en términos de componentes y subsistemas de implementación; y modelos de prueba, los cuales describen los casos de prueba que verifican la funcionalidad de cada caso de uso. En esta fase se obtiene la capacidad operativa inicial del producto.

**Modelo de Implementación**

El modelo de implementación describe como los elementos del modelo de diseño, se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc. Este modelo describe además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados, y cómo dependen los componentes unos de otros. Dicho modelo incluye además el diagrama de despliegue, donde queda definida la ubicación física del software implementado.

**Diagrama de despliegue**:

En el mismo se sitúa el software en el hardware que lo contiene. Cada Hardware se representa como un nodo. Un nodo se representa como un cubo, es un elemento donde se ejecutan los componentes, representan el despliegue físico de los mismos.

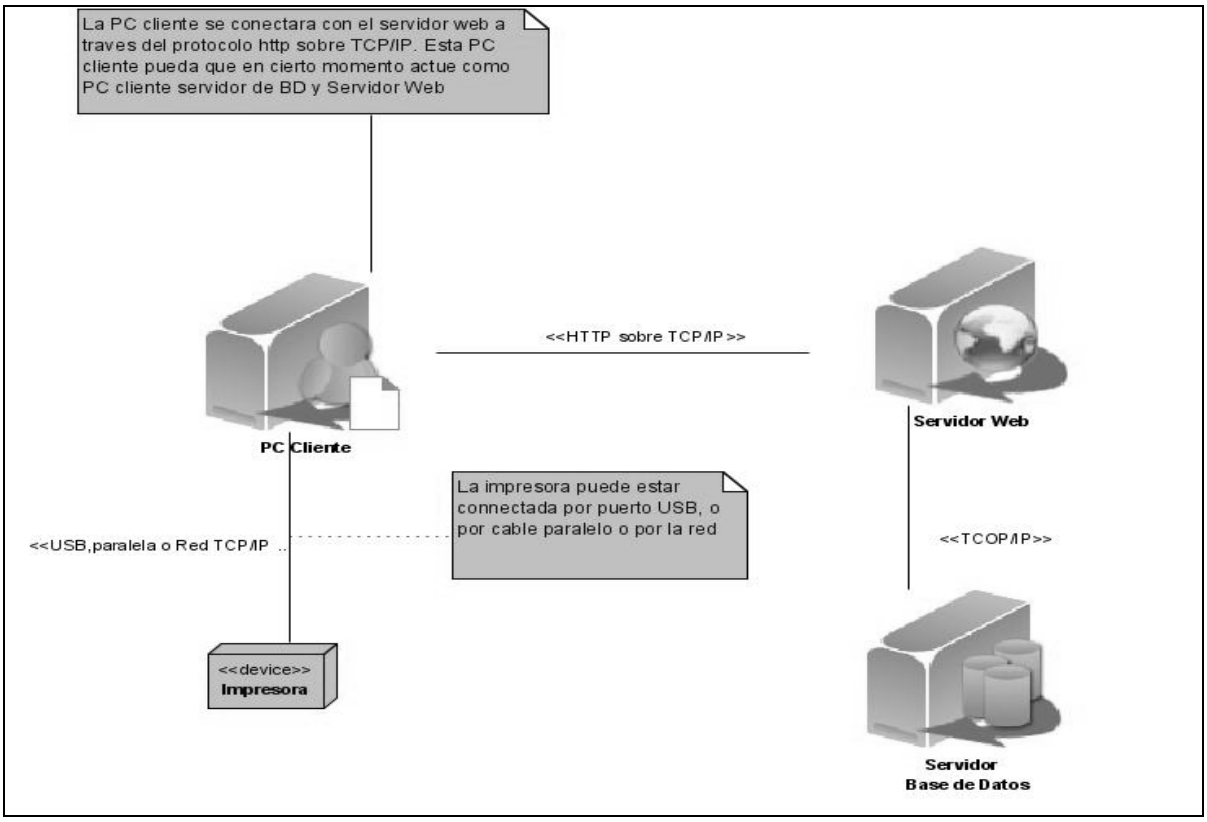



**Fig. 26 Diagrama de despliegue.**

**Diagrama de componentes**.

Es un diagrama que se utiliza para modelar la vista estática del sistema, para ello, puede mostrar un conjunto de elementos tales como componentes, subsistemas de implementación e interfaces. Los componentes pueden ser ejecutables, código fuente, librerías, bases de datos físicas, documentos, etc. y pueden implementar varios elementos. Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables, pueden estar formados por componentes, interfaces y otros subsistemas. Las interfaces son utilizadas en el modelo de implementación para especificar las operaciones implementadas por componentes y subsistemas de implementación.

A continuación se muestra el diagrama de componentes general del software, dicho diagrama está formado por subsistemas de implementación e interfaces. Cada subsistema tiene su propio diagrama de componentes, para ver cada uno de estos diagramas puede remitirse a los anexos.

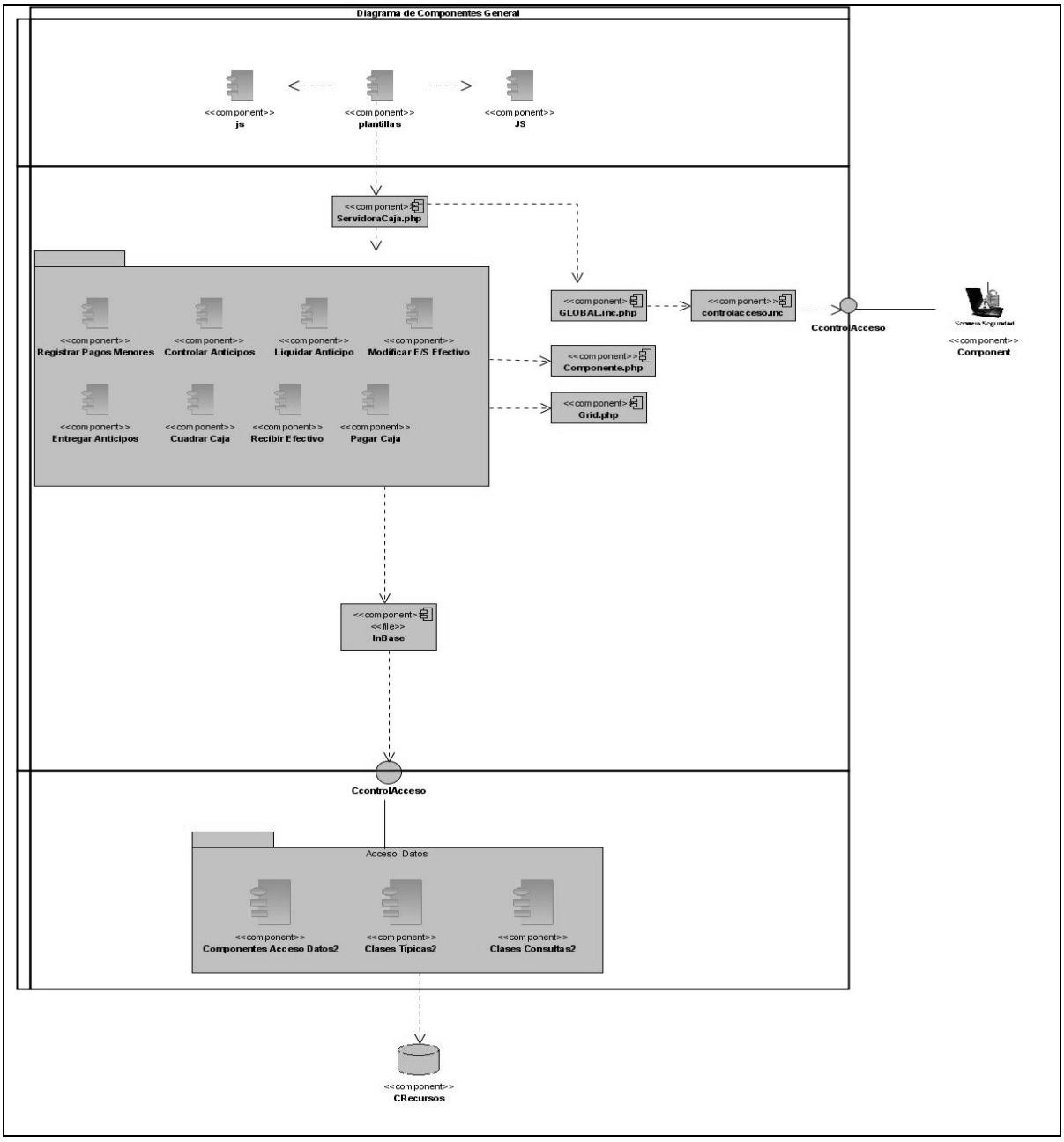



**Fig. 27 Diagrama de componentes.**

**Modelo de prueba**

Una de las últimas fases del ciclo de vida del desarrollo de un software es el flujo de trabajo de pruebas, al cual es necesario dedicarle un importante tiempo de esfuerzo y desarrollo pues dicha actividad está encaminada a encontrar errores en el software.

La prueba es el proceso de ejercitar un programa bajo condiciones específicas cuyos resultados deben ser registrados y luego analizados. Es posible hacerlas a distintos niveles, tales como unidad, integración, sistema y aceptación.

Las pruebas de caja negra son pruebas de sistema, este tipo de prueba es aceptable en la presente etapa de construcción. Las mismas están encaminadas a encontrar errores previos a la entrega del usuario final. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente los requisitos funcionales del programa, para esto se diseñan casos de pruebas.

Un caso de prueba especifica una forma de examinar el sistema, incluyendo la entrada y salida con la que se ha de probar y las condiciones bajo las que ha de hacerse dicha acción.

A continuación se desarrollan las pruebas de caja negra al software final, para esto fueron definidos diferentes casos de prueba por cada caso de uso del sistema chequeando de esta forma los requisitos funcionales que debe tener el software.

**Casos de Prueba**

Nombre del caso de uso: Recibir Efectivo.

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana para insertar los datos fecha, unidad militar y nombre de la persona que entrega el efectivo por caja. | |
| Fecha “26/05/2007”<br>Unidad Militar “LOL”<br>Nombre de la persona “Yasser Linares”<br>Pulsar botón Aceptar. | El sistema muestra un mensaje confirmando la acción que se va a ejecutar con los datos entrados.<br><br>El sistema muestra una ventana para que el cajero inserte los detalles del recibo de efectivo. | Debe existir conexión con la base de datos. |
| Debe “031.1”<br>Haber “S/D” | El sistema muestra en un grid los datos insertados | Debe existir conexión con la base de datos. |

| | | |
|---|---|---|
| Ingreso “1-Multas de código penal”<br>Importe “45”<br>Pulsar botón Aceptar. | correctamente. | Se deben entrar todos los datos excepto en el caso del Debe y el Haber, pues solo se puede insertar uno los dos. |
| Debe “S/D”<br>Haber “041.2”<br>Ingreso “11-ddd”<br>Importe “”<br>Pulsar botón Aceptar | El sistema muestra un mensaje de error indicando que el campo importe no puede estar vacío. | El campo importe no puede estar vacío. |

Nombre del caso de uso: Pagar por Caja.

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana para insertar los datos fecha, unidad militar y nombre de la persona que recibe el pago por caja. | |
| Fecha “25/05/2007”<br>Unidad Militar<br>“ ”<br>Nombre de la persona<br>“Yasser Linares”<br>Pulsar botón Aceptar. | El sistema muestra un mensaje de error informando que debe insertar la unidad militar a la que pertenece la persona que recibe el pago por caja. | Se deben entrar todos los datos. |
| Fecha “24/05/2007”<br>Unidad Militar<br>“Aurot ”<br>Nombre de la persona<br>“Yosveni Escalona”<br>Pulsar botón Aceptar. | El sistema muestra un mensaje confirmando la acción que se va a ejecutar con los datos entrados.<br><br>El sistema muestra una ventana para que el cajero inserte los detalles del recibo de efectivo. | Debe existir conexión con la base de datos. |
| Debe “031.1”<br>Haber “S/D”<br>Grupo “a”<br>Especialidad “r”<br>Partida “c”<br>Importe “50”<br>Pulsar botón Aceptar. | El sistema muestra en un grid los datos insertados correctamente. | Debe existir conexión con la base de datos.<br>Se deben entrar todos los datos excepto en el caso del Debe y el Haber, pues solo se puede insertar uno de los dos. |
| Debe “S/D” | El sistema muestra un mensaje | El campo importe tiene que ser |

| | | |
|---|---|---|
| Haber “041.2”<br>Grupo “11-ddd”<br>Especialidad “b”<br>Partida “f”<br>Importe “+”<br>Pulsar botón Aceptar | de error indicando que el campo Importe no es un dato válido. | numérico. |

Nombre del caso de uso: Modificar E/S Efectivo.

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana para buscar el modelo a modificar por varios criterios (fecha, nombre persona, órgano). | |
| Fecha “25/05/2007”<br>Pulsar botón Buscar. | El sistema muestra un grid sin datos. | Debe existir conexión con la base de datos.<br>No existe en la tabla dat_pagosefectivo un modelo (recibo de efectivo o vale para pagos menores) con esa fecha. |
| Nombre Persona “Guillermo Mora” | El sistema muestra un grid con los modelos (recibo de efectivo o vale para pagos menores) a nombre de Guillermo Mora que existen en la base de datos. | Debe existir conexión con la base de datos.<br>Debe existir en la tabla dat_pagosefectivo modelos con ese nombre. |
| Fecha “0/05/2007”<br>Nombre Persona “ Sibel Viera”<br>UM “LOL” | El sistema muestra en un grid el modelo que cumple con los datos indicados. | Debe existir conexión con la base de datos.<br>Debe existir en la tabla dat_pagosefectivo el modelo que cumple con los datos indicados. |

Nombre del caso de uso: Entregar Anticipo.

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana para insertar los datos de un anticipo. | |
| Fecha Entrega “25/05/2007”<br>Fecha Liquidación “31/05/2007” | El sistema muestra un mensaje de error informando que la persona tiene un anticipo pendiente a liquidar. | Debe existir conexión con la base de datos.<br>Debe existir en la tabla dat_anticipo la persona con un |

| | | |
|---|---|---|
| Órgano Financiero “Aurot”<br>Nombre de la persona “Gretter Mora Viera”<br>Debe “041.2”<br>Haber “S/D”<br>Labor a realizar “dfjdhfhjh”<br>Importe “40”<br>Pulsar botón Aceptar. | | anticipo pendiente a liquidar. |
| Fecha Entrega “23/05/2007”<br>Fecha Liquidación “28/05/2007”<br>Órgano Financiero “LOL”<br>Nombre de la persona “Yosveni Escalona”<br>Debe “042.1”<br>Haber “S/D”<br>Labor a realizar “dufhjdnf”<br>Importe “100”<br>Pulsar botón Aceptar. | El sistema muestra un mensaje informando que el anticipo ha sido registrado correctamente. | Debe existir conexión con la base de datos.<br>Esta persona no puede tener un anticipo pendiente por liquidar. |

Nombre del caso de uso: Liquidar Anticipo.

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana para realizar la liquidación de un anticipo. | |
| Marcar anticipo a liquidar en el grid.<br>Pulsar botón Liquidar. | El sistema muestra un mensaje informando que el anticipo ha sido liquidado.<br>Muestra el grid sin el anticipo liquidado.<br>Liquida el anticipo. | Debe existir conexión con la base de datos.<br>Debe existir el anticipo pendiente a liquidar. |

Nombre del caso de uso: Registrar Pagos menores

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana con el registro de pagos menores. | Debe existir conexión con la base de datos. |

Nombre del caso de uso: Registrar Pagos menores

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana con el registro de pagos menores. | Debe existir conexión con la base de datos. |

Nombre del caso de uso: Controlar anticipos

| Entrada | Resultados | Condiciones |
|---|---|---|
| | El sistema muestra una ventana con el registro de control de anticipos. | Debe existir conexión con la base de datos. |

**Conclusiones.**

Con la realización de este capítulo, se ha obtenido la funcionalidad del sistema propuesto, así como el diagrama de despliegue conociendo la situación física del software realizado. Se hicieron además pruebas de sistema garantizando el correcto cumplimiento de los requerimientos funcionales que se debían cumplir.

**Conclusiones**

Con la realización de este trabajo, se obtuvo un software funcional con las características apropiadas para ser desplegado en las unidades de las FAR e integrado al sistema de contabilidad financiera que forma parte del ERP-FAR. Dicho software cumple con los requerimientos funcionales y no funcionales previstos en la fase de inicio donde se estableció la viabilidad del proyecto. Con la realización del mismo, se puede dar solución a muchos de los problemas existentes hoy con las condiciones de trabajo presentes en las unidades de la mencionada institución, pues se tendrá ahora un mejor y más cómodo control de dichos procesos que implican una gran cantidad de documentos y tiempo de registro. Con la aplicación propuesta se realizan los principales procesos de caja de forma rápida y eficiente constituyendo la misma un importante apoyo para el correcto desempeño y calidad del trabajo de los cajeros en las FAR cumpliendo así con los objetivos propuestos.

**Recomendaciones.**

- ✓ Se recomienda desplegar el software en las condiciones especificadas.
- ✓ Se recomienda realizar el Caso de uso Registrar Pagos Especiales.
- ✓ Se recomienda realizar el Caso de uso Emitir Notificaciones.
- ✓ Se recomienda realizar una segunda iteración de desarrollo para perfeccionar el software construido.

**Bibliografía.**

**Citada.**

1. TOUZETT, E. Implementación del sistema de costos ABC en la empresa Kia Import Perú, 2006]. Disponible en: http://www.monografias.com/trabajos45/sistema-costos-kia/sistema-costos-kia2.shtml
2. ¿Qué es la contabilidad financiera? , 2006]. Disponible en: http://www.gestiopolis.com/recursos/experto/catsexp/pagans/fin/44/contafinan.htm
3. SAAVEDRA, F. Sistemas ERP en las PYMES. Claves para implementar un ERP de manera exitosa., 2006]. Disponible en: http://www.diariopyme.cl/newtenberg/1805/article-70424.html
4. TAMARGO, L. C. and J. T. SANABRIA. La informática en función de la actividad contable., 2007]. Disponible en: http://www.bc.gov.cu/anteriores/RevistaBCC/2004/AbrJun2004/informatica.htm
5. Sistema automatizado de la gestión municipal del área de la administración financiera. 2007]. Disponible en: http://cepadem.org

**Consultada.**

1. Antecedentes históricos de la contabilidad. 2006]. Disponible en: http://html.rincondelvago.com/historia-de-la-contabilidad_4.html
2. Primeras civilizaciones. 2006]. Disponible en: http://educahistorialog.educalogs.cl/2007/04/14/primeras-civilizaciones/
3. Software ERP. 2006]. Disponible en: http://www.infogestionsa.com.ar/plataforma/sistema-erp.htm
4. ¿Qué significa "Internet"? Definición de Internet. 2006]. Disponible en: http://www.masadelante.com/faq-internet.htm
5. World Wide Web. 2006]. Disponible en: http://www.learnthenet.com/spanish/glossary/www.htm
6. ADELL, J. and C. BELLVER. La Internet como telaraña: el World-Wide Web, 2006]. Disponible en: http://www.uv.es/~biblios/mei3/Web022.html
7. Principales definiciones de los términos más usados en Internet. 2006]. Disponible en: http://www.informaticamilenium.com.mx/paginas/espanol/sitioweb.htm
8. ¿Qué es JavaScript? Definición de Javascript. 2006]. Disponible en: http://www.masadelante.com/faq-javascript.htm
9. Estrategias de Desarrollo. 2006]. Disponible en: http://www.aberic.com/home.php?menu=26
10. SURASKI, Z. ¿Qué es el PHP?, 2006]. Disponible en: http://www.cv-team.com/zend.php
11. Glosario. PHP. 2006]. Disponible en: http://www.xmundo.net/glosario_p.html

12. SANCHEZ, M. A. M. Metodologías De Desarrollo De Software, 2004. [2007]. Disponible en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html
13. Metodología de Desarrollo de Software (MDS). 2007]. Disponible en: http://www.reynox.com.ar/sistemas/metodologia.php
14. QUEZADA, C. F. L. Desde el análisis hasta la solución 2007]. Disponible en: http://www.mmug.cl/articulos.php?id=292&tod=1
15. Bases de Datos Heterogéneas. 2007]. Disponible en: http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones/1/BDHeterogeneas.pdf
16. SQL. 2007]. Disponible en: http://es.geocities.com/lenguajes_recuperacion/sql.html
17. Servidores de base de datos usados en Software Libre. 2006. [2007]. Disponible en: http://www.cidsol.org/downloads.php?cat_id=2&download_id=3
18. SÁNCHEZ, C. S. Blog de un ingeniero, 2007]. Disponible en: http://carlos-serrano-sanchez.blogspot.com/2006/11/capitulo-7-mysql.html
19. Ventajas de PostgreSQL. 2007]. Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html
20. VIZCAÍNO, A.; F. Ó. GARCÍA, et al. Prácticas Ingeniería del Software 3º. Una Herramienta CASE para ADOO: Visual Paradigm, 2007]. Disponible en: http://alarcos.infcr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf

**Glosario**

- ✓ Red: Dos o más ordenadores conectados entre sí de manera que puedan compartir recursos
- ✓ TCP/IP: Siglas de Transmission Control Protocol/Internet Protocol, el lenguaje que rige todas las comunicaciones entre todos los ordenadores en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes. También tiene una función de verificación de errores para asegurarse que los paquetes llegan a su destino final en el orden apropiado.
- ✓ Navegador: Un navegador es un programa software que permite ver e interactuar con varios tipos de recursos de Internet disponibles en el World Wide Web.
- ✓ Interfaz de usuario: Es la forma en que los usuarios pueden comunicarse con una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.
- ✓ Lenguaje de programación: Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.
- ✓ Arquitectura Cliente/Servidor: La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.
- ✓ Cliente ligero: (Thin client) es una computadora (cliente) en una arquitectura de red cliente-servidor que tiene muy poca o ninguna lógica del programa, por lo tanto depende principalmente del servidor central para las tareas de procesamiento. Lenguajes del lado del servidor: Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP.
- ✓ Lenguajes del lado del cliente: Son aquellos que pueden ser directamente asimilados por el navegador y no necesitan un pretratamiento. Son totalmente independientes del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio.

- ✓ ASP: Active Server Pages (ASP) es una tecnología del lado servidor para páginas Web generadas dinámicamente. El tipo de servidores que emplean este lenguaje son aquellos que funcionan con sistema operativo de la familia de Windows NT. También se pueden visualizar páginas ASP sobre Windows 95/98
- ✓ JSP: Es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es una tecnología orientada a crear páginas Web con programación en Java.
- ✓ Perl: Lenguaje Práctico para la Extracción e Informe, es un lenguaje de programación imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas.
- ✓ Python: Lenguaje de programación de código abierto que permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python).
- ✓ HTML: El HTML, acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.
- ✓ VBScript: VBScript (abreviatura de Visual Basic Script Edition) es un lenguaje de programación cuya sintaxis refleja su origen como variación del lenguaje de programación Visual Basic.
- ✓ MySQL: Es un sistema de gestión de base de datos, multihilo y multiusuario con más de seis millones de instalaciones. Es desarrollado como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso.
- ✓ GNU GPL: (General Public License o licencia pública general) es una licencia creada por la Fundación de Software Libre a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

  Oracle: es un sistema de gestión de base de datos relacional fabricado por Oracle Corporation. Es considerado como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones , estabilidad, escalabilidad, es multiplataforma.

- ✓ Microsoft SQL Server: Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.
- ✓ SQLite: SQLite es un sistema de gestión de bases de datos relacional que está contenida en una relativamente pequeña librería en C. SQLite, es un proyecto de dominio público.
- ✓ UNIX: Es un sistema operativo robusto, estable, multiusuario, multitarea, multiplataforma y con gran capacidad para gestión de redes.
- ✓ Linux: Sistema operativo creado a partir de las características de UNIX, es software libre. El sistema ha sido diseñado y programado por multitud de programadores alrededor del mundo. El núcleo del sistema sigue en continuo desarrollo.
- ✓ Mac OS X: Sistema operativo de la familia UNIX
- ✓ CGI: Common Gateway Interface (en castellano Interfaz Común de Pasarela, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (explorador Web) solicitar datos de un programa ejecutado en un servidor Web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor Web y una aplicación externa.
- ✓ Apache: Servidor de páginas Web de código abierto. La arquitectura del servidor Apache es muy modular. Mucha de la funcionalidad que podría considerarse básica para un servidor Web es provista por módulos.
- ✓ ISAPI: Es una interfaz para el desarrollo de aplicaciones de servidor Web, desarrollado por Process Software y Microsoft Corporation, que se utiliza en lugar de CGI.
- ✓ Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.
- ✓ Caso de uso: Es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.